Transparent Machine Learning: Theory and Computation

Ian C. Covert

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:
Su-In Lee, Chair
Kevin Jamieson
Ludwig Schmidt

Program Authorized to Offer Degree:
Computer Science & Engineering

©Copyright 2023
Ian C. Covert

University of Washington

Abstract

Transparent Machine Learning: Theory and Computation

Ian C. Covert

Chair of the Supervisory Committee:
Su-In Lee
Paul G. Allen School of Computer Science

Modern machine learning is driven primarily by black-box models, which provide superior performance but offer limited transparency into how predictions are made. For applications where it is important to understand how models make decisions, and to assist in model debugging and data-driven knowledge discovery, we require tools that can answer questions about what influences a model's behavior. This is the goal of *explainable machine learning* (XML), a subfield that develops tools to understand complex models from various perspectives, including feature importance, concept attribution and data valuation. This dissertation presents several contributions to the field of XML, with the main ideas organized into three parts: (i) a framework that enables a unified analysis of many current methods, including their links with information theory and model robustness; (ii) a suite of techniques to accelerate the computation of Shapley values, which are the basis of several popular algorithms; and (iii) a range of methods for performing feature selection with deep learning models, e.g., in unsupervised and adaptive settings. Many of these ideas are motivated by applications in computational biology and medicine, but they also represent fundamental tools and perspectives that are useful across a variety of domains.

TABLE OF CONTENTS

Page

Chapter 1: Introduction 1
1.1 Outline 2
1.2 Publications 4

Chapter 2: Preliminaries 6
2.1 Data and Models 6
2.2 Feature Subsets 6
2.3 Empirical and Population Risk Minimization 7
2.4 Information Theory 8
2.5 Cooperative Game Theory 11

Part I: Foundations of Explainable Machine Learning 14

Chapter 3: Explaining by Removing: A Unified Framework 15
3.1 Chapter Notes 15
3.2 Introduction 15
3.3 Background 19
3.4 Removal-Based Explanations 21
3.5 Feature Removal 27
3.6 Explaining Different Model Behaviors 34
3.7 Summarizing Feature Influence 37
3.8 Game-Theoretic Explanations 45
3.9 Information-Theoretic Explanations 54
3.10 A Cognitive Perspective on Removal-Based Explanations 67
3.11 Experiments 73
3.12 Discussion 90

Chapter 4: Robustness of Removal-Based Explanations . . . 94
4.1 Chapter Notes . . . 94
4.2 Introduction . . . 94
4.3 Background . . . 97
4.4 Preliminary Results . . . 100
4.5 Main Results: Feature Attribution Robustness . . . 108
4.6 Experiments . . . 110
4.7 Discussion . . . 119

Part II: Shapley Value Computation . . . 123

Chapter 5: Practical Shapley Value Estimation via Linear Regression . . . 124
5.1 Chapter Notes . . . 124
5.2 Introduction . . . 124
5.3 Background . . . 126
5.4 Linear Regression Estimators . . . 128
5.5 Estimator Properties . . . 132
5.6 Stochastic Cooperative Games . . . 138
5.7 Experiments . . . 142
5.8 Discussion . . . 145

Chapter 6: Amortized Shapley Value Estimation . . . 146
6.1 Chapter Notes . . . 146
6.2 Introduction . . . 146
6.3 Background . . . 147
6.4 FastSHAP . . . 150
6.5 Related Work . . . 153
6.6 Experiments . . . 154
6.7 Conclusion . . . 164

Chapter 7: Learning to Estimate Shapley Values with Vision Transformers . . . 165
7.1 Chapter Notes . . . 165
7.2 Introduction . . . 165
7.3 Related Work . . . 167

7.4 Background . . . 169
7.5 Evaluating Vision Transformers with Partial Information . . . 170
7.6 Learning to Estimate Shapley Values . . . 171
7.7 Experiments . . . 174
7.8 Conclusion . . . 182

Part III: Feature Selection with Deep Learning . . . 183

Chapter 8: Gene Selection for Spatial Transcriptomics . . . 184
8.1 Chapter Notes . . . 184
8.2 Introduction . . . 184
8.3 Results . . . 188
8.4 Discussion . . . 204
8.5 Methods . . . 206

Chapter 9: Maximizing Mutual Information for Dynamic Feature Selection . . . 221
9.1 Chapter Notes . . . 221
9.2 Introduction . . . 221
9.3 Problem Formulation . . . 223
9.4 Greedy Information Maximization . . . 224
9.5 Proposed Method . . . 227
9.6 Related Work . . . 232
9.7 Experiments . . . 234
9.8 Conclusion . . . 238

Chapter 10: Estimating Mutual Information for Dynamic Feature Selection . . . 239
10.1 Chapter Notes . . . 239
10.2 Introduction . . . 239
10.3 Problem Formulation . . . 242
10.4 Related Work . . . 243
10.5 Proposed Method . . . 244
10.6 Experiments . . . 250
10.7 Conclusion . . . 257

Chapter 11: Discussion 259

Appendix A: Explaining by Removing: A Unified Framework 307
A.1 Method Details 307
A.2 Additive Model Proofs 307
A.3 Axioms for Other Approaches 310
A.4 Consistency Proofs 313
A.5 Conditional Distribution Approximations 319
A.6 Information-Theoretic Connections in Regression 324
A.7 Experiment Details 327

Appendix B: Robustness of Removal-Based Explanations 332
B.1 Proofs: Prediction Robustness to Input Perturbations 332
B.2 Proofs: Prediction Robustness to Model Perturbations 336
B.3 Proofs: Summary Technique Robustness 338
B.4 Proofs: Main Results 355
B.5 The Role of Sampling When Removing Features 358
B.6 Feature Grouping 362
B.7 LIME Weighted Least Squares Linear Operator 364
B.8 Additional Results for Synthetic Experiments 369
B.9 Additional Results for Practical Implications 373
B.10 Implementation Details 379

Appendix C: Practical Shapley Value Estimation via Linear Regression 380
C.1 Main Proofs 380
C.2 Stochastic Cooperative Game Proofs 388
C.3 Experiment Details 390
C.4 Convergence Experiments 392

Appendix D: Amortized Shapley Value Estimation 397
D.1 FastSHAP Global Optimizer 397
D.2 Additive Efficient Normalization 399
D.3 Reducing Gradient Variance 400
D.4 FastSHAP Models and Hyperparameters 401

D.5 Additional Results for Image Experiments 403

Appendix E: Learning to Estimate Shapley Values with Vision Transformers 412
E.1 Attention Masking 412
E.2 Masked Training 413
E.3 Explainer Training Approach 415
E.4 Proofs 419
E.5 Datasets 426
E.6 Baseline Methods 427
E.7 Metrics Details 429
E.8 Additional Results 432
E.9 Qualitative Examples 448

Appendix F: Gene Selection for Spatial Transcriptomics 450

Appendix G: Maximizing Mutual Information for Dynamic Feature Selection 464
G.1 Proofs 464
G.2 Datasets 469
G.3 Baselines 471
G.4 Training Approach and Hyperparameters 475
G.5 Additional Results 478

Appendix H: Estimating Mutual Information for Dynamic Feature Selection 484
H.1 Proofs 484
H.2 Predictor Suboptimality 492
H.3 Training Algorithm 493
H.4 Datasets 495
H.5 Models 497
H.6 Baselines 499
H.7 Additional Results 500

ACKNOWLEDGMENTS

I have many people to thank for their support throughout my graduate studies. First and foremost, I would like to thank my advisor, Su-In Lee, who provided mentorship on many projects, helped guide me towards my career goals, and made sure I stayed motivated and encouraged during the nearly five years we worked together. I'm grateful for all that she has taught me and I hope to learn more from her in the years to come. I would also like to thank all the members of my committee for carefully examining my work.

I would like to thank several professors who helped me along the way: Jeff Bilmes, Kevin Jamieson, Maryam Fazel, and Sasha Aravkin. Their courses and the conversations we had outside of them equipped me with fundamental research tools, and provided valuable external perspectives that improved my work. I'm also grateful to Emily Fox, who I worked with during my first year at UW. I would also like to thank Alex Tank and Scott Lundberg, two early research mentors who showed me how to do good research. I am grateful to Uygar Sümbül, my first research mentor from Columbia; I am very fortunate that we moved to Seattle at the same time, and your levelheaded advice helped me navigate multiple challenges during my PhD. Several older students at UW gave advice that helped me throughout the PhD, but especially Rahul Kidambi, Krishna Pillutla, John Thickstun, and Chris Aicher.

Su-In's lab has been a fantastic environment for me during my PhD, and that is largely thanks to the students. I made many friends here, and I learned a tremendous amount both from them and with them. From one generation, I would like to thank Safiye Celik, Nicasia Beebee-Wang, Pascal Sturmfels, Hugh Chen, Joseph Janizek, Gabriel Erion, Ethan Weinberger, Alex DeGrave, Wei Qiu and Ayse Dincer. From the next generation, I would like to thank Chanwoo Kim, Chris Lin, Soham Gadgil and Mingyu Lu. My thesis would not

be what it is without all of you.

I did two internships during my PhD that ended up being important learning experiences. I would like to thank Jiening Zhan and Ming Jack Po from Google for hiring me, and for giving me an ideal environment to learn and explore new research directions. I would also like to thank Dianqi Li and Martin Lorilla from Citadel, who gave me the chance to try something entirely new for a summer, and who let me spend a large part of my time reading papers and messing around on the whiteboard.

I have many collaborators to thank from throughout the PhD. In addition to those mentioned above, I would like to thank Mukund Sudarshan, Neil Jethani and Rajesh Ranganath from NYU; I am very lucky we met (virtually) at AISTATS, and getting FastSHAP to work was one of the most exciting parts of my PhD. I would also like to thank Rohan Gala, Tim Wang, Karel Svoboda, Nathan White, Ali Shojaie, and my entire team from Google.

From the computer science school at UW, thank you to Elise Dorough for supporting me and so many other students in our program. Thank you to Sandy Kaplan for your valuable writing feedback – I tell everyone I can that they should try working with you. And thank you to the technology support staff who maintain all our systems, and who I've bugged far too many times on nights and weekends.

I was lucky to make many wonderful friends during my time in Seattle. From our "UC Sonoma" group, living with Ofir Press, Sam Ainsworth, Ivan Evtimov, Sally Dong and Amanda Baughan made the COVID-19 lockdown surprisingly fun – I will always remember those times. From the ML office, Jennifer Brennan, Daniel Jiang, Andrew Wagenmaker, Rahul Nadkarni, Willie Agnew, and Jonathan Hayase; and from the greater UW community, Mitchell Wortsman, Sarah Pratt, Peter West, Sahil Verma, Naveena Karusala, Brian Hou, Matt Wallingford, Gabriel Ilharco, Jared Roesch, Eunice Jun, Jesse Dodge, Steve Mussman, Edward Misback, Nathan Hatch, and certainly many others.

Playing sports helped me stay happy and healthy during the PhD, and I am grateful

for the many people I befriended this way. From playing squash, Trevor Perrier, David Wadden, Mitchell and Krishna. From playing tennis at UW, Everet Wang, Curtis Wang, Jose Verdezoto, Matthew Chun, Noon Nikomborirak, Liang Luo, Kuikui Liu and Elizabeth Clark; and from playing at the Greenlake courts, Elmer, DeVaughn, Sonna, Alex and James. Finally, thank you to Jifan Zhang for teaching me how to play badminton properly – although everyone can still tell from my shots that I'm a tennis player.

I would also like to thank my family of housemates in Seattle, but especially Jake, who I lived with the entire time; maybe you should move too so we can keep the streak going. Thank you as well to Izzy, Liv, Ellen and Rina for all the fun times we had together. From before my time in Seattle, I would like to thank the friends who enjoyed this journey with me. Pierre-Alexander Low was a friend I hoped to have for life, and I think of him nearly every day. I'm also grateful for Antonio, Mayank, Sahir, Eunice, Josh, Hari, Rhea, Ankit, Kevin and Mookie, who have been a refreshing source of joy and reality over the years.

From earlier in my life, I have several mentors and teachers to thank. Thank you to Hubert Condemi for teaching me tennis and so much more about life. Thank you to David Surrenda, a longtime family friend who first suggested that I do a PhD, and to Pietro Cinquegrana, who re-ignited my interest in reading non-fiction books. Thank you to two of my high school teachers, Netta Maclean and Tarly Manak, whose kind words went a long way in encouraging my interest in math. And thank you to Donna: you were strict with me and Darcy, but you taught us to value education and become responsible people.

Finally, I thank my family – I could not have done this without your support. Thank you especially to Sarah and Liz, who coached me through some difficult times in my PhD. My sister Darcy puts up with me, and at times perhaps even appreciates my foibles. My parents – Mary and Derek – gave me every opportunity to succeed in life, and I cannot thank them enough for that. I especially thank my dad for making me and Darcy a priority in his life, and I wish he were here to celebrate with us today.

DEDICATION

To my family.

Chapter 1

INTRODUCTION

Modern machine learning (ML) is driven primarily by black-box models, with deep neural networks enabling recent advances for most data modalities (images, language, audio, proteins, graphs) and gradient boosted trees being widely used for tabular data. These models are characterized by their large number of parameters and internal calculations, which play a crucial role in learning and representing complex functions. ML researchers and practitioners increasingly rely on such models because they offer superior performance, but an important drawback is their limited transparency: compared to their traditional counterparts, these models make it far more difficult to identify the features, concepts and training data that influence their behavior. For example, it can be difficult to understand the prediction generated for a single patient's diagnosis.

A model's accuracy is often of paramount importance, but there are certain applications where it is also important to understand how our models work. These include applications in finance, healthcare and criminal justice, because model transparency can help identify unwanted dependencies, establish user or organizational trust, determine whether models operate like expert decision-makers, and understand and act on incorrect predictions. Furthermore, in scientific applications like computational biology, transparent models enable us to pursue the "information goal" of creating models: to learn how the response variable is naturally associated with the input variables, and improve our understanding of the underlying system (Breiman et al., 2001).

In the debate around model transparency, the conventional wisdom is that we face a trade-off between interpretability and accuracy.[1] Some argue that the trade-off does not

[1] *The Great AI Debate* at NeurIPS 2017, `https://www.youtube.com/watch?v=93Xv8vJ2acI`

exist, claiming that we can achieve near-optimal performance with "inherently interpretable" models (Rudin, 2019); this is often true for simple tabular datasets, but it is less common for complex data modalities like images and language. Here, we adopt a more accommodating position: given that black-box models currently offer the best performance and are already widely deployed, we explore whether it is possible to obtain sufficient insights from *any model*. In doing so, we develop a toolset that is largely indifferent to the model's internal mechanisms, or which is *model-agnostic*, and that can therefore operate even with today's most performant black-box models.

This goal is shared by many works in the subfield of *explainable machine learning* (XML), and significant progress has been made in recent years. Already, XML tools have been used to understand risk factors for novel diseases (Razavian et al., 2020; Snider et al., 2021), accelerate the discovery of mathematical conjectures (Davies et al., 2021), identify protein binding sites with limited training data labels (Gligorijević et al., 2021), audit faulty medical diagnosis systems (DeGrave et al., 2021) and derive new insights from functional systems (Ting et al., 2017; Sundararajan et al., 2017). These early successes point to the potential for such tools, but there is also room for progress, particularly in the theory underlying these methods and the computational procedures that make them efficient in practice. This thesis presents several works conducted during my PhD that aim to address these challenges.

1.1 Outline

This thesis contains most of the projects completed during my PhD, all of which are related to the central theme of transparent machine learning. We begin by setting up notation and several preliminary ideas in Chapter 2. Next, each chapter is based on a single first-author publication, which is in some cases shared with co-first authors. The individual works have been modified to improve their cohesion in a single document, but no new information is provided here and the papers can also be read individually. The works are organized into three parts, as described below.

Part I: Foundations of XML We first discuss a perspective that unifies a significant portion of the literature: that many current methods are based on a principle of *explaining by removing*, or quantifying the impact of removing features from a model. We describe a framework in which these methods differ based on three implementation choices, and we identify these choices for 26 existing algorithms (Chapter 3). Based on this perspective, we conduct a unified analysis of such methods and identify connections with information theory, game theory and cognitive psychology. We then explore the robustness properties of such methods, and we derive new results characterizing their robustness to both input and model perturbations (Chapter 4).

Part II: Shapley value computation Next, we explore one of the most widely used tools in XML: Shapley values, a game-theoretic credit allocation technique. These are the basis of one of the most popular feature attribution methods, SHAP (Lundberg and Lee, 2017), as well as a prominent data valuation technique (Ghorbani and Zou, 2019), but they are notoriously difficult to calculate. There are a range of methods to accelerate their calculation (Chen et al., 2022), and we discuss two here: approximations based on weighted linear regression (Chapter 5), and approximations based on amortized optimization with deep learning (Chapter 6, Chapter 7).

Part III: Feature selection with deep learning Finally, feature selection offers an alternative direction to provide transparency while also reducing feature acquisition costs. It is seemingly difficult to implement with deep learning due to the high cost of training multiple models with different feature sets, but we explore how it can be performed using differentiable layers that block feature information from entering a network (Chapter 8). We then discuss how to apply these ideas in an adaptive setting, where we select features separately for each prediction based on the currently available information (Chapter 9, Chapter 10).

Finally, we provide a concluding discussion in Chapter 11.

1.2 Publications

For the committee's convenience, the publications contributing to this thesis are the following (* indicates equal contribution):

- Ian Covert, Scott Lundberg, Su-In Lee. Explaining by Removing: A Unified Framework for Model Explanation. In *Journal of Machine Learning Research, JMLR*, 2021.

- Ian Covert, Su-In Lee. Improving KernelSHAP: Practical Shapley Value Estimation via Linear Regression. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2021.

- Neil Jethani*, Mukund Sudarshan*, Ian Covert*, Su-In Lee, Rajesh Ranganath. FastSHAP: Real-Time Shapley Value Estimation. In *International Conference on Learning Representations, ICLR*, 2022.

- Ian Covert, Rohan Gala, Tim Wang, Karel Svoboda, Uygar Sümbül*, Su-In Lee*. Predictive and Robust Gene Selection for Spatial Transcriptomics. In *Nature Communications*, 2023.

- Ian Covert*, Chanwoo Kim*, Su-In Lee. Learning to Estimate Shapley Values with Vision Transformers. In *International Conference on Learning Representations, ICLR*, 2023.

- Ian Covert, Wei Qiu, Mingyu Lu, Nayoon Kim, Nathan White, Su-In Lee. Learning to Maximize Mutual Information for Dynamic Feature Selection. In *International Conference on Machine Learning, ICML*, 2023.

- Chris Lin*, Ian Covert*, Su-In Lee. On the Robustness of Removal-Based Feature Attributions. *Preprint*, 2023.

- Soham Gadgil*, Ian Covert*, Su-In Lee. Estimating Conditional Mutual Information for Dynamic Feature Selection. *Preprint*, 2023.

There are several other projects completed during my PhD that I do not discuss here, but which were equally important for my learning process and are relevant contributions to the area of transparent machine learning. These include:

- Ian Covert, Balu Krishnan, Imad Njam, Jiening Zhan, Matthew Shore, John Hixson, Ming Jack Po. Temporal Graph Convolutional Networks for Automatic Seizure Detection. In *Machine Learning for Healthcare, MLHC*, 2019.

- Ian Covert, Uygar Sümbül, Su-In Lee. Deep Unsupervised Feature Selection. *Preprint*, 2019.

- Ian Covert, Scott Lundberg, Su-In Lee. Understanding Global Feature Contributions With Additive Importance Measures. In *Neural Information Processing Systems, NeurIPS*, 2020.

- Alex Tank*, Ian Covert*, Nicholas Foti, Ali Shojaie, Emily Fox. Neural Granger Causality. In *Transactions on Pattern Analysis and Machine Intelligence, TPAMI*, 2021.

- Hugh Chen*, Ian Covert*, Scott Lundberg, Su-In Lee. Algorithms to Estimate Shapley Value Feature Attributions. In *Nature Machine Intelligence*, 2023.

Chapter 2

PRELIMINARIES

We use this chapter to define notation used throughout the thesis, and to introduce several mathematical ideas that appear in multiple projects.

2.1 Data and Models

Many of the ideas discussed here focus on supervised ML models that predict a response variable $\boldsymbol{y}$ given an input variable $\boldsymbol{x}$. The symbols $\boldsymbol{x}, \boldsymbol{y}$ are used to refer to random variables, and x, y denote specific values. We use $p(\boldsymbol{x}, \boldsymbol{y})$ to represent the joint data distribution, and $p(\boldsymbol{x})$ is the data distribution with support on $\mathcal{X}$. We assume for simplicity that the input space is $\mathcal{X} \subseteq \mathbb{R}^d$, but several ideas presented here can be generalized to discrete feature spaces. As for the response variable, we can have either $\boldsymbol{y} \in \mathbb{R}$ for regression problems or $\boldsymbol{y} \in [K] \equiv \{1, \ldots, K\}$ for classification tasks.

The model is a function $f : \mathcal{X} \mapsto \mathcal{Y}$ with output space $\mathcal{Y} = \mathbb{R}$ for regression, or $\mathcal{Y} = \Delta^{K-1}$ for classification, where Δ^{K-1} represents the $K-1$ dimensional probability simplex. In the classification setting, we let $f_y(\boldsymbol{x}) \in [0, 1]$ denote the probability for the yth class. We indicate parameters when the model is being learned, e.g., $f(\boldsymbol{x}; \theta)$, but we omit parameters when the model is fixed. In most cases we require no assumptions about the model type that f represents; in cases where it is learned, it can usually be any differentiable model class (e.g., a MLP, CNN or vision transformer).

2.2 Feature Subsets

Many ideas discussed here involve reasoning about subsets of features. We therefore consider that the model input consists of d separate features, denoted by $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$. We

typically have scalar features $\boldsymbol{x}_i \in \mathbb{R}$, but for images we group pixels into patches; for example, when working with vision transformers we use patches of size $\boldsymbol{x}_i \in \mathbb{R}^{16\times16\times3}$. We denote a subset of feature indices as $S \subseteq [d] \equiv \{1, \ldots, d\}$, and we define a feature subset as $\boldsymbol{x}_S \equiv \{\boldsymbol{x}_i : i \in S\}$. This allows us to consider the marginal distribution $p(\boldsymbol{x}_S)$, as well as the conditional distributions for the response $p(\boldsymbol{y} \mid \boldsymbol{x}_S)$ or for the features $p(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S)$, where $\bar{S} = [d] \setminus S$ denotes the set complement. We use $\mathcal{P}(d)$ to denote the power set of $[d]$, and as a shorthand we often write $S \cup \{i\}$ as $S \cup i$ and $S \setminus \{i\}$ as $S \setminus i$ for singleton sets.

With abuse of notation, we will sometimes write a model's output given a subset of features by $f(\boldsymbol{x}_S)$. The precise definition of $f(\boldsymbol{x}_S)$ will be made clear by the context; for example, Chapter 3 discusses techniques for simulating feature removal with pre-trained models (e.g., setting unknown features to a neutral value), and Chapter 9 discusses training models with masked inputs to represent unknown feature values.

2.3 Empirical and Population Risk Minimization

The objective functions we use when training ML models can typically be expressed in two ways: as the average over a finite dataset, or as the expectation across an underlying data distribution. Consider a model $f(\boldsymbol{x})$ whose predictions $\hat{y}$ are scored with a loss function $\ell(\hat{y}, y)$. Given a finite set of labeled examples $\mathcal{D} = \{(x^j, y^j)\}_{j=1}^n$, we can write the *empirical risk* over the dataset as follows:

$$R_{\text{emp}}(f) = \mathop{\mathbb{E}}_{\mathcal{D}}[\ell(f(\boldsymbol{x}), \boldsymbol{y})] = \frac{1}{n}\sum_{j=1}^{n} \ell\left(f(x^j), y^j\right).$$

Minimizing this objective with respect to f is known as *empirical risk minimization* (Vapnik, 1999), and it is what we typically do in practice. However, minimizing the empirical risk is not our true goal, and focusing exclusively on minimizing $R_{\text{emp}}(f)$ may yield a solution that fails to generalize. Instead, our true goal is to minimize the *population risk*, which is the loss

taken in expectation over the data distribution $p(\boldsymbol{x}, \boldsymbol{y})$:

$$R(f) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \left[\ell(f(\boldsymbol{x}), \boldsymbol{y}) \right].$$

The relevance of the population risk in this thesis is that it provides a perspective to reason about our objective functions, and to understand what a model should learn when fit to near-optimality. The core idea is that for any loss function, we can fix the input value $x \in \mathcal{X}$ and consider which prediction $\hat{y} = f(x)$ minimizes the loss in expectation. For example, for classification models trained with cross entropy loss, we can see that the optimal prediction is the true probability distribution, or $f(x) = p(\boldsymbol{y} \mid x)$; this property is shared by several other *strictly proper scoring functions* (Gneiting and Raftery, 2007), which are loss functions that encourage the true probabilistic prediction. Similarly, for regression models trained with mean squared error loss, we can see that the optimal prediction is the response variable's conditional expectation, or $f(x) = \mathbb{E}[\boldsymbol{y} \mid x]$.

We use this perspective to reason about several loss functions in this thesis. For example, Chapter 3 reasons about standard loss functions for models trained with randomly masked inputs. Chapters 6 and 7 take a similar approach when the model is trained with a custom loss function to predict Shapley values. Finally, Chapters 9 and 10 apply a similar approach for objective functions related to the information associated with unobserved features.

2.4 Information Theory

For several of the methods presented in this thesis, it is helpful to define the basic tools of information theory and consider their links with common model training procedures. First, assuming a discrete response variable $\boldsymbol{y} \in [K]$ with probability mass function $p(\boldsymbol{y})$, the entropy $H(\boldsymbol{y})$ is defined as follows:

$$H(\boldsymbol{y}) = -\sum_{y=1}^{K} p(y) \log p(y).$$

This provides a measure of the uncertainty in $\boldsymbol{y}$'s outcome, with the maximum value being $H(\boldsymbol{y}) = \log K$ when $p(\boldsymbol{y})$ is uniform and the minimum being $H(\boldsymbol{y}) = 0$ when $\boldsymbol{y}$ is deterministic. Observing the input features $\boldsymbol{x}$ is helpful because it provides information about $\boldsymbol{y}$ and often reduces our uncertainty. Given the observed values $\boldsymbol{x} = x$, we have the updated conditional distribution $p(\boldsymbol{y} \mid x)$, which yields the entropy

$$H(\boldsymbol{y} \mid x) = -\sum_{y=1}^{K} p(y \mid x) \log p(y \mid x).$$

If we want to consider how much $\boldsymbol{x}$ reduces our uncertainty about $\boldsymbol{y}$ *on average*, we can take the expectation of the equation above. This is known as the conditional entropy (Cover and Thomas, 2012):

$$H(\boldsymbol{y} \mid \boldsymbol{x}) = \int_{\mathcal{X}} p(x) H(\boldsymbol{y} \mid x) dx.$$

We may expect that obtaining new information should reduce our uncertainty, and this is true—at least in expectation. The difference between the unconditional and conditional entropy defines the *mutual information* between $\boldsymbol{x}$ and $\boldsymbol{y}$, which can be viewed as a (non-negative) expected KL divergence:

$$I(\boldsymbol{y}; \boldsymbol{x}) \equiv H(\boldsymbol{y}) - H(\boldsymbol{y} \mid \boldsymbol{x}) = \int_{\mathcal{X}} p(x) D_{\mathrm{KL}}\big(p(\boldsymbol{y} \mid x) \,||\, p(\boldsymbol{y})\big) dx \geq 0.$$

The mutual information $I(\boldsymbol{y}; \boldsymbol{x})$ therefore represents how much observing $\boldsymbol{x}$ reduces our uncertainty about $\boldsymbol{y}$, or intuitively, how valuable $\boldsymbol{x}$ is as a predictor. The non-negativity property implies that we should always seek additional information to reduce our uncertainty. We can similarly define $I(\boldsymbol{y}; \boldsymbol{x}_S)$ for specific feature subsets; e.g., if the feature set $\boldsymbol{x}_S$ is just as informative as the complete feature set $\boldsymbol{x}$, we should expect $I(\boldsymbol{y}; \boldsymbol{x}_S) \approx I(\boldsymbol{y}; \boldsymbol{x})$. Note that by the data processing inequality, it is impossible to have $I(\boldsymbol{y}; \boldsymbol{x}_S) > I(\boldsymbol{y}; \boldsymbol{x})$ (Cover and Thomas, 2012).

Next, we consider how these definitions relate to common procedures for training ML models. Classification models are typically trained with cross entropy loss, and due to cross

entropy loss being a strictly proper scoring function (see Section 2.3), the optimal prediction for each input is the response variable's true conditional distribution. If we consider what loss value this achieves in expectation across the data distribution, we can observe the following:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}[\ell(p(\boldsymbol{y} \mid \boldsymbol{x}), \boldsymbol{y})] = \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \mathop{\mathbb{E}}_{p(\boldsymbol{y}|\boldsymbol{x})}[-\log p(\boldsymbol{y} \mid \boldsymbol{x})] = H(\boldsymbol{y} \mid \boldsymbol{x}).$$

In other words, the expected loss is the conditional entropy $H(\boldsymbol{y} \mid \boldsymbol{x})$. This interpretation of a classification model's loss lends itself to several practical use cases, all of which are explored in this thesis:

1. A flexible classifier trained with a large dataset can be interpreted as approximating the true conditional distribution, or $f_y(x) \approx p(y \mid x)$. If we repeat this process for multiple subsets, we can interpret the difference in predictions as the difference in the response variable's conditional probability.

2. A classifier's expected loss provides an approximation of the conditional entropy $H(\boldsymbol{y} \mid \boldsymbol{x})$, and it is related to the mutual information due to the relationship $I(\boldsymbol{y} \mid \boldsymbol{x}) = H(\boldsymbol{y}) - H(\boldsymbol{y} \mid \boldsymbol{x})$. The expected loss can be compared between models trained on different feature sets, and this represents differences in the mutual information (because $H(\boldsymbol{y})$ is constant).

3. Learning a feature set that achieves minimum loss can be interpreted as finding features with minimum conditional entropy $H(\boldsymbol{y} \mid \boldsymbol{x}_S)$, or equivalently, with maximum mutual information $I(\boldsymbol{y}; \boldsymbol{x}_S)$. This provides a lens to interpret feature selection procedures via which features they choose when fit optimally.

When we consider regression models fit with continuous response variables $\boldsymbol{y} \in \mathbb{R}$, these ideas from information theory do not apply directly. However, analogous ideas arise from training with squared error loss, and these can be framed in terms of variance rather than entropy. The key property is that squared error loss yields the conditional expectation as an

optimal prediction, or $f(x) = \mathbb{E}[\boldsymbol{y} \mid x]$. Then, the entropy $H(\boldsymbol{y})$ is replaced by the variance $\text{Var}(\boldsymbol{y})$, and the conditional entropy $H(\boldsymbol{y} \mid \boldsymbol{x})$ is replaced by the expected conditional variance $\mathbb{E}_{p(\boldsymbol{x})}[\text{Var}(\boldsymbol{y} \mid \boldsymbol{x})]$. The mutual information $I(\boldsymbol{y}; \boldsymbol{x}_S)$ does not have a direct analogue, but we can identify a similar property of the uncertainty reducing in expectation due to the *law of total variance*:

$$\text{Var}(\boldsymbol{y}) - \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[\text{Var}(\boldsymbol{y} \mid \boldsymbol{x})] = \text{Var}_{p(\boldsymbol{x})}(\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]) \geq 0.$$

Interpreting a regression model as $f(x) \approx \mathbb{E}[\boldsymbol{y} \mid x]$ enables the same use cases described above for classification models. For example, we can compare the loss between models trained on different features to approximate differences in conditional variance, and we can aim to find a small feature set that achieves the lowest possible conditional variance.

2.5 Cooperative Game Theory

Finally, we briefly introduce cooperative game theory and one of its most famous ideas: the Shapley value. Cooperative game theory is the study of *coalitional games*, wherein subsets of players opt to participate or not and thereby generate a certain amount of profit. The central questions in this field focus on how to design fair credit allocation schemes and predict which players will participate (Narahari, 2014); for example, allocations should exceed what players can achieve in isolation, or else they will not be inclined to participate. For our purpose, these games are effectively set functions of the form $v : \mathcal{P}(d) \mapsto \mathbb{R}$, where the input is a set of players and the output represents the profit. For example, in the context of a company, $v([d])$ is the profit when all employees choose to work (the *grand coalition*), $v(\varnothing)$ is the profit when no one works (the *null coalition*), and $v(S)$ for $S \subseteq [d]$ is the profit achieved by an arbitrary set of employees.

The Shapley value (Shapley, 1953) considers the situation where all employees choose to work, and it defines a principled approach to calculate each player's contribution. The idea is that a player's impact is defined by the change in value caused by their absence, or $v(S \cup i) - v(S)$ (also called i's *marginal contribution*), but this difference depends on the

preceding subset $S \subseteq [d] \setminus i$. The Shapley value therefore averages this contribution across all possible preceding subsets, and using a carefully chosen weighting function. The Shapley value for the ith player, denoted by $\phi_i(v) \in \mathbb{R}$, is defined as follows:

$$\phi_i(v) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \big(v(S \cup i) - v(S)\big).$$

We can understand this equation with multiple perspectives. First, the Shapley value $\phi_i(v)$ is equivalent to the average contribution across all player orderings; that is, we can imagine enumerating all possible player orderings, calculating i's marginal contribution within each ordering (i.e., the difference in value before and after including i), and then averaging the contributions (Shapley, 1953). Second, the Shapley value is the unique credit allocation technique to define a set of fairness axioms. There are in fact multiple sets of axioms that uniquely define the Shapley value (Monderer et al., 2002), but we present one such set below:

1. (Efficiency) The allocations $\phi_1(v), \ldots, \phi_d(v)$ sum to the difference in value between the grand coalition and the empty coalition:

$$\sum_{i \in [d]} \phi_i(v) = v([d]) - v(\varnothing).$$

2. (Linearity) If we consider two games v, v' and their allocations $\phi_i(v)$ and $\phi_i(v')$, then the cooperative game defined as their linear combination $\alpha v + \alpha' v'$ with $\alpha, \alpha' \in \mathbb{R}$ has the following allocations:

$$\phi_i(\alpha v + \alpha' v') = \alpha \phi_i(v) + \alpha' \phi_i(v') \;\; \forall \;\; i \in [d].$$

3. (Symmetry) Two players i, j that make equal marginal contributions to all coalitions

receive the same allocation:

$$v(S \cup i) = v(S \cup j) \ \ \forall \ \ S \subseteq [d] \implies \phi_i(v) = \phi_j(v).$$

4. (Null player) A player i that makes zero marginal contribution receives zero allocation:

$$v(S \cup i) = v(S) \ \ \forall \ \ S \subseteq [d] \implies \phi_i(v) = 0.$$

5. (Marginalism) For two games v, v' where all players have identical marginal contributions, the players receive equal allocations:

$$v(S \cup i) - v(S) = v'(S \cup i) - v'(S) \ \ \forall \ \ (i, S) \implies \phi_i(v) = \phi_i(v') \ \ \forall \ \ i \in [d].$$

In machine learning, the Shapley value has been adopted to quantify player contributions in several types of coalitional games. For example, SHAP focuses on the prediction for a single input given subsets of features (Lundberg and Lee, 2017), and Data Shapley considers the test set accuracy given subsets of training data examples (Ghorbani and Zou, 2019). In both cases, the Shapley values $\phi(v) = (\phi_1(v), \ldots, \phi_d(v)) \in \mathbb{R}^d$ are difficult to calculate due to their exponential complexity, which represents an obstacle to their usage in practice. Several chapters in this thesis focus on simplifying their calculation to enable efficient approximations (Chapter 5, Chapter 6 and Chapter 7).

Part I

FOUNDATIONS OF EXPLAINABLE MACHINE LEARNING

Chapter 3

EXPLAINING BY REMOVING: A UNIFIED FRAMEWORK

3.1 Chapter Notes

This chapter presents joint work with Scott Lundberg and Su-In Lee that was published in the Journal of Machine Learning Research (JMLR) in 2021. The content of this chapter is a minor variant of the published version of the paper. The main contribution of this work is to introduce a framework that encompasses many existing model explanation methods, and which enables a unified analysis of their theoretical foundations.

3.2 Introduction

The proliferation of black-box models has made machine learning (ML) explainability increasingly important, and researchers have now proposed a variety of model explanation approaches (Zeiler and Fergus, 2014; Ribeiro et al., 2016; Lundberg and Lee, 2017). Despite progress in the field, the relationships and trade-offs among these methods have not been rigorously investigated, and researchers have not always formalized their fundamental ideas about how to interpret models (Lipton, 2018). This makes the interpretability literature difficult to navigate, and raises questions about whether existing methods relate to human processes for explaining complex decisions (Miller et al., 2017; Miller, 2019).

Here, we present a comprehensive new framework that unifies a substantial portion of the model explanation literature. Our framework is based on the observation that many methods can be understood as *simulating feature removal* to quantify each feature's influence on a model. The intuition behind these methods is similar (depicted in Figure 3.1), but each one takes a slightly different approach to the removal operation: some replace features with neutral values (Zeiler and Fergus, 2014; Petsiuk et al., 2018), others marginalize over a

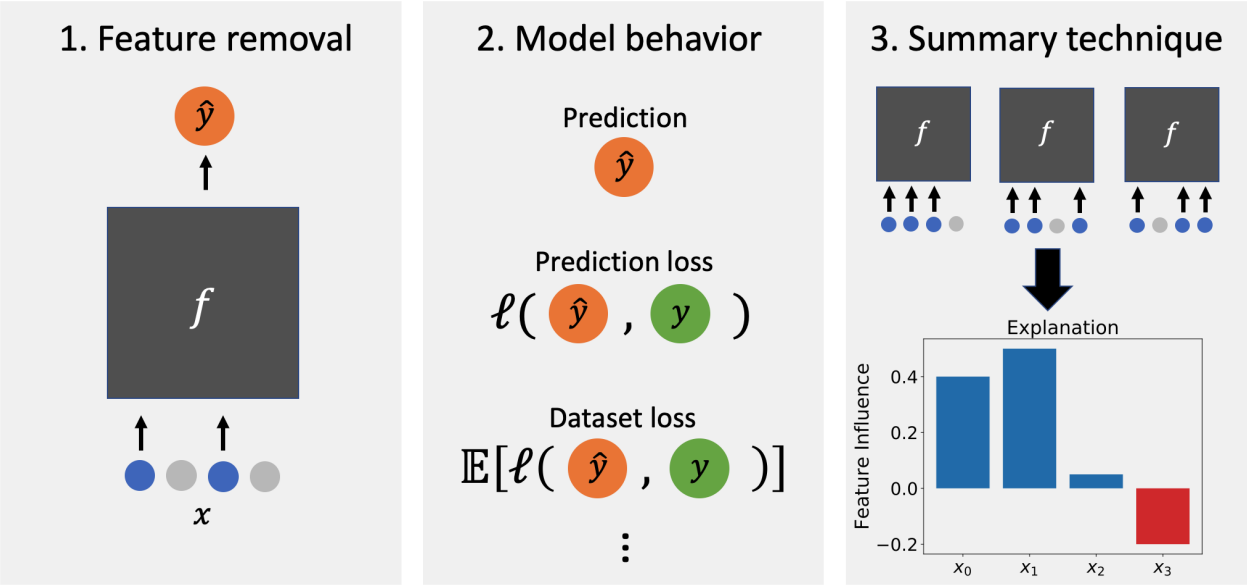


Figure 3.1: A unified framework for *removal-based explanations*. Each method is determined by three choices: how it removes features, what model behavior it analyzes, and how it summarizes feature influence.

distribution of values (Strobl et al., 2008; Lundberg and Lee, 2017), and still others train separate models for each subset of features (Lipovetsky and Conklin, 2001; Štrumbelj et al., 2009). These methods also vary in other respects, as we describe below.

We refer to this class of approaches as *removal-based explanations* and identify 26[1] existing methods that rely on the feature removal principle, including several of the most widely used methods (SHAP, LIME, Meaningful Perturbations, permutation tests). We then develop a framework that shows how each method arises from various combinations of three choices: (i) how the method removes features from the model, (ii) what model behavior the method analyzes, and (iii) how the method summarizes each feature's influence on the model. By characterizing each method in terms of three precise mathematical choices, we are able to systematize their shared elements and show that they all rely on the same fundamental

[1]This total count does not include minor variations on the approaches we identified. The number of methods in our framework has also increased since the original publication.

approach—feature removal.

The model explanation field has grown significantly in the past decade, and we take a broader view of the literature than existing unification theories. Our framework's flexibility lets us establish links between disparate classes of methods (e.g., computer vision-focused methods, global methods, game-theoretic methods, feature selection methods) and show that the literature is more interconnected than previously recognized. Exposing these relationships makes the literature more coherent, and it simplifies the process of reasoning about the benefits of each method by showing that their differences often amount to minor, interchangeable design choices.

To better understand the unique advantages of each method, we thoroughly analyze our framework's theoretical foundation by examining its connections with related fields. In particular, we find that cognitive psychology, cooperative game theory and information theory are intimately connected to removal-based explanations and help shed light on the trade-offs between different approaches. The extent of these links is perhaps surprising, because few methods explicitly reference these related fields.

Our approach yields many new results and provides a strong theoretical foundation for understanding existing methods and guiding future work. Our contributions include:

1. We present a unified framework that characterizes 26 existing explanation methods and formalizes a new class of removal-based explanations. The framework integrates classes of methods that may have previously appeared disjoint, including local and global approaches, as well as feature attribution and feature selection methods. In our experiments, we develop and compare 60+ new explanation approaches by mixing and matching the choices that methods make in each dimension of our framework.

2. We develop new mathematical tools to represent different approaches for removing features from ML models. Then, by incorporating an underlying data distribution, we argue that marginalizing out features using their conditional distribution is the only approach that is consistent with standard probability axioms. Finally, we prove that

several alternative choices approximate this approach and that this approach gives all removal-based explanations an information-theoretic interpretation.

3. We demonstrate that all removal-based explanations are implicitly tied to cooperative game theory, and we leverage decades of game theory research to highlight advantages of the Shapley value over alternative allocation strategies. Building on these findings, we also show that several feature attribution techniques are generalized by LIME (Ribeiro et al., 2016), and that several feature selection techniques are generalized by the Masking Model approach (Dabkowski and Gal, 2017).

4. We consult social science research to understand the intuition behind feature removal as an approach to model explanation. We find that feature removal is a simple application of subtractive counterfactual reasoning, or, equivalently, of Mill's method of difference from the philosophy of scientific induction (Mill, 1884).

The chapter is organized as follows. We begin with background on the model explanation problem and a review of prior work (Section 3.3), and we then give an overview of our framework (Section 3.4). We then present our framework in detail, describing how methods remove features (Section 3.5), formalizing the model behaviors analyzed by each method (Section 3.6), and examining each method's approach to summarizing feature influence (Section 3.7). We next explore connections to related fields. First, we describe our framework's relationship with cooperative game theory (Section 3.8). Next, we prove that under certain conditions, removal-based explanations analyze the information communicated by each feature (Section 3.9). Then, we refer to the psychology literature to establish a cognitive basis for removal-based explanations (Section 3.10). In our experiments, we provide empirical comparisons between existing methods and new combinations of existing approaches, as well as a discussion of our framework's implications for common explainability metrics (Section 3.11). Finally, we recap and conclude our discussion (Section 3.12).

3.3 Background

Here, we introduce the model explanation problem and briefly review existing approaches and related unification theories.

3.3.1 Preliminaries

Consider a supervised ML model f that is used to predict a response variable $\boldsymbol{y} \in \mathcal{Y}$ using the input $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) \in \mathcal{X}$, where each $\boldsymbol{x}_i \in \mathcal{X}_i$ represents an individual feature, such as a patient's age or blood pressure. ML interpretability broadly aims to provide insight into how models make predictions. This is particularly important when f is a complex model like a neural network or a decision forest. The most active area of recent research is *local interpretability*, which explains individual predictions, such as individual patient diagnoses (e.g., Ribeiro et al., 2016; Lundberg and Lee, 2017; Sundararajan et al., 2017); in contrast, *global interpretability* explains the model's behavior across the entire dataset (e.g., Breiman, 2001; Owen, 2014; Covert et al., 2020). Both problems are often addressed using *feature attribution*, where scores are assigned to quantify each feature's influence. However, recent work has also proposed the strategy of *local feature selection* (Chen et al., 2018a), and other papers have introduced methods to isolate sets of relevant features (Zhou et al., 2014; Fong and Vedaldi, 2017; Dabkowski and Gal, 2017).

Whether the aim is local or global interpretability, explaining the inner workings of complex models is fundamentally difficult, so it is no surprise that researchers continue to devise new approaches. Commonly cited categories of approaches include perturbation-based methods (e.g., Zeiler and Fergus, 2014; Lundberg and Lee, 2017), gradient-based methods (e.g., Simonyan et al., 2013; Montavon et al., 2017; Sundararajan et al., 2017; Selvaraju et al., 2017), more general propagation-based methods (e.g., Springenberg et al., 2014; Bach et al., 2015; Kindermans et al., 2017; Zhang et al., 2018), and inherently interpretable models (e.g., Zhou et al., 2016; Rudin, 2019). However, these categories refer to loose collections of approaches that seldom share a precise mechanism.

Among the various approaches in the literature, many methods generate explanations by considering some class of perturbation to the input and the corresponding impact on the model's predictions. Certain methods consider infinitesimal perturbations by calculating gradients[2] (Simonyan et al., 2013; Sundararajan et al., 2017; Smilkov et al., 2017; Erion et al., 2019; Xu et al., 2020), but there are many possible input perturbations (Zeiler and Fergus, 2014; Ribeiro et al., 2016; Fong and Vedaldi, 2017; Lundberg and Lee, 2017). Our work is based on the observation that many perturbation strategies can be understood as simulating feature removal.

3.3.2 Related Work

Prior work has made notable progress in exposing connections among disparate explanation methods. Lundberg and Lee (2017) proposed the unifying framework of *additive feature attribution methods* and showed that LIME, DeepLIFT, LRP and QII are all related to SHAP (Bach et al., 2015; Ribeiro et al., 2016; Datta et al., 2016; Shrikumar et al., 2017). Similarly, Ancona et al. (2018) showed that Grad $\times$ Input, DeepLIFT, LRP and Integrated Gradients can all be understood as modified gradient backpropagations. Covert et al. (2020) showed that several global explanation methods can be viewed as *additive importance measures*, including permutation tests, Shapley Net Effects, feature ablation and SAGE (Breiman, 2001; Lipovetsky and Conklin, 2001; Lei et al., 2018).

Relative to prior work, the unification we propose is considerably broader but nonetheless precise. By focusing on the common mechanism of removing features from a model, we encompass far more methods, including both local and global ones. We also provide richer theoretical analysis by exploring underlying connections with cooperative game theory, information theory and cognitive psychology.

As we describe below, our framework characterizes methods along three dimensions. The choice of how to remove features has been considered by many works (Lundberg and Lee,

[2]This perhaps unconventional perspective is also mentioned by Bhatt et al. (2020).

2017; Chang et al., 2018; Janzing et al., 2020; Sundararajan and Najmi, 2019; Merrick and Taly, 2019; Aas et al., 2019; Hooker and Mentch, 2019; Agarwal and Nguyen, 2020; Frye et al., 2021). However, the choice of what model behavior to analyze has been considered explicitly by only a few works (Lundberg et al., 2020; Covert et al., 2020), as has the choice of how to summarize each feature's influence based on a cooperative game (Štrumbelj et al., 2009; Datta et al., 2016; Lundberg and Lee, 2017; Frye et al., 2020; Covert et al., 2020). To our knowledge, ours is the first work to consider all three dimensions simultaneously and discuss them within a single unified framework.

Besides the methods that we focus on, there are also many proposals that do not rely on the feature removal principle. We direct readers to survey articles for a broader overview of the literature (Adadi and Berrada, 2018; Guidotti et al., 2018).

3.4 Removal-Based Explanations

We now introduce our framework and briefly describe the methods it unifies.

3.4.1 A Unified Framework

We develop a unified model explanation framework by connecting methods that define each feature's influence through the impact of removing it from a model. This principle describes a substantial portion of the explainability literature: we find that 26 existing methods rely on this mechanism, including many of the most widely used approaches (Breiman, 2001; Ribeiro et al., 2016; Fong and Vedaldi, 2017; Lundberg and Lee, 2017). Several works have described their methods as either *removing*, *ignoring* or *deleting* information from a model, but our work is the first to precisely characterize this approach and document its use throughout the model explanation field.

The methods that we identify all remove groups of features from the model, but, beyond that, they take a diverse set of approaches. For example, LIME fits a linear model to an interpretable representation of the input (Ribeiro et al., 2016), L2X selects the most informative features for a single example (Chen et al., 2018a), and Shapley Effects examines

Table 3.1: Choices made by existing removal-based explanations.

Method	Removal	Behavior	Summary
IME (2009)	Separate models	Prediction	Shapley value
IME (2010)	Marginalize (uniform)	Prediction	Shapley value
QII	Marginalize (marginals product)	Prediction	Shapley value
SHAP	Marginalize (conditional/marginal)	Prediction	Shapley value
KernelSHAP	Marginalize (marginal)	Prediction	Shapley value
TreeSHAP	Tree distribution	Prediction	Shapley value
LossSHAP	Marginalize (conditional)	Prediction loss	Shapley value
SAGE	Marginalize (conditional)	Dataset loss (label)	Shapley value
Shapley Net Effects	Separate models (linear)	Dataset loss (label)	Shapley value
SPVIM	Separate models	Dataset loss (label)	Shapley value
Shapley Effects	Marginalize (conditional)	Dataset loss (output)	Shapley value
Permutation Test	Marginalize (marginal)	Dataset loss (label)	Remove individual
Conditional Perm. Test	Marginalize (conditional)	Dataset loss (label)	Remove individual
Feature Ablation (LOCO)	Separate models	Dataset loss (label)	Remove individual
Univariate Predictors	Separate models	Dataset loss (label)	Include individual
L2X	Surrogate	Prediction loss (output)	High-value subset
REAL-X	Surrogate	Prediction loss (output)	High-value subset
INVASE	Missingness during training	Prediction mean loss	High-value subset
LIME (Images)	Default values	Prediction	Additive model
LIME (Tabular)	Marginalize (replacement dist.)	Prediction	Additive model
PredDiff	Marginalize (conditional)	Prediction	Remove individual
Occlusion	Zeros	Prediction	Remove individual
CXPlain	Zeros	Prediction loss	Remove individual
RISE	Zeros	Prediction	Mean when included
MM	Default values	Prediction	Partitioned subsets
MIR	Extend pixel values	Prediction	High-value subset
MP	Blurring	Prediction	Low-value subset
EP	Blurring	Prediction	High-value subset
FIDO-CA	Generative model	Prediction	High-value subset

how much of the model's variance is explained by each feature (Owen, 2014). Perhaps surprisingly, the differences between these methods are easy to systematize because they are all based on removing discrete sets of features.

As our main contribution, we introduce a framework that shows how these methods can be specified using only three choices.

Definition 3.1. ***Removal-based explanations*** *are model explanations that quantify the impact of removing groups of features from the model. These methods are determined by three choices:*

1. *(Feature removal) How the method removes features from the model (e.g. by setting*

them to default values, or by marginalizing over a distribution of values)

2. *(Model behavior) What model behavior the method analyzes (e.g., the probability of the true class, or the model loss)*

3. *(Summary technique) How the method summarizes each feature's impact on the model (e.g., by removing a feature individually, or by calculating Shapley values)*

These three dimensions are independent of one another (i.e., any combination of choices is possible), but all three are necessary to fully specify a removal-based explanation. The first two choices, feature removal and model behavior, allow us to probe how a model's predictions change when given access to arbitrary subsets of features—including the behavior with all features, or with one or more features removed. Then, because there are an exponential number of feature combinations to consider, a summary technique is required to condense this information into a human-interpretable explanation, typically using either attribution scores or a subset of highly influential features.

As we show in the following sections, each dimension of the framework is represented by a specific mathematical choice. This precise yet flexible framework allows us to unify disparate classes of explanation methods, and, by unraveling each method's choices, offers a step towards a better understanding of the literature.

3.4.2 Overview of Existing Approaches

We now outline our findings, which we present in more detail in the remainder of the chapter. In particular, we preview how existing methods fit into our framework and highlight groups of methods that appear closely related in light of our feature removal perspective.

Table 3.1 lists the methods unified by our framework, with acronyms and the original works introduced in Section 3.5. These methods represent diverse parts of the interpretability literature, including global interpretability methods (Breiman, 2001; Owen, 2014; Covert et al., 2020), computer vision-focused methods (Zeiler and Fergus, 2014; Zhou et al.,

2014; Fong and Vedaldi, 2017; Petsiuk et al., 2018), game-theoretic methods (Štrumbelj and Kononenko, 2010; Datta et al., 2016; Lundberg and Lee, 2017) and feature selection methods (Chen et al., 2018a; Yoon et al., 2018; Fong et al., 2019). They are all unified by their reliance on feature removal, and they can be described concisely via their three choices within our framework.

Disentangling the details of each method shows that many approaches share one or more of the same choices. For example, most methods choose to explain individual predictions (the model behavior), and the Shapley value (Shapley, 1953) is the most popular summary technique. These common choices reveal that methods sometimes differ along only one or two dimensions, making it easier to reason about the trade-offs among approaches that might otherwise be viewed as monolithic algorithms.

To further highlight similarities among these methods, we visually depict the space of removal-based explanations in Figure 3.2. Visualizing our framework reveals several regions in the space of methods that are crowded (e.g., methods that marginalize out features with their conditional distribution and calculate Shapley values), while certain methods are relatively unique and spatially isolated (e.g., RISE, LIME for tabular data, INVASE). Empty positions in the grid reveal opportunities to develop new methods; in our experiments, we explore these possibilities by filling out the space of removal-based explanations (Section 3.11).

Finally, Table 3.2 shows groups of methods that differ in only one dimension of the framework. These methods are "neighbors" in the space of explanation methods (Figure 3.2), and it is noteworthy how many instances of neighboring methods exist in the literature. Certain methods even have neighbors along every dimension of the framework (e.g., SHAP, SAGE, Occlusion, PredDiff, conditional permutation tests), reflecting how intimately connected the field has become. In the remainder of the chapter, after analyzing the choices made by each method, we consider perspectives from related fields that help reason about the conceptual and computational advantages that arise from the sometimes subtle differences between methods.

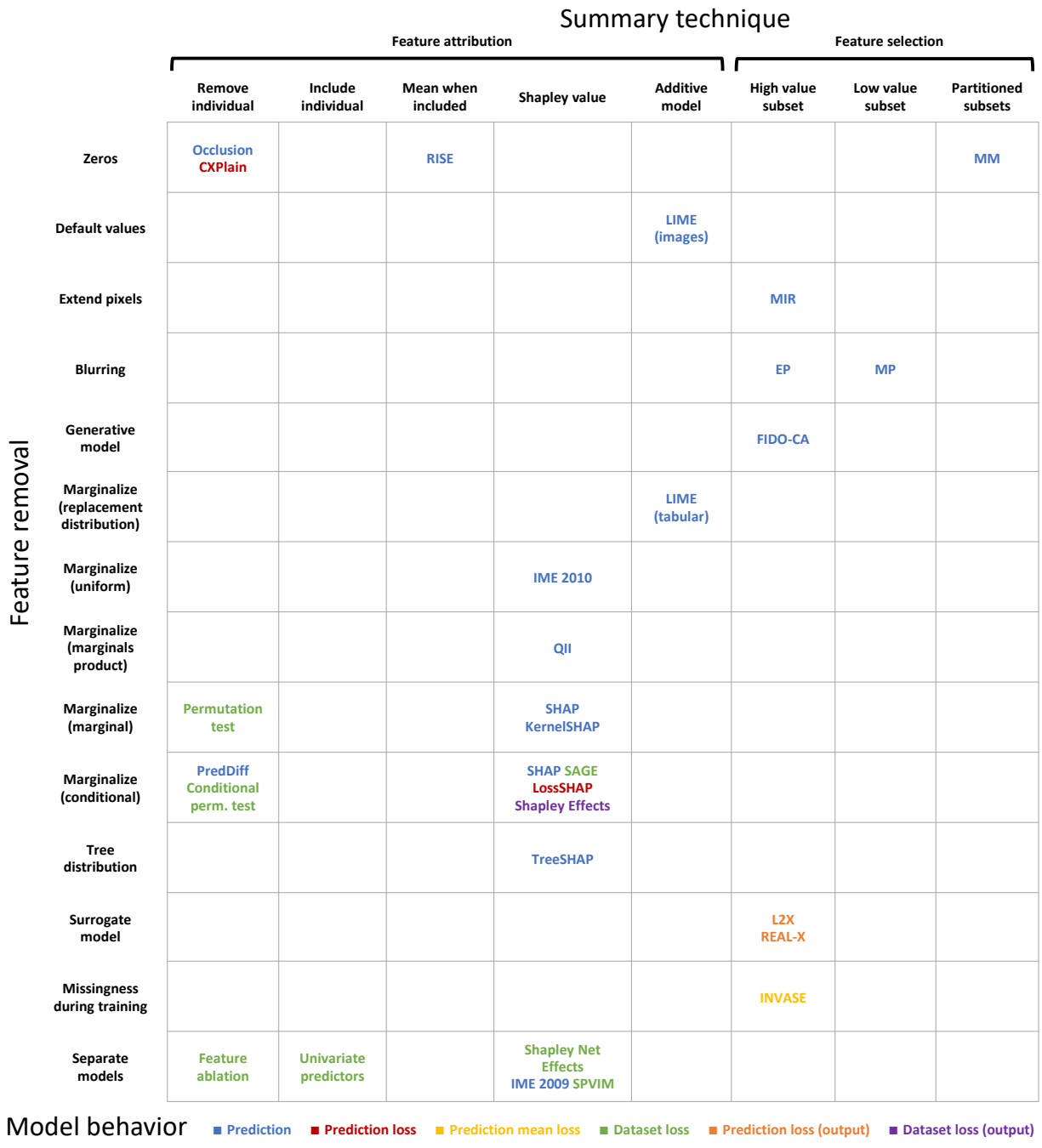


Figure 3.2: Visual depiction of the space of removal-based explanations.

Table 3.2: Common combinations of choices in existing methods. Check marks (✓) indicate choices that are identical between methods. Note that for methods that share all three choices, there can still be differences due to other implementation choices (Appendix A.1).

Removal	Behavior	Summary	Methods
	✓	✓	IME, QII, SHAP, KernelSHAP, TreeSHAP
✓		✓	SHAP, LossSHAP, SAGE, Shapley Effects
✓	✓		Occlusion, LIME (images), MM, RISE
	✓	✓	Feature ablation (LOCO), permutation tests, conditional permutation tests
✓	✓		Univariate predictors, feature ablation (LOCO), Shapley Net Effects, SPVIM
	✓	✓	SAGE, Shapley Net Effects, SPVIM
✓	✓		SAGE, conditional permutation tests
✓		✓	Shapley Net Effects, SPVIM, IME (2009)
✓		✓	Occlusion, CXPlain
	✓	✓	Occlusion, PredDiff
✓		✓	Conditional permutation tests, PredDiff
✓	✓		SHAP, PredDiff
✓	✓		MP, EP
	✓	✓	EP, FIDO-CA
✓	✓	✓	L2X, REAL-X
✓	✓	✓	Shapley Net Effects, SPVIM

3.5 Feature Removal

Here, we begin presenting our framework for removal-based explanations in detail. We first define the mathematical tools necessary to remove features from ML models, and we then examine how existing explanation methods remove features.

3.5.1 Functions on Feature Subsets

The principle behind removal-based explanations is to remove groups of features to understand their impact on a model, but since most models require all the features to make predictions, removing a feature is more complicated than simply not giving the model access to it. Most ML models make predictions given a specific set of features $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$, and mathematically, these models are functions are of the form $f : \mathcal{X} \mapsto \mathcal{Y}$, where we use $\mathcal{F}$ to denote the set of all such possible mappings.

To remove features from a model, or to make predictions given a subset of features, we require a different mathematical object than $f \in \mathcal{F}$. Instead of functions with domain $\mathcal{X}$, we consider functions with domain $\mathcal{X} \times \mathcal{P}(d)$, where $\mathcal{P}(d)$ denotes the power set of $[d]$. To ensure invariance to the held-out features, these functions must depend only on features specified by the subset $S \in \mathcal{P}(d)$, so we formalize *subset functions* as follows.[3]

Definition 3.2. *A **subset function** is a mapping of the form*

$$F : \mathcal{X} \times \mathcal{P}(d) \mapsto \mathcal{Y}$$

that is invariant to the dimensions that are not in the specified subset. That is, we have $F(x, S) = F(x', S)$ for all (x, x', S) such that $x_S = x'_S$. We define $F(x_S) \equiv F(x, S)$ for convenience because the held-out values $x_{\bar{S}}$ are not used by F.

A subset function's invariance property is crucial to ensure that only the specified feature values determine the function's output, while guaranteeing that the other feature values do

[3]In other chapters of this thesis, the notation $f(\boldsymbol{x}_S)$ is used informally to refer to such subset functions.

not matter. Another way of viewing subset functions is that they provide an approach to accommodate missing data. While we use $\mathcal{F}$ to represent standard prediction functions, we use $\mathfrak{F}$ to denote the set of all possible subset functions.

We introduce subset functions here because they help conceptualize how different methods remove features from ML models. Removal-based explanations typically begin with an existing model $f \in \mathcal{F}$, and in order to quantify each feature's influence, they must establish a convention for removing it from the model. A natural approach is to define a subset function $F \in \mathfrak{F}$ based on the original model f. To formalize this idea, we define a model's *subset extension* as follows.

Definition 3.3. *An **subset extension** of a model $f \in \mathcal{F}$ is a subset function $F \in \mathfrak{F}$ that agrees with f in the presence of all features. That is, the model f and its subset extension F must satisfy*

$$F(x) = f(x) \quad \forall x \in \mathcal{X}.$$

As we show next, specifying a subset function $F \in \mathfrak{F}$, often as a subset extension of an existing model $f \in \mathcal{F}$, is the first step towards defining a removal-based explanation.

3.5.2 Removing Features From Machine Learning Models

Existing methods have devised numerous ways to evaluate models while withholding groups of features. Although certain methods use different terminology to describe their approaches (e.g., deleting information, ignoring features, using neutral values, etc.), the goal of all these methods is to measure a feature's influence through the impact of removing it from the model. Most proposed techniques can be understood as subset extensions $F \in \mathfrak{F}$ of an existing model $f \in \mathcal{F}$ (Definition 3.3).

The various approaches used in existing work (see Appendix A.1 for more details) include:

- (Zeros) Occlusion (Zeiler and Fergus, 2014), RISE (Petsiuk et al., 2018) and CXPlain

(Schwab and Karlen, 2019) remove features simply by setting them to zero:

$$F(x_S) = f(x_S, 0). \tag{3.1}$$

- (Default values) LIME for image data (Ribeiro et al., 2016) and the Masking Model method (MM, Dabkowski and Gal, 2017) remove features by setting them to user-defined default values (e.g., gray pixels for images). Given default values $b \in \mathcal{X}$, these methods calculate

  $$F(x_S) = f(x_S, b_{\bar{S}}). \tag{3.2}$$

  Sometimes referred to as *baseline values*, (Sundararajan and Najmi, 2019), this is a generalization of the previous approach, and in some cases features may be given different default values (e.g., their mean).

- (Extend pixel values) Minimal image representation (MIR, Zhou et al., 2014) removes features from images by extending the values of neighboring pixels. This effect is achieved through a gradient-space manipulation.

- (Blurring) Meaningful Perturbations (MP, Fong and Vedaldi, 2017) and Extremal Perturbations (EP, Fong et al., 2019) remove features from images by blurring them with a Gaussian kernel. This approach is *not* a subset extension of f because the blurred image retains dependence on the removed features. Blurring fails to remove large, low frequency objects (e.g., mountains), but it provides an approximate way to remove information from images.

- (Generative model) FIDO-CA (Chang et al., 2018) removes features by replacing them with samples from a conditional generative model (e.g. Yu et al., 2018). The held-out features are drawn from a generative model represented by $p_G(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S)$, or $\tilde{x}_{\bar{S}} \sim$

$p_G(\boldsymbol{x}_{\bar{S}} \mid x_S)$, and predictions are made as follows:

$$F(x_S) = f(x_S, \tilde{x}_{\bar{S}}). \tag{3.3}$$

- (Marginalize with conditional) SHAP (Lundberg and Lee, 2017), LossSHAP (Lundberg et al., 2020) and SAGE (Covert et al., 2020) present a strategy for removing features by marginalizing them out using their conditional distribution $p(\boldsymbol{x}_{\bar{S}} \mid x_S)$:

$$F(x_S) = \mathbb{E}[f(\boldsymbol{x}) \mid x_S]. \tag{3.4}$$

  This approach is computationally challenging in practice, but recent work tries to achieve close approximations (Aas et al., 2019, 2021; Frye et al., 2021). Shapley Effects (Owen, 2014) implicitly uses this convention to analyze function sensitivity, while conditional permutation tests (Strobl et al., 2008) and Prediction Difference Analysis (PredDiff, Zintgraf et al., 2017) propose simple approximations, with the latter conditioning only on groups of bordering pixels.

- (Marginalize with marginal) KernelSHAP (a practical implementation of SHAP) removes features by marginalizing them out using their joint marginal distribution $p(\boldsymbol{x}_{\bar{S}})$:

$$F(x_S) = \mathbb{E}[f(x_S, \boldsymbol{x}_{\bar{S}})]. \tag{3.5}$$

  This is the default behavior in SHAP's implementation,[4] and recent work discusses its potential benefits over conditional marginalization (Janzing et al., 2020). Permutation tests (Breiman, 2001) use this approach to remove individual features.

- (Marginalize with product of marginals) Quantitative Input Influence (QII, Datta et al., 2016) removes held-out features by marginalizing them out with the product of the

[4] `https://github.com/slundberg/shap`

marginal distributions $p(\boldsymbol{x}_i)$:

$$F(x_S) = \mathop{\mathbb{E}}_{\prod_{i \in \bar{S}} p(\boldsymbol{x}_i)}[f(x_S, \boldsymbol{x}_{\bar{S}})]. \tag{3.6}$$

- (Marginalize with uniform) The updated version of the Interactions Method for Explanation (IME, Štrumbelj and Kononenko, 2010) removes features by marginalizing them out with a uniform distribution over the feature space. If we let $u_i(\boldsymbol{x}_i)$ denote a uniform distribution over $\mathcal{X}_i$ (with extremal values defining the boundaries for continuous features) and $u(\boldsymbol{x}) = \prod_{i \in [d]} u(\boldsymbol{x}_i)$, then features are removed as follows:

$$F(x_S) = \mathop{\mathbb{E}}_{u(\boldsymbol{x})}[f(x_S, \boldsymbol{x}_{\bar{S}})]. \tag{3.7}$$

- (Marginalize with replacement distributions) LIME for tabular data replaces features with independent draws from *replacement distributions* (our term), each of which depends on the original feature values. When a feature $\boldsymbol{x}_i$ with value x_i is removed, discrete features are drawn from the distribution $p(\boldsymbol{x}_i \mid \boldsymbol{x}_i \neq x_i)$; when quantization is used for continuous features (LIME's default behavior[5]), continuous features are sampled by first generating a different quantile and then sampling from a truncated normal distribution within that bin. If we denote each feature's replacement distribution given the original value x_i as $q_{x_i}(\boldsymbol{x}_i)$ and then denote $q_x(\boldsymbol{x}) = \prod_{i \in [d]} q_{x_i}(\boldsymbol{x}_i)$, then LIME for tabular data removes features as follows:

$$F(x, S) = \mathop{\mathbb{E}}_{q_x(\boldsymbol{x})}[f(x_S, \boldsymbol{x}_{\bar{S}})]. \tag{3.8}$$

  Although this function F agrees with f given all features, it is *not* a subset extension because it does not satisfy the invariance property necessary for subset functions.

[5] `https://github.com/marcotcr/lime`

- (Tree distribution) Dependent TreeSHAP (Lundberg et al., 2020) removes features using the distribution induced by the underlying tree model, which roughly approximates the conditional distribution. When splits for removed features are encountered in the model's trees, TreeSHAP averages predictions from the multiple paths in proportion to how often the dataset follows each path.

- (Surrogate model) Learning to Explain (L2X, Chen et al., 2018a) and REAL-X (Jethani et al., 2021a) train separate surrogate models F to match the original model's predictions when groups of features are held out. The surrogate model accommodates missing features, allowing us to represent it as a subset function $F \in \mathfrak{F}$, and it aims to provide the following approximation:

$$F(x_S) \approx \mathbb{E}[f(\boldsymbol{x}) \mid x_S]. \tag{3.9}$$

  The surrogate model approach was also proposed separately in the context of Shapley values (Frye et al., 2021).

- (Missingness during training) Instance-wise Variable Selection (INVASE, Yoon et al., 2018) uses a model that has missingness introduced at training time. Removed features are replaced with zeros, so that the model makes the following approximation:

$$F(x_S) = f(x_S, 0) \approx p(\boldsymbol{y} \mid x_S). \tag{3.10}$$

  This approximation occurs for models trained with cross entropy loss, but other loss functions may lead to different results (e.g., the conditional expectation for MSE loss). Introducing missingness during training differs from the default values approach because the model is trained to recognize zeros (or other replacement values) as missing values rather than zero-valued features.

- (Separate models) The original version of IME (Štrumbelj et al., 2009) is not based on a single model f, but rather on separate models trained for each feature subset, or $\{f_S : S \subseteq [d]\}$. The prediction for a subset of features is given by that subset's model:

$$F(x_S) = f_S(x_S). \tag{3.11}$$

  Shapley Net Effects (Lipovetsky and Conklin, 2001) uses an identical approach in the context of linear models, with SPVIM generalizing the approach to black-box models (Williamson and Feng, 2020). Similarly, feature ablation, also known as leave-one-covariate-out (LOCO, Lei et al., 2018), trains models to remove individual features, and the univariate predictors approach (used mainly for feature selection) uses models trained with individual features (Guyon and Elisseeff, 2003). Although the separate models approach is technically a subset extension of the model $f_{[d]}$ trained with all features, its predictions given subsets of features are not technically tied to $f_{[d]}$.

Most of these approaches can be viewed as subset extensions of an existing model f, so our formalisms provide useful tools for understanding how removal-based explanations remove features from models. However, there are two exceptions: the blurring technique (MP and EP) and LIME's approach with tabular data. Both provide functions of the form $F : \mathcal{X} \times \mathcal{P}(d) \mapsto \mathcal{Y}$ that agree with f given all features, but that retain dependence on removed features. Based on our invariance property for held-out features (Definition 3.2), we argue that these approaches do not fully remove features from the model.

We conclude that the first dimension of our framework amounts to choosing a subset function $F \in \mathfrak{F}$, often via a subset extension to an existing model $f \in \mathcal{F}$. We defer consideration of the trade-offs between these approaches to Section 3.9, where we show that one approach to removing features yields connections to information theory.

3.6 Explaining Different Model Behaviors

Removal-based explanations all aim to demonstrate how a model functions, but they can do so by analyzing different model behaviors. We now consider the various choices of target quantities to observe as different features are withheld from the model.

The feature removal principle is flexible enough to explain virtually any function. For example, methods can explain a model's prediction, a model's loss function, a hidden layer in a neural network, or any node in a computation graph. In fact, removal-based explanations need not be restricted to the ML context: any function that accommodates missing inputs can be explained via feature removal by observing either its output or some function of its output as groups of inputs are removed. This perspective shows the broad potential applications for removal-based explanations.

Because our focus is the ML context, we proceed by examining how existing methods work. Each explanation method's target quantity can be understood as a function of the model output, which for simplicity is represented by a subset function $F(x_S)$. Many methods explain the model output or a simple function of the output, such as the logits or log-odds ratio. Other methods take into account a measure of the model's loss, for either an individual input or the entire dataset. Ultimately, as we show below, each method generates explanations based on a set function of the form

$$u : \mathcal{P}(d) \mapsto \mathbb{R}, \tag{3.12}$$

which represents the value associated with each subset of features $S \subseteq [d]$. This set function corresponds to the model behavior that a method is designed to explain.

We now examine the specific choices made by existing methods (see Appendix A.1 for further details). The various model behaviors that methods analyze, and their corresponding set functions, include:

- (Prediction) Occlusion, RISE, PredDiff, MP, EP, MM, FIDO-CA, MIR, LIME, SHAP

(including KernelSHAP and TreeSHAP), IME and QII all analyze a model's prediction for an individual input $x \in \mathcal{X}$:

$$u_x(S) = F(x_S). \tag{3.13}$$

These methods examine how holding out different features makes an individual prediction either higher or lower. For multi-class classification models, methods often use a single output that corresponds to the class of interest, and they can optionally apply a simple function to the model's output (for example, using the log-odds ratio rather than the classification probability).

- (Prediction loss) LossSHAP and CXPlain take into account the true label y for an input x and calculate the prediction loss using a loss function ℓ:

$$v_{xy}(S) = -\ell(F(x_S), y). \tag{3.14}$$

By incorporating the label, these methods quantify whether certain features make the prediction more or less correct. Note that the minus sign is necessary to give the set function a higher value when more informative features are included.

- (Prediction mean loss) INVASE considers the expected loss for an input x according to the label's conditional distribution $p(\boldsymbol{y} \mid x)$:

$$v_x(S) = -\mathop{\mathbb{E}}_{p(\boldsymbol{y}|x)}\big[\ell(F(x_S), \boldsymbol{y})\big]. \tag{3.15}$$

By averaging the loss across the label's distribution, INVASE highlights features that correctly predict what *could* have occurred, on average.

- (Dataset loss) Shapley Net Effects, SAGE, SPVIM, feature ablation, permutation tests

and univariate predictors consider the expected loss across the entire dataset:

$$v(S) = - \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \big[\ell(F(\boldsymbol{x}_S), \boldsymbol{y})\big]. \tag{3.16}$$

These methods quantify how much the model's performance degrades when different features are removed. This set function can also be viewed as the predictive power derived from sets of features (Covert et al., 2020), and recent work has proposed a SHAP value aggregation that is a special case of this approach (Frye et al., 2021).

- (Prediction loss w.r.t. output) L2X and REAL-X consider the loss between the full model output and the prediction given a subset of features:

$$w_x(S) = -\ell(F(x_S), F(x)). \tag{3.17}$$

These methods highlight features that on their own lead to similar predictions as the full feature set.

- (Dataset loss w.r.t. output) Shapley Effects considers the expected loss with respect to the full model output:

$$w(S) = - \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \big[\ell(F(\boldsymbol{x}_S), F(\boldsymbol{x}))\big]. \tag{3.18}$$

Though related to the dataset loss approach (Covert et al., 2020), this approach focuses on each feature's influence on the model output rather than on the model performance.

Each set function serves a distinct purpose in exposing a model's dependence on different features. Several of the approaches listed above analyze the model's behavior for individual predictions (local explanations), while some take into account the model's behavior across the entire dataset (global explanations). Although their aims differ, these set functions are all related. Each builds upon the previous ones by accounting for either the loss or data

distribution, and their relationships can be summarized as follows:

$$v_{xy}(S) = -\ell(u_x(S), y) \tag{3.19}$$

$$w_x(S) = -\ell(u_x(S), u_x([d])) \tag{3.20}$$

$$v_x(S) = \mathop{\mathbb{E}}_{p(\boldsymbol{y}|x)}[v_{x\boldsymbol{y}}(S)] \tag{3.21}$$

$$v(S) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}[v_{\boldsymbol{xy}}(S)] \tag{3.22}$$

$$w(S) = \mathop{\mathbb{E}}_{p(\boldsymbol{x})}[w_{\boldsymbol{x}}(S)] \tag{3.23}$$

These relationships show that explanations based on one set function are in some cases related to explanations based on another. For example, Covert et al. (2020) showed that SAGE explanations are the expectation of explanations provided by LossSHAP—a relationship reflected in eq. 3.22.

Understanding these connections is made easier by the fact that our framework disentangles each method's choices rather than viewing each method as a monolithic algorithm. We conclude by reiterating that removal-based explanations can explain virtually any function, and that choosing what model behavior to explain amounts to selecting a set function $u : \mathcal{P}(d) \mapsto \mathbb{R}$ to represent the model's dependence on different sets of features.

3.7 Summarizing Feature Influence

The third choice for removal-based explanations is how to summarize each feature's influence on the model. We examine the various summarization techniques and then discuss their computational complexity and approximation approaches.

3.7.1 Explaining Set Functions

The set functions that represent a model's dependence on different features (Section 3.6) are complicated mathematical objects that are difficult to communicate to users: the model's behavior can be observed for any subset of features, but there are an exponential number

of subsets to consider. Removal-based explanations handle this challenge by providing users with a concise summary of each feature's influence.

We distinguish between two main types of summarization approaches: *feature attribution* and *feature selection*. Many methods provide explanations in the form of feature attributions, which are numerical scores $a_i \in \mathbb{R}$ given to each feature $i \in [d]$. If we use $\mathcal{U}$ to denote the set of all functions $u : \mathcal{P}(d) \mapsto \mathbb{R}$, then we can represent feature attributions as mappings of the form $E : \mathcal{U} \mapsto \mathbb{R}^d$, which we refer to as *explanation mappings*. Other methods take the alternative approach of summarizing set functions with a set $S^* \subseteq [d]$ of the most influential features. We represent these feature selection summaries as explanation mappings of the form $E : \mathcal{U} \mapsto \mathcal{P}(d)$. Both approaches provide users with simple summaries of a feature's contribution to the set function.

We now consider the specific choices made by each method (see Appendix A.1 for further details). For simplicity, we let u denote the set function each method analyzes. Surveying the various removal-based explanation methods, the techniques for summarizing each feature's influence include:

- (Remove individual) Occlusion, PredDiff, CXPlain, permutation tests and feature ablation (LOCO) calculate the impact of removing a single feature from the model, resulting in the following attribution values:

$$a_i = u([d]) - u([d] \setminus i). \tag{3.24}$$

  Occlusion, PredDiff and CXPlain can also be applied with groups of features, or superpixels, in image contexts.

- (Include individual) The univariate predictors approach calculates the impact of including individual features, resulting in the following attribution values:

$$a_i = u(\{i\}) - u(\varnothing). \tag{3.25}$$

This is essentially the reverse of the previous approach: rather than removing individual features from the complete set, this approach adds individual features to the empty set.

- (Additive model) LIME fits a regularized additive model to a dataset of perturbed examples. In the limit of sampling all subsets, this process approximates the following attribution values:

$$a_1, \ldots, a_d = \underset{\beta_0,\ldots,\beta_d}{\arg\min} \sum_{S \subseteq [d]} \pi(S)\Big(\beta_0 + \sum_{i \in S} \beta_i - u(S)\Big)^2 + \Omega(\beta_1, \ldots, \beta_d). \tag{3.26}$$

  In this problem, π represents a weighting kernel and Ω is a regularization function that is often set to the ℓ_1 penalty to encourage sparse attributions (Tibshirani, 1996). Since this summary is based on an additive model, the learned coefficients $(a_1, \ldots, a_d)$ represent the incremental value associated with including each feature.

- (Mean when included) RISE determines feature attributions by sampling many subsets $S \subseteq [d]$ and then calculating the mean value when a feature is included. Denoting the distribution of subsets as $p(\boldsymbol{S})$ and the conditional distribution as $p(\boldsymbol{S} \mid i \in \boldsymbol{S})$, the attribution values are defined as

$$a_i = \mathop{\mathbb{E}}_{p(\boldsymbol{S} \mid i \in \boldsymbol{S})} \big[u(\boldsymbol{S})\big]. \tag{3.27}$$

  In practice, RISE samples the subsets $S \subseteq [d]$ by removing each feature i independently with probability p, using $p = 0.5$ in their experiments (Petsiuk et al., 2018).

- (Shapley value) Shapley Net Effects, IME, Shapley Effects, QII, SHAP (including KernelSHAP, TreeSHAP and LossSHAP), SPVIM and SAGE all calculate feature attributions using the Shapley value, which we denote as $a_i = \phi_i(u)$. Described in more detail in Section 3.8, Shapley values are the only attributions that satisfy several desirable

properties.

- (Low-value subset) MP selects a small set of features S^* that can be removed to give the set function a low value. It does so by solving the following optimization problem:

$$S^* = \arg\min_S \; u([d] \setminus S) + \lambda |S|. \tag{3.28}$$

  In practice, MP incorporates additional regularizers and solves a relaxed version of this problem (see details below).

- (High-value subset) MIR solves an optimization problem to select a small set of features S^* that alone can give the set function a high value. For a user-defined minimum value t, the problem is given by:

$$S^* = \arg\min_S \; |S| \quad \text{s.t.} \;\; u(S) \geq t. \tag{3.29}$$

  L2X and EP solve a similar problem but switch the terms in the constraint and optimization objective. For a user-defined subset size k, the optimization problem is given by:

$$S^* = \arg\max_S \; u(S) \quad \text{s.t.} \;\; |S| = k. \tag{3.30}$$

  Finally, INVASE, REAL-X and FIDO-CA solve a regularized version of the problem with a parameter $\lambda > 0$ controlling the trade-off between the subset value and subset size:

$$S^* = \arg\max_S \; u(S) - \lambda |S|. \tag{3.31}$$

- (Partitioned subsets) MM solves an optimization problem to partition the features into S^* and $[d] \setminus S^*$ while maximizing the difference in the set function's values. This approach is based on the idea that removing features to find a low-value subset (as in MP) and retaining features to get a high-value subset (as in MIR, L2X, EP, INVASE,

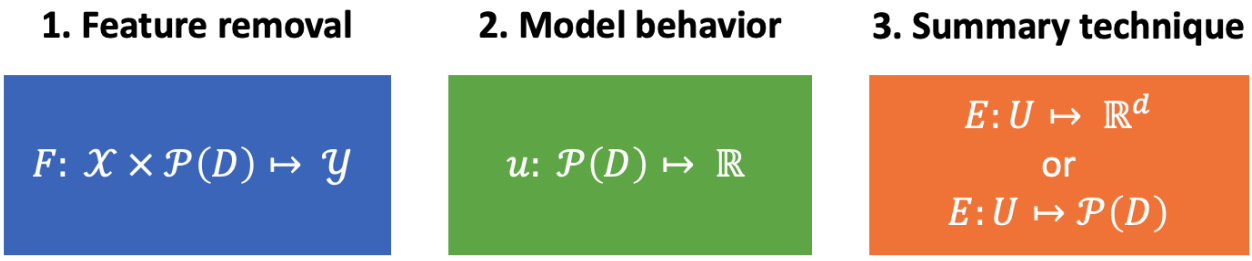


Figure 3.3: Removal-based explanations are specified by three precise mathematical choices: a subset function $F \in \mathfrak{F}$, a set function $u \in \mathcal{U}$, and an explanation mapping E (for feature attribution or selection).

REAL-X and FIDO-CA) are both reasonable approaches for identifying influential features. Given two parameters $\lambda > 0$ and $\gamma > 0$, the problem is defined as:

$$S^* = \arg\max_S \; u(S) - \gamma u([d] \setminus S) - \lambda |S|. \tag{3.32}$$

In practice, MM also incorporates regularizers and monotonic link functions to enable a more flexible trade-off between $u(S)$ and $u([d] \setminus S)$ (see Appendix A.1).

As this discussion shows, every removal-based explanation generates a summary of each feature's influence on the underlying set function. In general, a model's dependencies are too complex to communicate fully, so explanations must provide users with a concise summary instead. Feature attributions provide a granular view of each feature's influence on the model, while feature selection summaries can be understood as coarse attributions that assign binary rather than real-valued importance.

Interestingly, if the high-value subset optimization problems solved by MIR, L2X, EP, INVASE, REAL-X and FIDO-CA were applied to the set function that represents the dataset loss (eq. 3.22), they would resemble conventional global feature selection methods (Guyon and Elisseeff, 2003). The problem in eq. 3.30 determines the set of k features with maximum predictive power, the problem in eq. 3.29 determines the smallest set of features that achieve

the performance given by t, and the problem in eq. 3.31 uses a parameter λ to control the trade-off. Though not generally viewed as a model explanation approach, global feature selection serves a similar purpose of identifying highly predictive features.

We conclude by reiterating that the third dimension of our framework amounts to a choice of explanation mapping, which takes the form $E : \mathcal{U} \mapsto \mathbb{R}^d$ for feature attribution or $E : \mathcal{U} \mapsto \mathcal{P}(d)$ for feature selection. Our discussion so far has shown that removal-based explanations can be specified using three precise mathematical choices, as depicted in Figure 3.3. These methods, which are often presented in ways that make their connections difficult to discern, are constructed in a remarkably similar fashion. The remainder of this chapter addresses the relationships and trade-offs among these different choices, beginning by examining the computational complexity of each summarization approach.

3.7.2 Complexity and Approximations

Showing how certain explanation methods fit into our framework requires distinguishing between their implicit objectives and the approximations that make them practical. Our presentation of these methods deviates in some instances from the original papers, which often focus on details of a method's implementation. We now bridge the gap by describing these methods' computational complexity and the approximations that are sometimes used out of necessity.

The challenge with most of the summarization techniques described above is that they require calculating the underlying set function's value $u(S)$ for many subsets of features. In fact, without making any simplifying assumptions about the model or data distribution, several techniques must examine all 2^d subsets of features. This includes the Shapley value, RISE's summarization technique and LIME's additive model. Finding exact solutions to several of the optimization problems (MP, MIR, MM, INVASE, REAL-X, FIDO-CA) also requires examining all subsets of features, and solving the constrained optimization problem (EP, L2X) for k features requires examining $\binom{d}{k}$ subsets, or $\mathcal{O}(2^d d^{-\frac{1}{2}})$ subsets in the worst

case.[6]

The only approaches with lower computational complexity are those that remove individual features (Occlusion, PredDiff, CXPlain, permutation tests, feature ablation) or include individual features (univariate predictors). These require only one subset per feature, or d feature subsets in total.

Many summarization techniques have superpolynomial complexity in d, making them intractable for large numbers of features, at least with naive implementations. These methods work in practice due to fast approximation approaches, and in some cases techniques have even been devised to generate real-time explanations. Several strategies that yield fast approximations include:

- Attribution values that are the expectation of a random variable can be estimated by Monte Carlo approximation. IME (Štrumbelj and Kononenko, 2010), Shapley Effects (Song et al., 2016) and SAGE (Covert et al., 2020) use sampling strategies to approximate Shapley values, and RISE also estimates its attributions via sampling (Petsiuk et al., 2018).

- KernelSHAP, LIME and SPVIM are based on linear regression models fitted to datasets containing an exponential number of datapoints. In practice, these techniques fit models to smaller sampled datasets, which means optimizing an approximate version of their objective function (Lundberg and Lee, 2017; Covert and Lee, 2021).[7]

- TreeSHAP calculates Shapley values in polynomial time using a dynamic programming algorithm that exploits the structure of tree-based models. Similarly, L-Shapley and C-Shapley exploit the properties of models for structured data to provide fast Shapley value approximations (Chen et al., 2018b).

---

[6]This can be seen by applying Stirling's approximation to $\binom{d}{d/2}$ as d becomes large.

[7]See Chapter 5 for a detailed discussion of KernelSHAP.

- Several of the feature selection methods (MP, L2X, REAL-X, EP, MM, FIDO-CA) solve continuous relaxations of their discrete optimization problems. While these optimization problems can be solved by representing the set of features $S \subseteq [d]$ as a mask $m \in \{0,1\}^d$, these methods instead use a continuous mask variable of the form $m \in [0,1]^d$. When these methods incorporate a penalty on the subset size $|S|$, they also sometimes use the convex relaxation $||m||_1$.

- One feature selection method (MIR) uses a greedy optimization algorithm. MIR determines a set of influential features $S \subseteq [d]$ by iteratively removing groups of features that do not reduce the predicted probability for the correct class.

- One feature attribution method (CXPlain) and several feature selection methods (L2X, INVASE, REAL-X, MM) generate real-time explanations by learning separate explainer models. CXPlain learns an explainer model using a dataset consisting of manually calculated explanations, which removes the need to iterate over each feature when generating new explanations. L2X learns a model that outputs a set of features (represented by a k-hot vector) and INVASE/REAL-X learn similar selector models that can output arbitrary numbers of features. MM learns a model that outputs masks of the form $m \in [0,1]^d$ for images. These techniques can be viewed as *amortized* approaches because they learn models that perform the summarization step in a single forward pass.

In conclusion, many methods have developed approximations that enable efficient model explanation, despite sometimes using summarization techniques that are inherently intractable (e.g., Shapley values). Certain techniques are considerably faster than others (i.e., the amortized approaches), and some can trade off computational cost for approximation accuracy (Štrumbelj and Kononenko, 2010; Covert and Lee, 2021), but they are all sufficiently fast to be used in practice.

We speculate, however, that more approaches will be made to run in real-time by learning

separate explainer models, as in the MM, L2X, INVASE, CXPlain and REAL-X approaches (Dabkowski and Gal, 2017; Chen et al., 2018a; Yoon et al., 2018; Schwab and Karlen, 2019; Jethani et al., 2021a). Besides these methods, others have been proposed that learn the explanation process either as a component of the original model (Fan et al., 2017; Taghanaki et al., 2019) or as a separate model after training (Schulz et al., 2020). Such approaches may be necessary to bypass the need for multiple model evaluations and make removal-based explanations as fast as gradient-based and propagation-based methods.[8]

3.8 Game-Theoretic Explanations

The set functions analyzed by removal-based explanations (Section 3.6) can be viewed as *cooperative games*—mathematical objects studied in the field of cooperative game theory. Only a few explanation methods explicitly consider game-theoretic connections, but we show that every method described thus far can be understood through the lens of cooperative game theory.

3.8.1 Cooperative Game Theory Background

As described in Chapter 2, cooperative games are functions of the form $u : \mathcal{P}(d) \mapsto \mathbb{R}$ (i.e., set functions) that describe the value achieved when sets of players $S \subseteq [d]$ participate in a game. Intuitively, a game might represent the profit made when a particular group of employees chooses to work together. Cooperative game theory research focuses on understanding the properties of different payoffs that can be offered to incentivize participation in the game, as well as predicting which groups of players will ultimately agree to participate.

For this discussion, we use u to denote a cooperative game. To introduce terminology from cooperative game theory, the features $i \in [d]$ are referred to as *players*, sets of players $S \subseteq [d]$ are referred to as *coalitions*, and the output $u(S)$ is referred to as the *value* of S. Player i's *marginal contribution* to the coalition $S \in [d] \setminus i$ is defined as the difference in

[8]Chapter 6 and Chapter 7 present the first methods to amortize Shapley value computation, which were published after this work.

value $u(S \cup i) - u(S)$. *Allocations* are vectors $z \in \mathbb{R}^d$ that represent payoffs proposed to each player in return for participating in the game.

Several fundamental concepts in cooperative game theory are related to the properties of allocations: the *core* of a game, a game's *nucleolus*, and its *bargaining sets* are all based on whether players view certain allocations as favorable (Narahari, 2014). Perhaps surprisingly, every summarization technique used by removal-based explanations (Section 3.7) can be viewed in terms of allocations to players in the underlying game, enabling us to connect these explanation methods to ideas from the game theory literature.

3.8.2 Allocation Strategies

Several summarization techniques used by removal-based explanation methods are related to *solution concepts*, which in the cooperative game theory context are credit allocation strategies designed to be fair to the players. If we let $\mathcal{U}$ represent the set of all cooperative games with d players, then solution concepts are represented by mappings of the form $E : \mathcal{U} \mapsto \mathbb{R}^d$, similar to explanation mappings that represent feature attributions (Section 3.7).

We first discuss the Shapley value, which assumes that the *grand coalition* (the coalition containing all players) is participating and distributes the total value in proportion to each player's contributions (Shapley, 1953). The Shapley values $\phi_i(u) \in \mathbb{R}$ for each player $i \in [d]$ in a game u are designed to satisfy several useful theoretical properties: described in detail in Chapter 2, these properties include *efficiency*, *symmetry*, *linearity*, *null player* and *marginalism*. They are in fact the unique allocations to satisfy these properties, and the expression for each Shapley value $\phi_i(u)$ is the following:

$$\phi_i(u) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \Big(u(S \cup i) - u(S) \Big). \tag{3.33}$$

The Shapley value has found widespread use because of its axiomatic derivation, both within game theory and in other disciplines (Aumann, 1994; Shorrocks, 1999; Petrosjan and Zaccour, 2003; Tarashev et al., 2016). In the context of model explanation, Shapley values define each

feature's contribution while accounting for complex feature interactions, such as correlations, redundancy, and complementary behavior (Lipovetsky and Conklin, 2001; Štrumbelj et al., 2009; Owen, 2014; Datta et al., 2016; Lundberg and Lee, 2017; Lundberg et al., 2020; Covert et al., 2020; Williamson and Feng, 2020).

Like the Shapley value, the Banzhaf value attempts to define fair allocations for each player in a cooperative game. It generalizes the Banzhaf power index, which is a technique for measuring the impact of players in the context of voting games (Banzhaf, 1964). Links between Shapley and Banzhaf values are described by Dubey and Shapley (1979), who show that the Shapley value can be understood as an enumeration over all *permutations* of players, while the Banzhaf value can be understood as an enumeration over all *subsets* of players. The expression for each Banzhaf value $\psi_i(u)$ is:

$$\psi_i(u) = \frac{1}{2^{d-1}} \sum_{S \subseteq [d] \setminus i} \Big(u(S \cup i) - u(S) \Big). \tag{3.34}$$

The Banzhaf value fails to satisfy the Shapley value's efficiency property, but it can be derived axiomatically by introducing a variation on the efficiency axiom (Nowak, 1997):

- (2-Efficiency) If two players i, j merge into a new player $\{i, j\}$ and we re-define the game u as a game u' on the smaller set of players $\big([d] \setminus \{i, j\}\big) \cup \big\{\{i, j\}\big\}$, then the Banzhaf values satisfy the following property:

$$\psi_{\{i,j\}}(u') = \psi_i(u) + \psi_j(u). \tag{3.35}$$

The 2-efficiency property roughly states that credit allocation is immune to the merging of players. The Banzhaf value has multiple interpretations, but the most useful for our purpose is that it represents the difference between the mean value of coalitions that do and do not include the ith player when coalitions are chosen uniformly at random (see eq. 3.34). With this perspective, we observe that the RISE summarization technique is closely related to the Banzhaf value: RISE calculates the mean value of coalitions that include i, but, unlike

the Banzhaf value, it disregards the value of coalitions that do not include i. While the RISE technique (Petsiuk et al., 2018) was not motivated by the Banzhaf value, it is unsurprising that such a natural idea has been explored in cooperative game theory.

Both Shapley and Banzhaf values are mathematically appealing because they can be understood as the weighted average of a player's marginal contributions (eqs. 3.33 and 3.34). This is a stronger version of the marginalism property introduced by Young's axiomatization of the Shapley value (Young, 1985), and solution concepts of this form are known as *probabilistic values* (Weber, 1988). Probabilistic values have their own axiomatic characterization: they have been shown to be the unique values that satisfy a specific subset of the Shapley value's properties (Monderer et al., 2002).

The notion of probabilistic values reveals links with two other solution concepts. The techniques of removing individual players (e.g., Occlusion, PredDiff, CXPlain, permutation tests and feature ablation) and including individual players (e.g., univariate predictors) can also be understood as probabilistic values, although they are simple averages that put all their weight on a single marginal contribution (eqs. 3.24 and 3.25). Unlike the Shapley and Banzhaf values, these methods neglect complex interactions when quantifying each player's contribution.

The methods discussed thus far (Shapley value, Banzhaf value, removing individual players, including individual players) all satisfy the symmetry, null player, linearity, and marginalism axioms. What makes the Shapley value unique among these approaches is the efficiency axiom, which lets us view it as a distribution of the grand coalition's value among the players (Dubey and Shapley, 1979). However, whether the efficiency property or the Banzhaf value's 2-efficiency property is preferable may depend on the use case.

3.8.3 Modeling Cooperative Games

Compared to the methods discussed so far, LIME provides a more flexible approach for summarizing each player's influence. As its summarization technique, LIME fits an additive model to the cooperative game (eq. 3.26), leaving the user the option of specifying a weighting

kernel π and regularization term Ω for the weighted least squares objective.

Although fitting a model to a cooperative game seems distinct from the credit allocation strategies discussed so far, these ideas are in fact intimately connected. Fitting models to cooperative games, including both linear and nonlinear models, has been studied by numerous works in cooperative game theory (Charnes et al., 1988; Hammer and Holzman, 1992; Grabisch et al., 2000; Ding et al., 2008, 2010; Marichal and Mathonet, 2011), and specific choices for the weighting kernel can yield recognizable attribution values.

If we fit an additive model to the underlying game with no regularization ($\Omega = 0$), then we can identify several weighting kernels that correspond to other summarization techniques:

- When we use the weighting kernel $\pi_{\text{Rem}}(S) = \mathbb{1}(|S| \geq d - 1)$, where $\mathbb{1}(\cdot)$ is an indicator function, the attribution values are the marginal contributions from removing individual players from the grand coalition, or the values $a_i = u([d]) - u([d] \setminus i)$. This is the summarization technique used by Occlusion, PredDiff, CXPlain, permutation tests, and feature ablation.

- When we use the weighting kernel $\pi_{\text{Inc}}(S) = \mathbb{1}(|S| \leq 1)$, the attribution values are the marginal contributions from adding individual players to the empty coalition, or the values $a_i = u(\{i\}) - u(\varnothing)$. This is the summarization technique used by the univariate predictors approach.

- Results from Hammer and Holzman (1992) show that when we use the weighting kernel $\pi_{\text{B}}(S) = 1$, the attribution values are the Banzhaf values $a_i = \psi_i(u)$.

- Results from Charnes et al. (1988) and Lundberg and Lee (2017) show that the attribution values are equal to the Shapley values $a_i = \phi_i(u)$ when we use the weighting kernel π_{Sh}, defined as follows:

$$\pi_{\text{Sh}}(S) = \frac{d - 1}{\binom{d}{|S|}|S|(d - |S|)}. \tag{3.36}$$

Although the Shapley value connection has been noted in the model explanation context, the other results we present are new observations about LIME (proofs in Appendix A.2). These results show that the weighted least squares problem solved by LIME provides sufficient flexibility to yield both the Shapley and Banzhaf values, as well as simpler quantities likw the marginal contributions from removing or including individual players. The additive model approach captures every feature attribution technique discussed so far, with the caveat that RISE uses a modified version of the Banzhaf value. And, like the other allocation strategies, attributions arising from fitting an additive model can be shown to satisfy different sets of Shapley axioms (Appendix A.3).

We have thus demonstrated that the additive model approach for summarizing feature influence not only has precedent in cooperative game theory, but that it is intimately connected to the other allocation strategies. This suggests that the feature attributions generated by many removal-based explanations can be understood as additive decompositions of the underlying cooperative game.

3.8.4 Identifying Coalitions Using Excess

To better understand removal-based explanations that perform feature selection, we look to a different concept in cooperative game theory: the notion of *excess*. Excess is defined for a given allocation $z \in \mathbb{R}^d$ and coalition $S \subseteq [d]$ as the difference between the coalition's value and its total allocation (Narahari, 2014). More formally, it is defined as follows.

Definition 3.4. *Given a cooperative game $u : \mathcal{P}(d) \mapsto \mathbb{R}$, a coalition $S \subseteq [d]$, and an allocation $z \in \mathbb{R}^d$, the **excess** of S at z is defined as*

$$e(S, z) = u(S) - \sum_{i \in S} z_i. \tag{3.37}$$

The excess $e(S, z)$ represents the coalition's degree of unhappiness under the allocation z, because an allocation is unfavorable to a coalition if its value $u(S)$ exceeds its cumulative allocation $\sum_{i \in S} z_i$. For cooperative game theory concepts including the core, the nucleolus

and bargaining sets, a basic assumption is that coalitions with higher excess (greater unhappiness) are more likely to break off and refuse participation in the game (Narahari, 2014). We have no analogue for features refusing participation in ML, but the notion of excess is useful for understanding several removal-based explanation methods.

Below, we show that removal-based explanations that select sets of influential features are equivalent to proposing equal allocations to all players and then determining a high-valued coalition by examining each coalition's excess. Given equal allocations, players with high contributions will be less satisfied than players with low contributions, so solving this problem leads to a set of high- and low-valued players. We show this by reformulating each method's optimization problem as follows:

- (Minimize excess) MP isolates a low-value coalition by finding the coalition with the lowest excess, or the highest satisfaction, given equal allocations $z = \mathbf{1}\lambda$:

$$S^* = \arg\min_S \ e(\bar{S}, \mathbf{1}\lambda). \tag{3.38}$$

  The result S^* is a coalition whose complement $[d] \setminus S^*$ is most satisfied with the equal allocations.

- (Maximize excess) MIR isolates the smallest possible coalition that achieves a sufficient level of excess given allocations equal to zero. For a level of excess t, the optimization problem is given by:

$$S^* = \arg\min_S \ |S| \quad \text{s.t.} \ \ e(S, \mathbf{0}) \geq t. \tag{3.39}$$

  L2X and EP solve a similar problem but switch the objective and constraint. The optimization problem represents the coalition of size k with the highest excess given allocations of zero to each player:

$$S^* = \arg\max_S \ e(S, \mathbf{0}) \quad \text{s.t.} \ \ |S| = k. \tag{3.40}$$

Finally, FIDO-CA, INVASE and REAL-X find the coalition with the highest excess given equal allocations $z = \mathbf{1}\lambda$:

$$S^* = \arg\max_S \ e(S, \mathbf{1}\lambda). \tag{3.41}$$

- (Maximize difference in excess) MM combines the previous approaches. Rather than focusing on a coalition with low or high excess, MM partitions players into two coalitions while maximizing the difference in excess given equal allocations $z = \mathbf{1}\frac{\lambda}{1+\gamma}$:

$$S^* = \arg\max_S \ e\Big(S, \mathbf{1}\frac{\lambda}{1+\gamma}\Big) - \gamma e\Big(\bar{S}, \mathbf{1}\frac{\lambda}{1+\gamma}\Big). \tag{3.42}$$

  The result S^* is a coalition of dissatisfied players whose complement $[d] \setminus S^*$ is comparably more satisfied.

All of the approaches listed above are different formulations of the same multi-objective optimization problem: the intuition is that there is a small set of high-valued players S and a comparably larger set of low-valued players $[d] \setminus S$. Most methods focus on just one of these coalitions (MP focuses on $[d] \setminus S$, and MIR, L2X, EP, INVASE, REAL-X and FIDO-CA focus on S), while the optimization problem solved by MM considers both coalitions.

MM's summarization technique can therefore be understood as a generalization of the other methods. The MP and INVASE/REAL-X/FIDO-CA problems (eqs. 3.38 and 3.41), for example, are special cases of the MM problem. The other problems (eqs. 3.39 and 3.40) cannot necessarily be cast as special cases of the MM problem (eq. 3.42), but the MM problem resembles the Lagrangians of these constrained problems; more precisely, a special case of the MM problem shares the same optimal coalition with the dual to these constrained problems (Boyd et al., 2004).

By reformulating the optimization problems solved by each method, we can see that each feature selection summarization technique can be described as minimizing or maximizing excess given equal allocations for all players. In cooperative game theory, examining each

coalition's level of excess (or dissatisfaction) helps determine whether allocations will incentivize participation, and the same tool is used by removal-based explanations to find the most influential features for a model.

These feature selection approaches can be viewed as mappings of the form $E : \mathcal{U} \mapsto \mathcal{P}(d)$ because they identify coalitions $S^* \subseteq [d]$. Although the Shapley axioms apply only to mappings of the form $E : \mathcal{U} \mapsto \mathbb{R}^d$ (i.e., attribution methods), we find that these feature selection approaches satisfy certain analogous properties (Appendix A.3); however, the properties we identify are insufficient to derive an axiomatically unique method.

3.8.5 Summary

This discussion has shown that every removal-based explanation can be understood using ideas from cooperative game theory. Table 3.3 displays our findings regarding each method's game-theoretic interpretation and lists the relevant aspects of the literature for each summarization technique. Under our framework, removal-based explanations are implicitly based on an underlying cooperative game (Section 3.6), and these connections show that the model explanation field has in many cases either reinvented or borrowed ideas that were previously well-understood in cooperative game theory. We speculate that ongoing model explanation research may benefit from borrowing more ideas from cooperative game theory.

These connections are also important because they help use reason about the advantages of different summarization techniques via the properties that each method satisfies. Among the various techniques, we argue that the Shapley value provides the most complete explanation because it satisfies many desirable properties and gives a granular view of each player's contributions. Unlike the other methods, it divides the grand coalition's value (due to the efficiency property) while capturing the nuances of each player's contributions.

In contrast, the other methods have potential shortcomings, although such issues depend on the use case. Measuring a single marginal contribution, either by removing or including individual players, ignores player interactions; for example, removing individual players may lead to near-zero attributions for groups of correlated features, even if they are collectively

Table 3.3: Each method's summarization technique can be understood in terms of concepts from cooperative game theory.

Summarization	Methods	Related to
Shapley value	Shapley Net Effects, IME, QII, SHAP, TreeSHAP, KernelSHAP, LossSHAP, Shapley Effects, SAGE, SPVIM	Shapley value, probabilistic values, modeling cooperative games
Mean value when included	RISE	Banzhaf value, probabilistic values, modeling cooperative games
Remove/include individual players	Occlusion, PredDiff, CXPlain, permutation tests, univariate predictors, feature ablation (LOCO)	Probabilistic values, modeling cooperative games
Fit additive model	LIME	Shapley value, Banzhaf value, modeling cooperative games
High/low value coalitions	MP, EP, MIR, MM, L2X, INVASE, REAL-X, FIDO-CA	Maximum/minimum excess

important. The Banzhaf value provides a more nuanced view of each player's contributions, but it cannot be viewed as a division of the grand coalition's value because it violates the efficiency axiom (Dubey and Shapley, 1979). Finally, feature selection explanations provide only a coarse summary of each player's value contribution, and their results depend on user-specified hyperparameters; moreover, these methods are liable to select only one member out of a group of correlated features, which may be misleading to users.

3.9 Information-Theoretic Explanations

We now examine how removal-based explanations are connected to information theory. We begin by describing how features can be removed using knowledge of the underlying data

distribution, and we then show that many feature removal approaches approximate marginalizing out features using their conditional distribution. Finally, we prove that this approach gives every removal-based explanation an information-theoretic interpretation.

3.9.1 Removing Features Consistently

There are many ways to remove features from a model (Section 3.5.2), so we consider how to do so while accounting in some way for the underlying data distribution. Recall that removal-based explanations evaluate models while withholding groups of features using a subset function $F \in \mathfrak{F}$, which is typically a subset extension of an existing model $f \in \mathcal{F}$ (Definition 3.3). We begin by introducing a specific perspective on how to interpret a subset function's predictions, which are represented by $F(x_S)$.

Supervised ML models typically approximate the response variable's conditional distribution given the input. This is clear for classification models that estimate the conditional probability $f(x) \approx p(\boldsymbol{y} \mid x)$, but it is also true for regression models that estimate the conditional expectation $f(x) \approx \mathbb{E}[\boldsymbol{y} \mid x]$. (These approximations are implicit in conventional loss functions like cross entropy and MSE.) We propose that a subset function $F \in \mathfrak{F}$ can be viewed equivalently as a conditional probability/expectation estimate given *subsets* of features.

To illustrate this idea in the classification case, we denote the model's estimates as $q(\boldsymbol{y} \mid x) \equiv f(x)$, where the model's output is a discrete probability distribution. Although it is not necessarily equal to the true conditional distribution $p(\boldsymbol{y} \mid x)$, the estimate $q(\boldsymbol{y} \mid x)$ represents a valid probability distribution for all $x \in \mathcal{X}$. Similarly, we denote the subset function's estimates as $q(\boldsymbol{y} \mid x_S) \equiv F(x_S)$. A subset function F represents a set of conditional distributions $\{q(\boldsymbol{y} \mid \boldsymbol{x}_S) : S \subseteq [d]\}$, and this interpretation is important because we must consider whether these distributions are probabilistically valid. In particular, we must verify that standard probability laws (non-negativity, unitarity, countable additivity, Bayes rule) are not violated.

As we show below, this cannot be guaranteed without constraints on the subset function.

The laws of probability impose a relationship between $q(\boldsymbol{y} \mid x)$ and $q(\boldsymbol{y} \mid x_S)$ for any $x \in \mathcal{X}$ and $S \subset [d]$: they are linked by the underlying distribution on $\mathcal{X}$, or, more specifically, by the conditional distribution $\boldsymbol{x}_{\bar{S}} \mid x_S$. In fact, any distribution $q(\boldsymbol{x})$ implies a unique definition for $q(\boldsymbol{y} \mid \boldsymbol{x}_S)$ based on $q(\boldsymbol{y} \mid \boldsymbol{x})$ due to Bayes rule and the countable additivity property (described below). The flexibility of subset functions regarding how to remove features is therefore problematic, because certain removal approaches do not yield a valid set of conditional distributions.

Constraining the feature removal approach to be probabilistically valid can ensure that the model's subset extension F is faithful both to the original model f and an underlying data distribution. Building on this perspective, we define the notion of *consistency* between a subset function and a data distribution as follows.

Definition 3.5. *A subset function $F \in \mathfrak{F}$ that estimates a random variable $\boldsymbol{y}$'s conditional distribution is* ***consistent*** *with a data distribution $q(\boldsymbol{x})$ if its estimates satisfy the following properties:*

1. *(Countable additivity) The probability of the union of a countable number of disjoint events is the sum of their probabilities. Given events $A_1, A_2, \ldots$ such that $A_i \cap A_j = \varnothing$ for $i \neq j$, we have*
$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$
2. *(Bayes rule) The conditional probability $P(A \mid B)$ for events A and B is defined as*
$$P(A \mid B) = \frac{P(A, B)}{P(B)}.$$

This definition of consistency describes a class of subset functions that obey fundamental probability axioms (Laplace, 1781; Kolmogorov, 1950). Restricting a subset function to be consistent does not make its predictions correct, but it makes them compatible with a particular data distribution $q(\boldsymbol{x})$. Allowing for a distribution $q(\boldsymbol{x})$ that differs from the true

distribution $p(\boldsymbol{x})$ reveals that certain approaches implicitly assume modified data distributions.

Based on Definition 3.5, the next two results show that there is a unique subset extension $F \in \mathfrak{F}$ for a model $f \in \mathcal{F}$ that is consistent with a given data distribution $q(\boldsymbol{x})$ (proofs in Appendix A.4). The first result relates to subset extensions of classification models that estimate conditional probabilities.

Proposition 3.1. *For a classification model $f \in \mathcal{F}$ that estimates a discrete $\boldsymbol{y}$'s conditional probability, there is a unique subset extension $F \in \mathfrak{F}$ that is consistent with $q(\boldsymbol{x})$,*

$$F(x_S) = \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} | x_S)}[f(x_S, \boldsymbol{x}_{\bar{S}})],$$

where $q(\boldsymbol{x}_{\bar{S}} \mid x_S)$ is the conditional distribution induced by $q(\boldsymbol{x})$, i.e., the distribution

$$q(x_{\bar{S}} \mid x_S) = \frac{q(x_{\bar{S}}, x_S)}{\int_{\boldsymbol{x}_{\bar{S}}} q(\boldsymbol{x}_{\bar{S}}, x_S)}.$$

The next result arrives at a similar conclusion, but it is specific to subset extensions of regression models that estimate the response variable's conditional expectation.

Proposition 3.2. *For a regression model $f \in \mathcal{F}$ that estimates a real-valued $\boldsymbol{y}$'s conditional expectation, there is a unique subset extension $F \in \mathfrak{F}$ that is consistent with $q(\boldsymbol{x})$,*

$$F(x_S) = \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} | x_S)}[f(x_S, \boldsymbol{x}_{\bar{S}})],$$

where $q(\boldsymbol{x}_{\bar{S}} \mid x_S)$ is the conditional distribution induced by $q(\boldsymbol{x})$.

These results differ in their focus on classification and regression models, but the conclusion in both cases is the same: the only subset extension of a model f that is consistent with $q(\boldsymbol{x})$ is one that averages the full model output $f(\boldsymbol{x})$ over the distribution of values $\boldsymbol{x}_{\bar{S}}$ given by $q(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S)$.

When defining a model's subset extension, the natural choice is to make it consistent with the true data distribution, or $q(\boldsymbol{x}) = p(\boldsymbol{x})$. This yields precisely the approach presented by SHAP (the conditional version), SAGE, and several other methods, which is to marginalize out the removed features using their conditional distribution $p(\boldsymbol{x}_{\bar{S}} \mid x_S)$ (Strobl et al., 2008; Zintgraf et al., 2017; Lundberg and Lee, 2017; Aas et al., 2019; Covert et al., 2020; Frye et al., 2021).

Besides this approach, only a few other approaches are consistent with any distribution. The QII approach (eq. 3.6) is consistent with a distribution that is the product of marginals, $q(\boldsymbol{x}) = \prod_{i=1}^{d} p(\boldsymbol{x}_i)$. The more recent IME approach (eq. 3.7) is consistent with a distribution that is the product of uniform distributions, $q(\boldsymbol{x}) = \prod_{i=1}^{d} u_i(\boldsymbol{x}_i)$. And finally, any approach that sets features to default values $b \in \mathcal{X}$ (eqs. 3.1 and 3.2) is consistent with a distribution that puts all of its mass on those values—which we refer to as a *constant distribution*. These approaches all achieve consistency because they are based on a simplifying assumption of feature independence.

3.9.2 Conditional Distribution Approximations

While few removal-based explanations explicitly suggest marginalizing out features using their conditional distribution, several methods can be understood as approximations of this approach. These represent practical alternatives to the exact conditional distribution, which is unavailable in practice and often difficult to estimate.

The core challenge in using the conditional distribution is modeling it accurately. Methods that use the marginal distribution sample rows from the dataset (Breiman, 2001; Lundberg and Lee, 2017), and it is possible to filter for rows that agree with the features to be conditioned on (Sundararajan and Najmi, 2019); however, this technique does not work well for high-dimensional or continuous-valued data. A relaxed alternative to this approach is using cohorts of rows with similar values (Mase et al., 2019).

While properly representing the conditional distribution for every subset of features is challenging, there are several approaches that provide either rough or high-quality approxi-

mations. These approaches include:

- **Assume feature independence.** If we assume that the features $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$ are independent, then the conditional distribution $p(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S)$ is equivalent to the joint marginal distribution $p(\boldsymbol{x}_{\bar{S}})$, and it is even equivalent to the product of marginals $\prod_{i \in \bar{S}} p(\boldsymbol{x}_i)$. The removal approaches used by KernelSHAP and QII can therefore be understood as rough approximations to the conditional distribution that assume feature independence (Datta et al., 2016; Lundberg and Lee, 2017).

- **Assume model linearity (and feature independence).** As Lundberg and Lee (2017) pointed out, replacing features with their mean can be interpreted as an additional assumption of model linearity:

$$\begin{aligned}
\mathbb{E}[f(\boldsymbol{x}) \mid x_S] &= \mathop{\mathbb{E}}_{p(\boldsymbol{x}_{\bar{S}} | x_S)}[f(x_S, \boldsymbol{x}_{\bar{S}})] && \text{(Conditional distribution)} \\
&\approx \mathop{\mathbb{E}}_{p(\boldsymbol{x}_{\bar{S}})}[f(x_S, \boldsymbol{x}_{\bar{S}})] && \text{(Assume feature independence)} \\
&\approx f(x_S, \mathbb{E}[\boldsymbol{x}_{\bar{S}}]). && \text{(Assume model linearity)}
\end{aligned}$$

  While this pair of assumptions rarely holds in practice, particularly with the complex models for which explanation methods are most useful, it provides some justification for methods that replace features with default values (e.g., LIME, Occlusion, MM, CXPlain, RISE).

- **Parametric assumptions.** Recent work on Shapley values has proposed parametric approximations of the conditional distribution, e.g., multivariate Gaussian and copula-based models (Aas et al., 2019, 2021). While the parametric assumptions may not hold exactly, these approaches can provide better conditional distribution approximations than feature independence assumptions.

- **Generative model.** FIDO-CA proposes removing features by drawing samples from

a conditional generative model (Chang et al., 2018). If the generative model p_G (e.g., a conditional GAN) is trained to optimality, then it produces samples from the true conditional distribution. We can then write

$$p_G(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S) \overset{d}{=} p(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S), \tag{3.43}$$

where $\overset{d}{=}$ denotes equality in distribution. Given a sample $\tilde{x}_{\bar{S}} \sim p_G(\boldsymbol{x}_{\bar{S}} \mid x_S)$, the prediction $f(x_S, \tilde{x}_{\bar{S}})$ can be understood as a single-sample Monte Carlo approximation of the expectation $\mathbb{E}[f(\boldsymbol{x}) \mid x_S]$. Agarwal and Nguyen (2020) substituted the generative model approach into several existing methods (Occlusion, MP, LIME) and observed improvements in their performance across numerous metrics. Frye et al. (2021) demonstrated a similar approach using a variational autoencoder-like model (Ivanov et al., 2018), and future work could leverage other conditional generative models (Douglas et al., 2017; Belghazi et al., 2019).

- **Surrogate model.** Several methods require a surrogate model that is trained to match the original model's predictions given subsets of features, where missing features are represented by zeros (or other values that do not appear in the dataset). This circumvents the task of modeling an exponential number of conditional distributions and is equivalent to parameterizing a subset function $F \in \mathfrak{F}$ and training it with the following objective:

$$\min_{F} \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} [\ell(F(\boldsymbol{x}_{\boldsymbol{S}}), f(\boldsymbol{x}))]. \tag{3.44}$$

In Appendix A.5, we prove that for certain loss functions ℓ, the subset function F that optimizes this objective is equivalent to marginalizing out features using their conditional distribution. Frye et al. (2021) show a similar result for MSE loss, and we also show that a cross entropy loss can be used for classification models. Finally, we find that L2X (Chen et al., 2018a) may not provide a faithful approximation because the subsets S are not distributed independently from the inputs $\boldsymbol{x}$—an issue recently

addressed by REAL-X (Jethani et al., 2021a).

- **Missingness during training.** Rather than training a surrogate with missing features, we can instead learn the original model with missingness introduced during training. This is equivalent to parameterizing a subset function $F \in \mathfrak{F}$ and optimizing the following objective, which resembles the standard training loss:

$$\min_{F} \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} [\ell(F(\boldsymbol{x}_{\boldsymbol{S}}), \boldsymbol{y})]. \tag{3.45}$$

  In Appendix A.5, we prove that if the model optimizes this objective, then it is equivalent to marginalizing out features from the full model predictions $F(x)$ using the conditional distribution. An important aspect of this objective is that the subsets S must be independently distributed from the data $(\boldsymbol{x}, \boldsymbol{y})$. INVASE (Yoon et al., 2018) does not satisfy this property, which may prevent it from providing an accurate approximation.

- **Separate models.** Finally, Shapley Net Effects (Lipovetsky and Conklin, 2001), SPVIM (Williamson and Feng, 2020) and the original IME (Štrumbelj et al., 2009) propose training separate models for each feature subset, or $\{f_S : S \subseteq [d]\}$. If every model is optimal, then this approach is equivalent to marginalizing out features with their conditional distribution (Appendix A.5). This is due to a relationship that arises between models that optimize the population risk for different sets of features; for example, with cross entropy loss, the optimal model (the Bayes classifier) for $\boldsymbol{x}_S$ is given by $f_S(x_S) = p(\boldsymbol{y} \mid x_S)$, which is equivalent to $\mathbb{E}[f_{[d]}(\boldsymbol{x}) \mid x_S]$ because the optimal model given all features is $f_{[d]}(x) = p(\boldsymbol{y} \mid x)$.

This discussion shows that although few methods explicitly suggest removing features with the conditional distribution, numerous methods approximate this approach. Training separate models should provide the best approximation because each model is given a

relatively simple prediction task, but this approach is unable to scale to high-dimensional datasets. The generative model approach amortizes knowledge of an exponential number of conditional distributions into a single model, which is more scalable and effective for image data (Yu et al., 2018). The supervised surrogate and missingness during training approaches also require learning up to just one additional model, and these are trained with far simpler optimization objectives (eqs. 3.44 and 3.45) than conditional generative models.

We conclude that, under certain assumptions of feature independence or model optimality, several feature removal strategies are consistent with the data distribution $p(\boldsymbol{x})$. Our definition of consistency provides a new lens for comparing different feature removal strategies, and Table 3.4 summarizes our findings.

3.9.3 Connections With Information Theory

Conventional wisdom suggests that explanation methods quantify the information contained in each feature. However, we find that precise information-theoretic connections can be identified only when held-out features are marginalized out with their conditional distribution. Our analysis expands on prior work by showing that every removal-based explanation has a probabilistic or information-theoretic interpretation when features are removed properly (Owen, 2014; Chen et al., 2018a; Covert et al., 2020).

To aide our presentation, we assume that the model f is optimal, i.e., it is the Bayes classifier $f(x) = p(\boldsymbol{y} \mid x)$ for classification tasks or the conditional expectation $f(x) = \mathbb{E}[\boldsymbol{y} \mid x]$ for regression tasks. This assumption is optimistic, but because models are typically trained to approximate one of these functions, the resulting explanations are approximately based on the information-theoretic quantities derived here.

By assuming model optimality and marginalizing out removed features with their conditional distribution, we can guarantee that the prediction given any subset of features is optimal for those features. Specifically, we have the Bayes classifier $F(x_S) = p(\boldsymbol{y} \mid x_S)$ in the classification case and the conditional expectation $F(x_S) = \mathbb{E}[\boldsymbol{y} \mid x_S]$ in the regression case. Using these subset functions, we can derive probabilistic and information-theoretic

Table 3.4: Consistency properties of different feature removal strategies.

Removal	Methods	Consistency
Marginalize (conditional)	Cond. permutation tests, PredDiff, SHAP, LossSHAP, SAGE, Shapley Effects	Consistent with $p(\boldsymbol{x})$
Generative model	FIDO-CA	Consistent with $p(\boldsymbol{x})$ (assuming model optimality)
Supervised surrogate	L2X, REAL-X, Frye et al. (2021)	
Missingness during training	INVASE	
Separate models	Feature ablation (LOCO), univariate predictors, Shapley Net Effects, SPVIM, IME (2009)	
Marginalize (marginal)	Permutation tests, KernelSHAP	Consistent with $p(\boldsymbol{x})$ (assuming independence)
Marginalize (marginals product)	QII	
Marginalize (marginals product)	QII	Consistent with $q(\boldsymbol{x})$ with feature independence
Marginalize (uniform)	IME (2010)	
Zeros	Occlusion, PredDiff RISE, CXPlain	Consistent with constant distributions $q(\boldsymbol{x})$
Default values	LIME (images), MM	
Tree distribution	TreeSHAP	Not consistent with any $q(\boldsymbol{x})$
Extend pixel values	MIR	
Blurring	MP, EP	Not valid $F \in \mathfrak{F}$
Marginalize (replacement dist.)	LIME (tabular)	

interpretations for each explanation method.

These connections focus on the set functions analyzed by each removal-based explanation (Section 3.6). We present results for classification models that use cross entropy loss here, and we show analogous results for regression models in Appendix A.6. Under the assumptions

described above, the set functions analyzed by each method can be interpreted as follows:

- The set function $u_x(S) = F(x_S)$ represents the response variable's conditional probability for the chosen class y:

$$u_x(S) = p(y \mid x_S). \tag{3.46}$$

  This lets us examine each feature's true association with the response variable.

- The set function $v_{xy}(S) = -\ell(F(x_S), y)$ represents the log probability of the correct class y, which is equivalent to the *pointwise mutual information* $\mathrm{I}(y; x_S)$ (up to a constant value):

$$v_{xy}(S) = \mathrm{I}(y; x_S) + c. \tag{3.47}$$

  This quantifies how much information x_S contains about the outcome y, or how much less surprising y is given knowledge of x_S (Fano, 1961).

- The set function $v_x(S) = -\mathbb{E}_{p(\boldsymbol{y}|x)}[\ell(F(x_S), \boldsymbol{y})]$ represents the negative Kullback-Leibler (KL) divergence between the label's conditional distribution and its partial conditional distribution (up to a constant value):

$$v_x(S) = c - D_{\mathrm{KL}}(p(\boldsymbol{y} \mid x) \mid\mid p(\boldsymbol{y} \mid x_S)). \tag{3.48}$$

  As mentioned by Yoon et al. (2018), this provides an information-theoretic measure of the deviation between the response variable's true distribution and its distribution when conditioned on a subset of features (Cover and Thomas, 2012).

- The set function $v(S) = -\mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}[\ell(F(\boldsymbol{x}_S), \boldsymbol{y})]$ represents the *mutual information* with the response variable (up to a constant value):

$$v(S) = \mathrm{I}(\boldsymbol{y}; \boldsymbol{x}_S) + c. \tag{3.49}$$

As discussed by Covert et al. (2020), this quantifies the amount of information, or the predictive power, that the features $\boldsymbol{x}_S$ communicate about the response variable $\boldsymbol{y}$ (Cover and Thomas, 2012).

- The set function $w_x(S) = -\ell(F(x_S), f(x))$ represents the KL divergence between the full model output and its output given a subset of features. Specifically, if we define $\boldsymbol{z}$ to be a categorical random variable $\boldsymbol{z} \sim \text{Cat}(f(\boldsymbol{x}))$, then we have:

$$w_x(S) = c - D_{\text{KL}}(p(\boldsymbol{z} \mid x) \mid\mid p(\boldsymbol{z} \mid x_S)). \tag{3.50}$$

This result does not require model optimality, but under the assumption that f is the Bayes classifier, this quantity is equivalent to the KL divergence between the label's conditional and partial conditional distribution (up to a constant value):

$$w_x(S) = c - D_{\text{KL}}(p(\boldsymbol{y} \mid x) \mid\mid p(\boldsymbol{y} \mid x_S)). \tag{3.51}$$

We can therefore see that under these assumptions, L2X, INVASE and REAL-X are in a sense based on the same set function and have a similar information-theoretic interpretation (Chen et al., 2018a; Yoon et al., 2018; Jethani et al., 2021a).

- The set function $w(S) = -\mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}[\ell(F(\boldsymbol{x}_S), f(\boldsymbol{x}))]$ represents the information that $\boldsymbol{x}_S$ communicates about the model output $f(\boldsymbol{x})$. Specifically, if we let $\boldsymbol{z} \sim \text{Cat}(f(\boldsymbol{x}))$, then we have:

$$w(S) = \text{I}(\boldsymbol{z}; \boldsymbol{x}_S) + c. \tag{3.52}$$

This is a version of the conditional variance decomposition provided by Shapley Effects, but for classification tasks (Owen, 2014). This result does not require model optimality, but if f is the Bayes classifier, then this is equivalent to the mutual information with

Table 3.5: Each method's underlying set function has an information-theoretic interpretation when features are removed appropriately.

Model Behavior	Set Function	Methods	Related to
Prediction	u_x	Occlusion, MIR, MM, IME, QII, LIME, MP, EP, FIDO-CA, RISE, SHAP, KernelSHAP, TreeSHAP	Conditional probability, conditional expectation
Prediction loss	v_{xy}	LossSHAP, CXPlain	Pointwise mutual information
Prediction mean loss	v_x	INVASE	KL divergence with conditional distribution
Dataset loss	v	Permutation tests, univariate predictors, feature ablation (LOCO), Shapley Net Effects, SAGE, SPVIM	Mutual information (with label)
Prediction loss (output)	w_x	L2X, REAL-X	KL divergence with full model output
Dataset loss (output)	w	Shapley Effects	Mutual information (with output)

the response variable (up to a constant):

$$w(S) = \mathrm{I}(\boldsymbol{y}; \boldsymbol{x}_S) + c. \tag{3.53}$$

As noted, two assumptions are required to derive these results. The first is that features are marginalized out using the conditional distribution; although many methods use different removal approaches, they can be modified to use this approach or an approximation. The second is that models are optimal; this assumption rarely holds in practice, but since conventional loss functions train models to approximate either the Bayes classifier or the conditional expectation, we can view these information-theoretic quantities as the values

that each set function approximates.

We conclude that when features are removed appropriately, explanation methods quantify the information communicated by each feature (see summary in Table 3.5). No single set function provides the "right" approach to model explanation; rather, these information-theoretic quantities span a range of perspectives that could be useful for understanding a complex ML model.

Removal-based explanations that are consistent with the observed data distribution can provide well-grounded insight into intrinsic statistical relationships in the data, and this is useful for finding hidden model influences (e.g., detecting bias from sensitive attributes) or when using ML as a tool to discover real-world relationships. However, this approach has the potentially undesirable property that features may appear important even if they are not used by the model in a functional sense (Merrick and Taly, 2019; Chen et al., 2020). When users are more interested in the model's mechanism for calculating predictions, other removal approaches may be preferable, such as interventional approaches motivated by a causal analysis of the model (Janzing et al., 2020; Heskes et al., 2020).

3.10 A Cognitive Perspective on Removal-Based Explanations

The previous sections provide a mathematical perspective on removal-based explanations, so we now consider this class of methods through a different lens: that of the social sciences. Analyzing this broad class of methods provides an opportunity to discuss how they all relate to cognitive psychology due to their shared reliance on feature removal.

Model explanation tools are not typically designed based on research from the social sciences (Miller et al., 2017), but, as we show, removal-based explanations have clear parallels with cognitive theories about how humans understand causality. We first discuss our framework's foundation in counterfactual reasoning and then describe a trade-off between simple explanations and those that convey richer information about models.

3.10.1 Subtractive Counterfactual Reasoning

Explaining a model's predictions is fundamentally a causality question: *what makes the model behave this way?* Each input feature is a potential cause, multiple features may be causal, and explanations should quantify each feature's degree of influence on the model. We emphasize the distinction between this model-focused causality and causality between the input and response (e.g., whether a feature causes the outcome), because real-world causality is difficult to discern from observational data (Pearl, 2009).

In philosophy and psychology, *counterfactual reasoning* is a standard tool for understanding causality. A counterfactual example changes certain facts of a situation (e.g., the route a person drove to get home) to potentially achieve a different outcome (e.g., getting home safely), and counterfactuals shed light on whether each aspect of a situation caused the actual outcome (e.g., a fatal car crash). In an influential philosophical account of causality, John Stuart Mill presented five methods of induction that use counterfactual reasoning to explain cause-effect relationships (Mill, 1884; Mackie, 1974). In the psychology literature, counterfactual thinking is the basis of multiple theories about how people explain the causes of events (Kahneman and Tversky, 1982; Hilton, 1990).

Removal-based explanations perform a specific type of counterfactual reasoning. In psychology, the process of removing an event to understand its influence on an outcome is called a *subtractive counterfactual* (Epstude and Roese, 2008), and this is precisely how removal-based explanations work. In philosophy, the same principle is called the *method of difference*, and it is one of Mill's five methods for inferring cause-effect relationships (Mill, 1884). This type of logic is also found in cognitive theories about how people understand and discuss causality (Kahneman and Tversky, 1982; Jaspars et al., 1983; Hilton, 1990).

The principle of removing something to examine its influence is pervasive in social sciences, not only as a philosophical approach but as part of descriptive psychological theories; this explains the remarkable prevalence of the feature removal principle in model explanation, even among computational researchers who were not explicitly inspired by psychology

research. Perhaps surprisingly, the reliance on subtractive counterfactual reasoning (or equivalently, the method of difference) has been overlooked thus far, even by work that examined the psychological basis for SHAP (Merrick and Taly, 2019; Kumar et al., 2020).

Some prior work applies a different form of counterfactual reasoning to model explanation (Verma et al., 2020). For example, one influential approach suggests showing users counterfactuals that adjust a small number of features to change the model output (Wachter et al., 2017). This approach undoubtedly provides information about how a model works, but many such counterfactuals are required to fully illustrate each feature's influence. Removal-based explanations can provide more insight by concisely summarizing the results of many subtractive counterfactuals, e.g., via Shapley values.

3.10.2 Norm Theory and the Downhill Rule

Subtractive counterfactuals are an intuitive way to understand each feature's influence, but their implementation is not straightforward. Removal-based explanations aim to remove the information that a feature communicates, or subtract the fact that it was observed, but it is not obvious how to do this: given an input $x \in \mathcal{X}$ and subset $S \subseteq [d]$, it is unclear how to retain x_S while removing $x_{\bar{S}}$. We consult two psychological theories to contextualize the approaches that have been considered by different methods (Section 3.5).

Many removal-based explanations remove features by averaging the model output over a distribution of possible values for those features; this has a clear correspondence with the cognitive model described by *Norm theory* (Kahneman and Miller, 1986). According to this theory, people assess normality by gathering summary statistics from a set of recalled and simulated representations of a phenomenon (e.g., loan application outcomes for individuals with a set of characteristics). In these representations, certain features are fixed (or *immutable*) while others are allowed to vary (*mutable*); in our case these correspond to the retained and removed features, respectively.

Taking inspiration from Norm theory, we may equate a model's behavior when certain features are blocked from exerting influence (i.e., the removed features) with a "normal"

outcome for the remaining features. With this perspective, we can see that Norm theory provides a cognitive justification for averaging the model output over a distribution of values for the removed features. Merrick and Taly (2019) make a similar observation to justify how SHAP removes features.

The choice of distribution for averaging outcomes is important, but Norm theory does not prescribe a specific approach. Rather, Norm theory is a descriptive cognitive model that recognizes that individuals may have different perspectives of normality based on their experiences (Kahneman and Miller, 1986). Future model explanation research may consider how to tailor explanations to each user, as suggested by Miller (2019), but we also require systematic methods that do not solicit user input. We therefore consider whether any approach used by existing methods is justifiable from a cognitive perspective.

For guidance on the choice of distribution, we look to research on human tendencies when assigning blame. In their study of *mental undoing*, Kahneman and Tversky (1982) examined people's biases when proposing counterfactuals that change an undesirable event's outcome. One of their clearest findings was the *downhill rule*, which states that people are more likely to propose changes that remove a surprising aspect of a story or otherwise increase the story's internal coherence. In other words, people are more likely to assign blame to an aspect of a situation if it has a more likely alternative that would change the outcome.

One feature removal strategy is reminiscent of the downhill rule, because it considers alternative values in proportion to their plausibility: when marginalizing out removed features using their conditional distribution, alternative values and their corresponding outcomes are averaged in proportion to the coherence of the full feature set, which is represented by the data distribution $p(\boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_S) \propto p(\boldsymbol{x})$. This is consistent with the downhill rule because if certain high-likelihood values change the outcome, their influence on the model will be apparent when integrating $f(\boldsymbol{x})$ over the distribution $p(\boldsymbol{x} \mid x_S)$.

In summary, Norm theory provides a cognitive analogue for removing features by averaging over a distribution of alternative values, and the downhill rule suggests that the plausibility or likelihood of alternative values should be taken into account. These theories

provide cognitive justification for certain approaches to removing features, but our review of relevant psychology research is far from comprehensive. However, interestingly, our findings lend support to the same approach that yields connections with information theory (Section 3.9), which is marginalizing out features using their conditional distribution.

3.10.3 Simplicity Versus Completeness

Building on our discussion of the human psychology aspect of removal-based explanations, we now discuss a trade-off between the amount of information conveyed by an explanation and the user's likelihood of drawing the correct conclusions. We describe this trade-off and then show that this class of methods provides the flexibility to balance these competing objectives.

Consider two explanation strategies with different levels of complexity. A counterfactual example that perturbs several features to change the model's prediction is easy to understand, but it does not convey detailed information about a model's dependence on each feature (Wachter et al., 2017). By contrast, SHAP's feature attributions provide a complex summary of each feature's influence by quantifying the impact of removing different groups of features. Perhaps due to their greater complexity, a recent user study showed that users were less likely to understand SHAP visualizations, experiencing a higher cognitive load than users who viewed simpler explanations (Kaur et al., 2020). Similarly, a different study found that longer explanations required more time and effort to understand, in some cases impeding a user's ability to draw appropriate conclusions (Lage et al., 2019).

We view this as a trade-off between simplicity and completeness, because explanations are typically more complex when they provide more information about a model. Providing a more complete characterization may be preferable for helping users build a mental picture of how a model works, but complicated explanations also risk overloading users, potentially leading to a false or limited sense of model comprehension.

Recognizing this trade-off and its implications, we consider simplicity as a design goal and find that removal-based explanations have the flexibility to adjust the balance between these

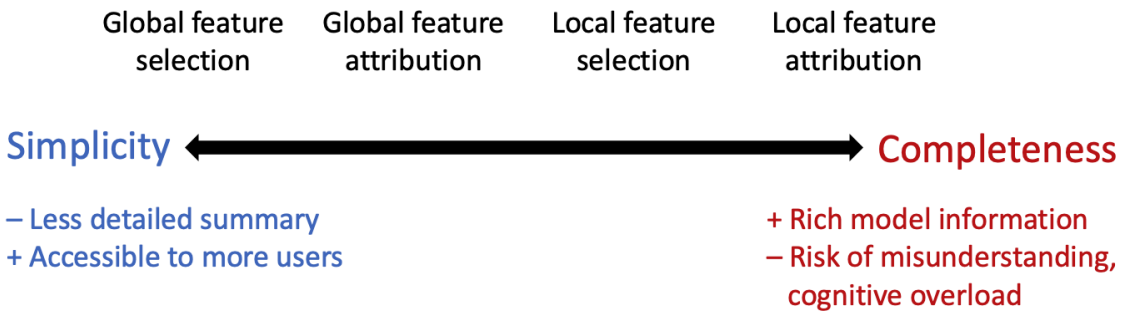


Figure 3.4: Each model explanation strategy represents a trade-off between an explanation's simplicity and its completeness.

goals. To reduce cognitive burden, one might focus on global rather than local explanations, e.g., by providing visualizations that display many local explanations (Lundberg et al., 2020), or by using global methods that summarize a model's behavior across the entire dataset (Owen, 2014; Covert et al., 2020). Alternatively, one may generate explanations that operate on feature groups rather than individual features, e.g., sets of correlated features or nearby pixels (Zeiler and Fergus, 2014; Ribeiro et al., 2016).

Certain explanation formats convey richer information than others (Figure 3.4). For example, feature attributions provide a granular view of a model by considering every feature as a cause and quantifying each feature's influence, and local explanations are more granular than global ones. Richer information may not always be desirable, because psychology research shows that people report higher satisfaction with explanations that cite fewer causes (Thagard, 1989; Read and Marcus-Newhall, 1993; Lombrozo, 2007; Miller, 2019). Users may therefore derive more insight from explanations that highlight fewer causes, such as the sparse feature attributions provided by LIME (Ribeiro et al., 2016).

Feature selection explanations offer the potential to go even further in the direction of simplicity. These explanations directly penalize the number of selected features (Section 3.7), guaranteeing that fewer features are labeled as important; and furthermore, they omit information about the granularity of each feature's influence. For a non-technical user, it may be

simpler to understand that a model's prediction was dominated by a small number of highly informative features, whereas it may require a more sophisticated user to interpret real-valued attributions for individual features, e.g., as coefficients in an additive decomposition of a model's behavior (Section 3.8).

Put simply, an explanation that paints an incomplete picture of a model may prove more useful if the end-users are able to understand it properly. Designers of explanation methods should be wary of overestimating people's abilities to store complex mental models (Norman, 1983), and the ML community can be mindful of this by tailoring explanations to users' degrees of sophistication.

3.11 Experiments

We have thus far analyzed removal-based explanations from a primarily theoretical standpoint, so we now conduct experiments to provide a complementary empirical perspective. Our experiments aim to accomplish three goals:

1. Implement and compare many new methods by filling out the space of removal-based explanations (Figure 3.2).

2. Demonstrate the advantages of removing features by marginalizing them out using their conditional distribution—an approach that we showed provides information-theoretic explanations (Section 3.9).

3. Verify the existence of relationships between various explanation methods. Specifically, explanations may be similar if they use (i) summary techniques that are probabilistic values of the same cooperative game (Section 3.8), or (ii) feature removal strategies that are approximately equivalent (Section 3.9).

To cover a wide range of methods, we consider many combinations of removal strategies (Section 3.5), as well as model behaviors (Section 3.6) and summary techniques (Section 3.7).

Our implementation is available online,[9] and we tested 80 total methods (68 of which are new) that span our framework as follows:

- For *feature removal*, we considered replacing features with default values, and marginalizing out features using either (i) uniform distributions, (ii) the product of marginals, or (iii) the joint marginal distribution. Next, to approximate marginalizing out features using their conditional distribution, we trained surrogate models to match the original model's predictions with held-out features (Section 3.9). Finally, for one dataset we also trained separate models with all feature subsets.

- Our experiments analyze three *model behaviors* using three different datasets. We explained individual classification probabilities for the census income dataset (Lichman et al., 2013), the model's loss on individual predictions for MNIST (LeCun et al., 2010), and the dataset loss for a breast cancer subtype classification task (Berger et al., 2018).

- For *summary techniques*, we considered removing or including individual features, the mean when included strategy, and Banzhaf and Shapley values. For techniques that involve an exponential number of feature subsets, we used sampling-based approximations similar to those previously used for Shapley values (Štrumbelj and Kononenko, 2010), and we detected convergence based on the width of confidence intervals.[10]

Implementing and comparing all combinations of these choices helps us better understand the relationships between methods and identify the most promising approaches. For more details about the models, datasets and hyperparameters, see Appendix A.7.

---

[9] `https://github.com/iancovert/removal-explanations`

[10] We follow the approach from Covert et al. (2020), which identifies convergence via the ratio of confidence interval width to the difference between the minimum and maximum values.

3.11.1 Census Income

The census income dataset provides basic demographic information about individuals, and the task is to predict whether a person's annual income exceeds $50k. We trained a LightGBM model (Ke et al., 2017) and then generated explanations using the combinations of removal and summary strategies described above, including training separate models for all feature subsets (as there are only 12 features). For the default values removal strategy, we used the mean for continuous features and the mode for discrete ones.

These combinations of choices resulted in 30 distinct explanation methods, several of which are nearly equivalent to existing approaches (SHAP, QII, IME, Occlusion, RISE, PredDiff),[11] but most of which are new. Figure 3.5 shows a grid of explanations generated for a single person whose income did not exceed $50k. We offer two qualitative remarks about the relationships between these explanations:

1. The explanations are sometimes similar despite using different feature removal strategies (see the bottom four rows of Figure 3.5). This is most likely because these removal approaches all approximate the conditional distribution (product, marginal, surrogate, separate models). The first two rows (default values, uniform) deviate from the others because they offer low-quality conditional distribution approximations.

2. The explanations are sometimes similar despite using different summary techniques (see the columns for include individual, Banzhaf and Shapley values). The similarity between probabilistic values suggests that each feature's marginal contributions are largely similar, at least outside of a saturating regime; the remove individual technique deviates from the others, possibly due to saturation effects when most features are included. In contrast, the mean when included technique differs significantly from the others because it is not a probabilistic value (in the game-theoretic sense).

[11] There are some implementation differences, e.g., our conditional distribution approximation for PredDiff.

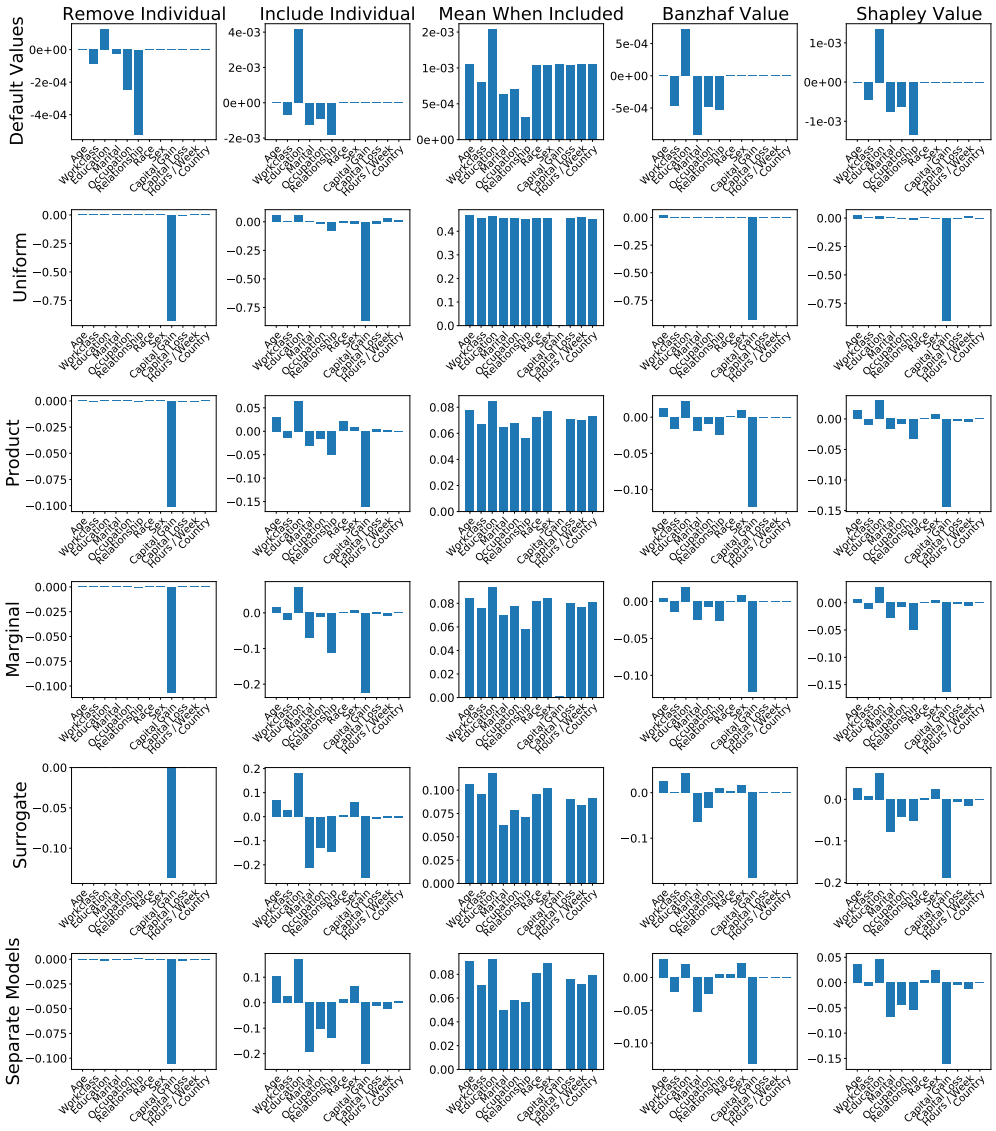


Figure 3.5: Explanations for a single example in the census income dataset. Each bar chart represents attribution values for a different explanation method. The vertical axis represents feature removal strategies and the horizontal axis represents summary techniques.

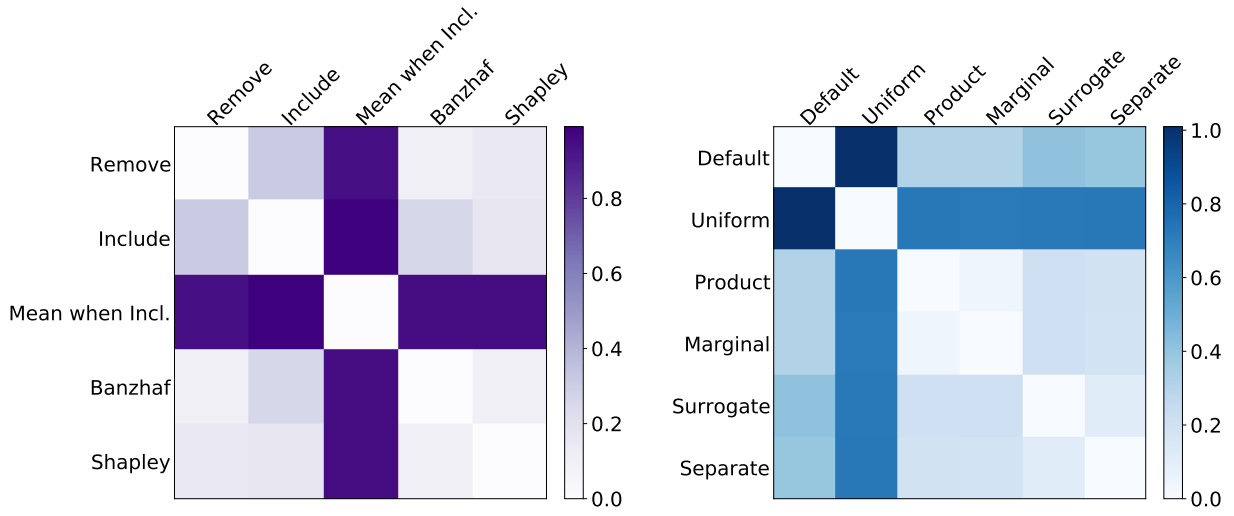


Figure 3.6: Census income explanation differences. The colored entries in each plot represent the mean Euclidean distance between explanations that differ in only their summary technique (left) or their feature removal strategy (right). Lighter entries indicate greater similarity.

To verify whether these relationships hold in general, we scaled up our comparison by generating explanations for 256 instances and quantifying the similarity between methods. We calculated the mean Euclidean distance between explanations generated by each method (averaged across the instances), and, rather than displaying the pair-wise distances for all 30 methods, we only considered comparisons between methods that differ just in their summary techniques or just in their removal strategies.

Figure 3.6 shows the results, with the distances between explanations grouped by either summary technique (left) or by feature removal strategy (right). Regarding the summary techniques, the clearest finding is that the mean when included approach produces very different explanations than all the other techniques. Among the remaining ones, Shapley values are relatively close to Banzhaf values, and they are similarly close to explanations that remove or include individual features; this matches the Shapley value's formulation, because like the Banzhaf value, it is a weighted average of all marginal contributions.

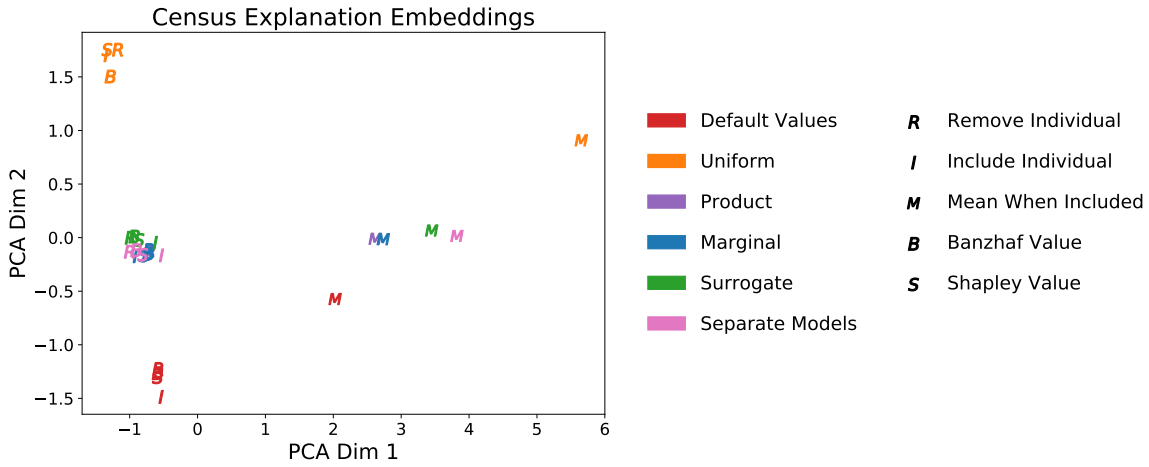


Figure 3.7: Census income explanation embeddings. PCA was used to generate two-dimensional embeddings for each method, allowing us to observe which methods tend to produce similar results.

Regarding the different feature removal strategies (Figure 3.6 right), we observe that the default values and uniform marginalization strategies produce very different explanations than the other approaches. The remaining approaches are relatively similar; for example, the product of marginals and joint marginal produce nearly identical explanations, which suggests that the feature dependencies either are not strong in this dataset or are not captured by our model. We also observe that the surrogate approach is closest to training separate models, as we would expect, but that they are not identical.

Next, we visualized the different explanations using low-dimensional embeddings (Figure 3.7). We generated embeddings by applying principal components analysis (PCA, Jolliffe, 1986) to vectors containing the concatenated explanations for all 256 explanations (i.e., 30 vectors of size 3,072). The results further support our observation that the mean when included technique differs most from the others. The explanations whose feature removal strategy approximates the conditional distribution are strongly clustered, with the product of marginals and joint marginal approaches overlapping almost entirely.

Finally, since many feature removal strategies can be understood as approximating the

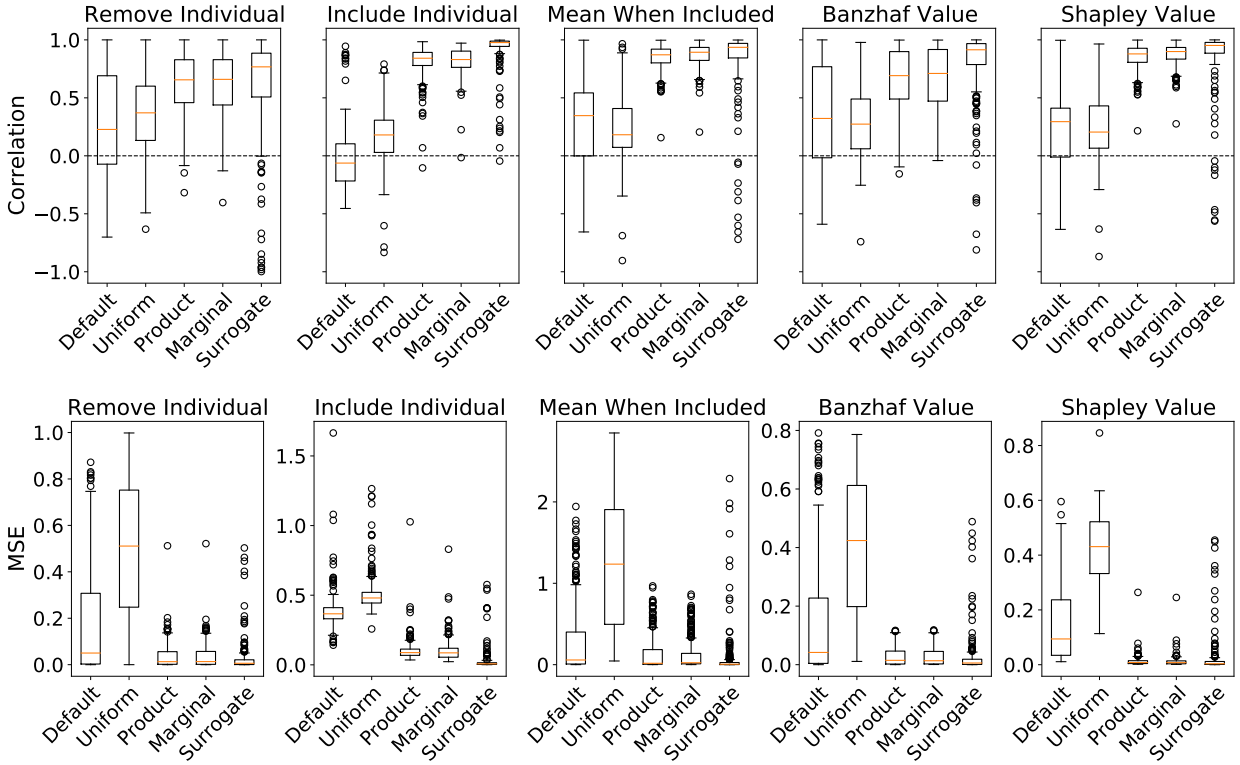


Figure 3.8: Boxplots quantifying the similarity of census income explanations to comparable explanations generated by using separate models to remove features. The similarity metrics are correlation (top, higher is better) and MSE (bottom, lower is better).

conditional distribution approach (Section 3.9), we attempted to quantify the approximation quality. We cannot measure this exactly because we lack access to the conditional distributions, so we considered the explanations generated with separate models to be our ground truth, because this is our most reliable proxy. To measure each method's faithfulness to the underlying data distribution, we grouped explanations by their summary technique (e.g., Banzhaf value, Shapley value) and quantified their similarity to explanations generated with separate models (Figure 3.8).

We calculated two similarity metrics, MSE and correlation, and both show that the surrogate approach is closest to training separate models. This reflects the fact that both

approaches provide flexible approximations to marginalizing out features using their conditional distribution (Section 3.9). The default and uniform approaches tend to produce very different explanations. The product and marginal approaches are sometimes competitive with the surrogate approach, but they are noticeably less similar in most cases. We remark, however, that the surrogate approach has more outliers; this suggests that although it is closest to using separate models on average, the surrogate approach is prone to occasionally making large errors.

In sum, these results show that the surrogate approach provides a reliable proxy for the conditional distribution, producing explanations that are faithful to the underlying data distribution. Unlike the separate models approach, the surrogate approach scales to high-dimensional datasets because it requires training just one additional model. And in comparison to other approaches that marginalize out features (uniform, product, marginal), the surrogate approach produces relatively fast explanations because it does not require a sampling-based approximation (i.e., considering multiple values for held-out features) when evaluating the prediction for each feature subset.

3.11.2 MNIST

For the MNIST digit recognition dataset, we trained a 14-layer CNN and generated explanations for the model's loss rather than its classification probabilities. We used zeros as default values, and, as in the previous experiment, we trained a surrogate model to approximate the conditional distribution.

Combining different removal and summary strategies resulted in 25 explanation methods, only two of which corresponded to existing approaches (LossSHAP, and CXPlain without the amortized explainer model). Using these methods, we first generated a grid of explanations for a single example in the dataset (Figure 3.9). More so than in the previous experiment, we now observe significant differences between explanations generated by each method. We make the following qualitative observations about these explanations:

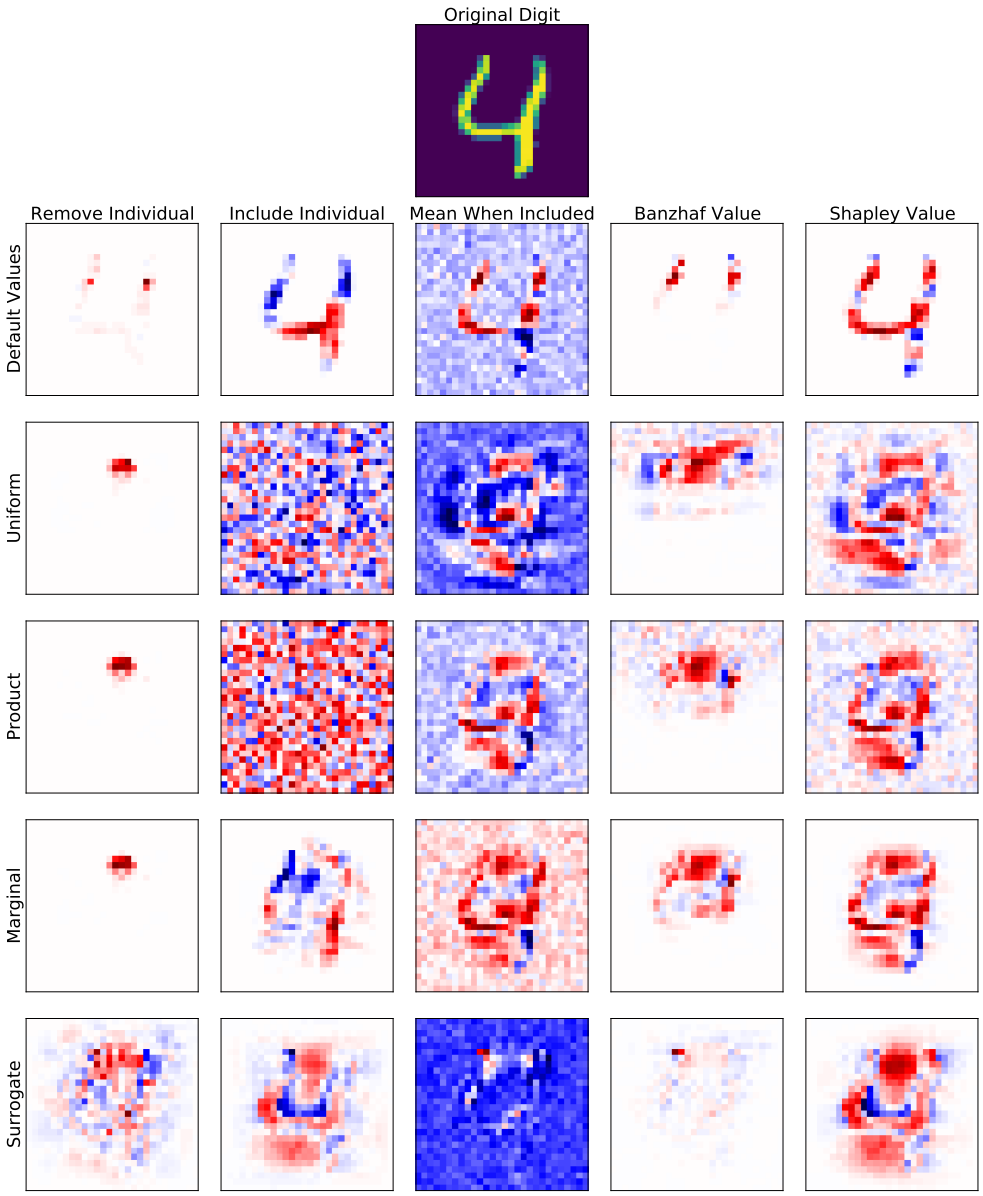


Figure 3.9: Loss explanations for a single MNIST digit. Each heatmap represents attribution values for a different explanation method, with red (blue) pixels improving (hurting) the loss, except for explanations that use the mean when included technique. The vertical axis represents feature removal strategies, and the horizontal axis represents summary techniques.

1. The empty region at the top of the digit should be relevant because it clearly distinguishes fours from nines. This is successfully highlighted by most, but not all methods.

2. Two removal strategies (uniform, product) frequently produce low-quality, noisy explanations. In contrast, the default value explanations are noiseless, but zero pixels always receive zero attribution because removing them does not impact the model; this is not ideal, because zero pixels can be highly informative.

3. The mean when included technique is difficult to visualize because its attributions are not marginal contributions in the game-theoretic sense, where positive (negative) values improve (hurt) the model's loss.

4. When using the surrogate approach, the Shapley value explanation (Figure 3.9 bottom right) is most similar to the one that includes individual features. In contrast, removing individual features has a negligible impact on the model, leading to significant noise artifacts (Figure 3.9 bottom left). This suggests that removing individual features, which is far more common than including individual features (see Table 3.1), can be incompatible with close approximations of the conditional distribution, likely due to strong feature correlations.

Overall, the most visually appealing explanation is the one produced by the surrogate and Shapley value combination (Figure 3.9 bottom right); this method roughly corresponds to LossSHAP (Lundberg et al., 2020), although it uses the surrogate approach as a conditional distribution approximation. For the digit displayed in Figure 3.9, the explanation highlights two regions at the top and bottom that distinguish the four from an eight or a nine; it has minimal noise artifacts; and it indicates that the unusual curvature on the left side of the four may hurt the model's loss, which is highlighted by only a few other explanations. Appendix A.7 shows more LossSHAP explanations on MNIST digits.

To avoid the pitfalls of a purely qualitative analysis, we also evaluate these explanations using quantitative metrics. Many works have proposed techniques for evaluating explanation

methods (Cao et al., 2015; Ancona et al., 2018; Petsiuk et al., 2018; Zhang et al., 2018; Adebayo et al., 2018; Hooker et al., 2019; Lundberg et al., 2020; Jethani et al., 2021a), including sanity checks and measures of correctness. Among these, several consider human-assigned definitions of importance (e.g., the pointing game), while others focus on the model's prediction mechanism. Most of the model-focused metrics involve removing features and measuring their impact on the model's predictions (Ancona et al., 2018; Petsiuk et al., 2018; Hooker et al., 2019; Lundberg et al., 2020; Jethani et al., 2021a), an approach that we explore here.

Evaluation metrics that rely on removing, masking or deleting features have a clear connection with our framework: they mirror the process that removal-based methods follow when generating explanations. With this perspective, we can see that seemingly neutral quantitative metrics may implicitly favor explanations that share their choices of how to remove features from the model. For example, the sensitivity-n metric (Ancona et al., 2018) evaluates the model's predictions when features are replaced with zeros, which favors methods that use this approach when generating explanations (e.g., Occlusion, RISE, CXPlain). Similarly, ROAR (Hooker et al., 2019) measures the decrease in model accuracy after training a new model, which favors methods that account for model retraining.

Rather than relying on a single evaluation metric, we demonstrate how we can create metrics that implicitly favor certain explanations. Following the *insertion* and *deletion* metrics from Petsiuk et al. (2018), we used explanations from each method to produce feature rankings, and we iteratively removed either the most important (deletion) or least important (insertion) features to observe the impact on the model. We evaluated the model's loss for each feature subset, tracing a curve whose area we then measured; we focus on the loss rather than the predictions because our MNIST explanations are based on the model's loss. Critically, we handled removed features using three different strategies: (i) replacing them with zeros, (ii) using a surrogate model (Section 3.9), and (iii) using a new model trained with missing features. (Note that this choice is made separately from the explanation method.)

Based on this approach, Figure 3.10 shows examples of insertion and deletion curves for

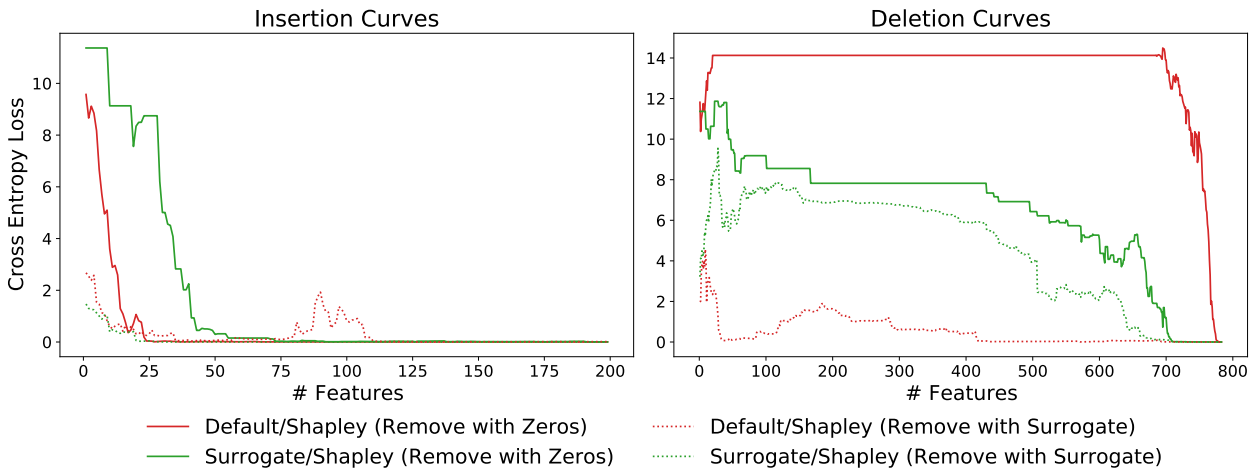


Figure 3.10: Insertion and deletion curves for a single MNIST digit. Curves are displayed for two explanations (Default/Shapley and Surrogate/Shapley) and two feature removal approaches (Zeros, Surrogate). The insertion curves are only shown for up to 200/784 features due to near-zero loss.

a single MNIST digit. We show curves for only two explanation methods—default values and the surrogate approach, both with Shapley values—and we use two masking strategies to illustrate how each one is advantageous for a particular method. When our evaluation replaces missing features with zeros, the explanation that uses default values produces a lower insertion curve and higher deletion curve, both of which represent better performance. When we switch to removing features using the surrogate model, the results are reversed, with the explanation that uses the surrogate model performing significantly better. This experiment demonstrates how evaluation metrics can be aligned with the explanation method.

Next, we performed a similar analysis with 100 MNIST digits: we generated insertion and deletion curves for each digit and measured the mean area under the curve (normalized by the number of features). We considered all combinations of removal strategies and summary techniques, and we generated results separately for the three versions of our insertion/deletion metrics (Table 3.6, Table 3.7 and Table 3.8). The results confirm our previous analysis: replacing missing features with zeros favors the default values approach (Table 3.6), with

Table 3.6: Insertion and deletion scores for MNIST explanations when masking removed features with zeros. Results for the favored removal strategy are italicized, and the best scores are bolded (accounting for 95% confidence intervals).

	Insertion (lower is better)					Deletion (higher is better)				
	RI	II	MWI	BV	SV	RI	II	MWI	BV	SV
Default	*0.683*	*0.478*	***0.131***	***0.176***	***0.134***	*4.797*	***8.356***	***9.508***	*5.636*	***10.286***
Uniform	1.523	1.188	0.427	0.791	0.424	4.121	2.325	4.708	4.105	5.003
Product	1.204	1.017	0.204	0.588	0.224	4.316	2.308	5.428	4.415	5.611
Marginal	1.208	1.038	**0.126**	0.335	**0.125**	4.289	3.726	6.213	4.895	6.797
Surrogate	1.745	0.479	0.516	0.803	0.347	1.486	3.239	2.913	1.943	4.443

Table 3.7: Insertion and deletion scores for MNIST explanations when masking removed features with a surrogate model. Results for the favored removal strategy are italicized, and the best scores are bolded (accounting for 95% confidence intervals).

	Insertion (lower is better)					Deletion (higher is better)				
	RI	II	MWI	BV	SV	RI	II	MWI	BV	SV
Default	0.085	0.152	0.053	0.096	0.100	0.403	0.638	0.325	0.296	0.466
Uniform	0.086	0.080	0.036	0.057	0.038	0.807	0.200	0.836	0.682	0.720
Product	0.068	0.067	0.027	0.046	0.027	0.873	0.179	0.934	0.777	0.888
Marginal	0.067	0.071	0.025	0.033	0.025	0.875	0.189	0.966	1.029	1.451
Surrogate	*0.044*	***0.016***	*0.032*	*0.033*	***0.014***	*0.426*	*1.059*	*1.716*	*0.864*	***3.151***

Table 3.8: Insertion and deletion scores for MNIST explanations when masking removed features using a model trained with missingness. The best scores for each metric are bolded (accounting for 95% confidence intervals).

	Insertion (lower is better)					Deletion (higher is better)				
	RI	II	MWI	BV	SV	RI	II	MWI	BV	SV
Default	0.060	0.105	0.043	0.054	0.054	0.449	0.757	0.477	0.353	0.690
Uniform	0.112	0.074	0.034	0.052	0.034	0.971	0.220	0.893	0.871	0.833
Product	0.079	0.068	0.024	0.044	0.024	1.080	0.191	1.189	0.933	1.060
Marginal	0.079	0.063	0.023	0.035	0.023	1.054	0.181	1.262	1.327	1.781
Surrogate	0.051	**0.011**	0.035	0.037	**0.015**	0.300	1.189	1.418	0.651	**3.728**

the corresponding explanations achieving the best insertion and deletion scores; similarly, handling them using a surrogate model favors the surrogate model approach (Table 3.7). Interestingly, explanations that use the Shapley value produce the best results in both cases; we understand this as a consequence of the metrics checking subsets of all cardinalities, which matches the Shapley value's formulation (Section 3.8).

Evaluating held-out features using a separate model trained with missingness does not explicitly favor any explanation method, because it does not mirror the process by which any explanations are generated. Rather, it can be understood as measuring the information content of the features identified by each method, independently of the original model's prediction mechanism. However, we find that the results in Table 3.8 are very close to those in Table 3.7, with the surrogate approach outperforming the other removal strategies by a significant margin. We understand this as a consequence of the relationship between the surrogate and the model trained with missingness: although the explanation and evaluation metric are based on different models, both models approximate removing features from the Bayes classifier using the conditional distribution (Section 3.9). According to this metric, the method that performs best is also the surrogate model and Shapley value combination, or LossSHAP (Lundberg et al., 2020)—the same method that provided the most visually appealing explanations above.

Measuring an explanation's faithfulness to a model is challenging because it requires making a choice for how to accommodate missing features. As our experiments show, the metrics proposed by recent work implicitly favor methods that align with how the metric is calculated. While using a separate model trained with missingness is less obviously aligned with any explanation method, it still favors methods that approximate the conditional distribution; and furthermore, it measures information content rather than faithfulness to the original model. Users must be wary of the hidden biases in available metrics, and they should rely on evaluations that are most relevant to their intended use-case.

3.11.3 Breast Cancer Subtype Classification

In our final experiment, we analyzed gene microarray data from The Cancer Genome Atlas (TCGA)[12] for breast cancer (BRCA) patients whose tumors were categorized into different molecular subtypes (Berger et al., 2018). Due to the small dataset size (only 510 patients), we prevented overfitting by analyzing a random subset of 100 genes (details in Appendix A.7) and training a regularized logistic regression model. Rather than explaining individual predictions, we explained the dataset loss to determine which genes contain the most information about BRCA subtypes.

We used the same removal and summary strategies as in the previous experiments, including using the mean expression as default values and training a surrogate model to approximate the conditional distribution. Figure 3.11 shows a grid of explanations generated by each method. Three of these explanations correspond to existing approaches (permutation tests, conditional permutation tests, SAGE), but many are new methods. We observe that most explanations identify the same gene as being most important (ESR1), but, besides this, the explanations are difficult to compare qualitatively. In Appendix A.7, we measure the similarity between each of these explanations, finding that there are often similarities across removal strategies and sometimes across summary techniques.

For a quantitative evaluation, we assessed each method by training new models with the top-ranked genes. The best performing method for the top n genes may differ for different values of n, so we trained separate models with the top $n = 1, 2, \ldots, 20$ genes and averaged their performance (Figure 3.12). This is similar to the insertion metric (Petsiuk et al., 2018) used above, except that we measure the dataset loss and focus on subsets with relatively small cardinalities. The results represent each method's usefulness for performing global feature selection.

According to Figure 3.12, the surrogate and Shapley value combination performs best

[12] `https://www.cancer.gov/tcga`

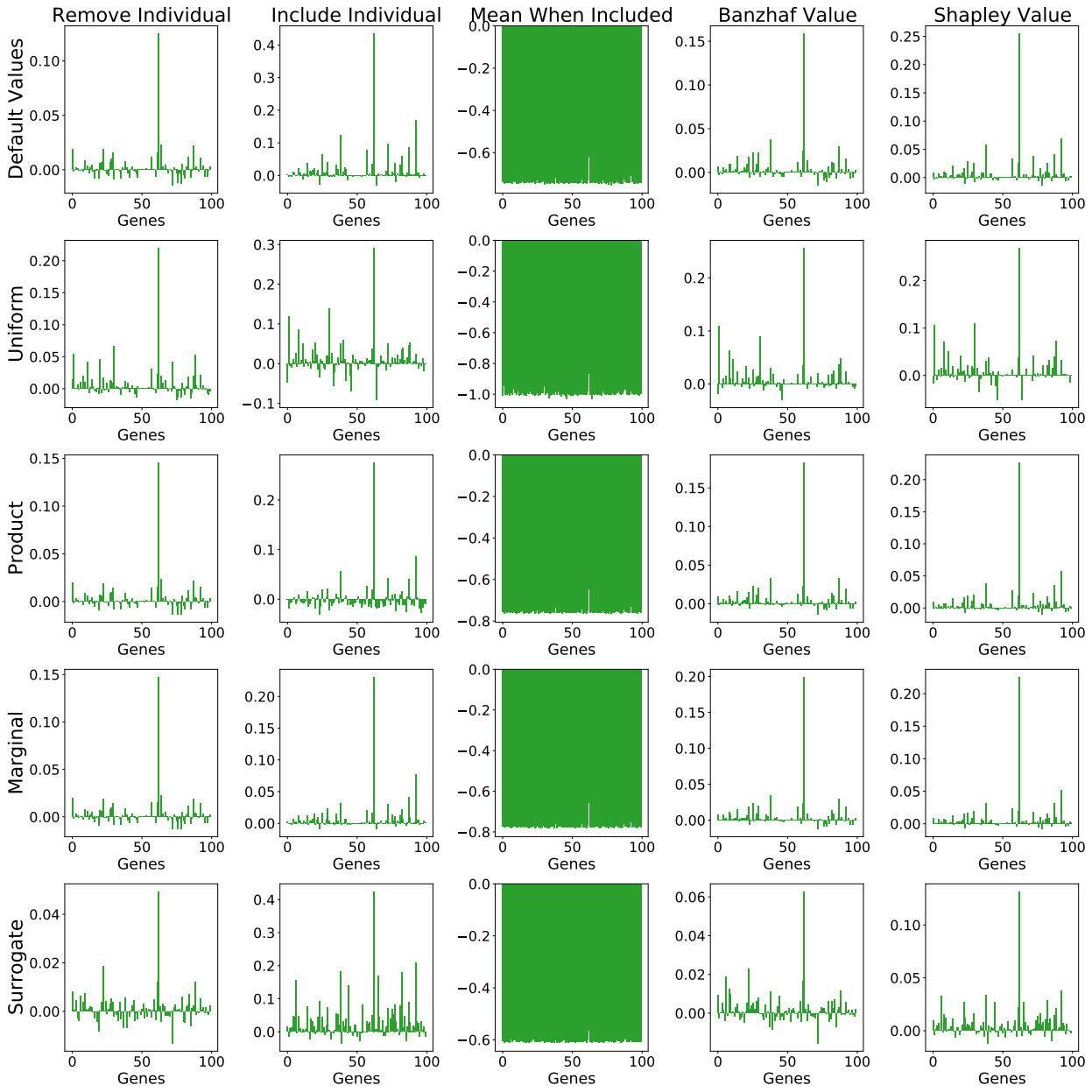


Figure 3.11: Dataset loss explanations for BRCA subtype classification. Each bar chart represents gene attribution values for a different explanation method, with positive (negative) values improving (hurting) the dataset loss, except for the mean when included technique. The vertical axis represents feature removal strategies and the horizontal axis represents summary techniques.

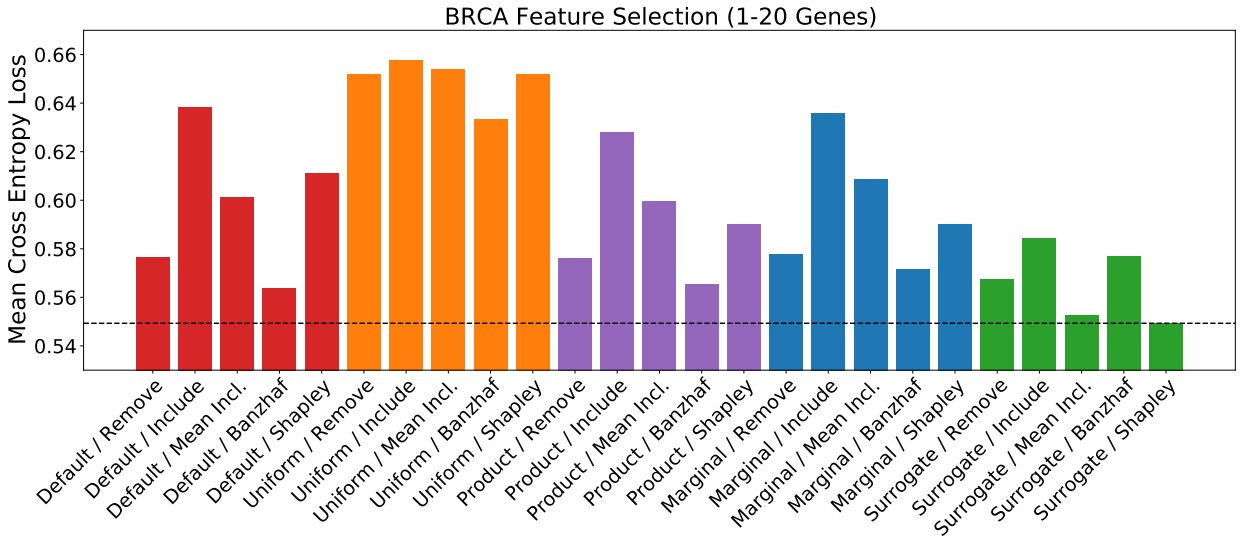


Figure 3.12: Feature selection results for BRCA subtype classification when using top genes identified by each global explanation. Each bar represents the average loss for models trained using $1-20$ top genes (lower is better).

overall; this method roughly corresponds to SAGE (Covert et al., 2020).[13] The mean when included summary performs similarly, and the results are generally better when using the surrogate rather than any other feature removal approach. Including individual features performs poorly, possibly because it neglects feature interactions; and removing features with uniform distributions consistently yields poor results.

We performed a literature review for the most important genes identified by the surrogate and Shapley value combination (SAGE), and we found that many had documented BRCA associations, including ESR1 (Robinson et al., 2013), CCNB2 (Shubbar et al., 2013) and TXNL4B (Nordgard et al., 2008). Other highly ranked genes had known associations with other cancers, including DDC (Koutalellis et al., 2012) and GSS (Kim et al., 2015). We did not evaluate explanation methods based their ability to identify true associations, however,

[13]The original work suggested marginalizing out features with their conditional distribution, but the surrogate model approximation had not been developed at the time of its publication.

due to the ambiguity in verifying an association from the literature. Beyond confirming known relationships, we remark that these global explanations can also be used to generate new scientific hypotheses to be tested in a lab setting.

3.12 Discussion

In this work, we developed a unified framework that encompasses a significant portion of the model explanation field, including 26 existing methods. These methods vary in numerous ways and represent disparate parts of the explainability literature, but our framework systematizes them by showing that each method is specified by three precise mathematical choices:

1. **How the method removes features.** Each method specifies a subset function $F \in \mathfrak{F}$ to make predictions with subsets of features, often based on an existing model $f \in \mathcal{F}$ trained using all the features.

2. **What model behavior the method analyzes.** Each method relies on a set function $u : \mathcal{P}(d) \mapsto \mathbb{R}$ to represent the model's dependence on different groups of features. The set function describes the model's behavior either for an individual prediction (local explanations) or across the entire dataset (global explanations).

3. **How the method summarizes each feature's influence.** Each method generates explanations that provide a concise summary of the features' contributions to the set function $u \in \mathcal{U}$. Mappings of the form $E : \mathcal{U} \mapsto \mathbb{R}^d$ generate feature attribution explanations, and mappings of the form $E : \mathcal{U} \mapsto \mathcal{P}(d)$ generate feature selection explanations.

Our framework reveals that the field is highly interconnected: we find that many state-of-the-art methods are built on the same foundation, are based on different combinations of interchangeable choices, and in many cases differ along only one or two dimensions (recall

Table 3.2). This perspective provides a systematic characterization of the literature and makes it easier to reason about the conceptual and computational advantages of different approaches.

To shed light on the relationships and trade-offs between different methods, we explored perspectives from three related fields that have been overlooked by most explainability research: cooperative game theory (Section 3.8), information theory (Section 3.9) and cognitive psychology (Section 3.10). We found that removal-based explanations are a simple application of *subtractive counterfactual reasoning*, or, equivalently, of Mill's *method of difference*; and we showed that most explanation methods can be interpreted using existing ideas in cooperative game theory and information theory, in many cases leading to a richer understanding of how each method should be interpreted.

Consulting the game theory literature helped us draw connections between several approaches that can be viewed as *probabilistic values* (in the game-theoretic sense), and which are equivalent to fitting a weighted additive model to a cooperative game (Ribeiro et al., 2016). We also found that many feature selection methods can be understood in terms of coalitional excess, and that these approaches are generalized by the optimization problem solved in the Masking Model approach (Dabkowski and Gal, 2017). These game-theoretic connections allow us to compare the various summarization techniques through the properties they satisfy (e.g., the Shapley axioms), as well as through their ease of interpretation for end-users (Section 3.10).

Because of its axiomatic derivation and its many desirable properties, the Shapley value provides, in our view, the most complete summary of how a model works. However, while several approaches provide fast Shapley value approximations (Štrumbelj and Kononenko, 2010; Lundberg and Lee, 2017; Lundberg et al., 2020; Covert and Lee, 2021), these techniques are considerably slower than the fastest removal-based explanations (Section 3.7), particularly in the model-agnostic setting.[14] Furthermore, Shapley value-based explanations

[14]The methods presented in Chapter 6 and Chapter 7 have since helped bridge this gap.

may not always be the best choice: they can be difficult for users to interpret due to their complexity, and variations of the Shapley value may be required to reflect causal relationships in the data (Frye et al., 2020; Heskes et al., 2020; Wang et al., 2021).

Building on work that discusses probabilistic and information-theoretic explanations (Owen, 2014; Chen et al., 2018a; Covert et al., 2020), we found that multiple model behaviors can be understood as information-theoretic quantities. These connections require that removed features are marginalized out using their conditional distribution, and although this approach is challenging to implement, we showed that several removal strategies provide high-quality conditional distribution approximations (e.g., generative modeling approaches and models trained with missing features). Our work is among a growing number of papers lending support to this approach (Strobl et al., 2008; Zintgraf et al., 2017; Aas et al., 2019; Agarwal and Nguyen, 2020; Slack et al., 2020; Covert et al., 2020; Frye et al., 2021), but we present a novel perspective on why this choice is justified: besides providing information-theoretic explanations, we find that marginalizing out features with their conditional distribution is the only approach that yields predictions that satisfy standard probability axioms (Section 3.9).

We recognize, however, that users do not always require information-theoretic explanations. These explanations have the potentially undesirable property that features can be deemed important even if they are not used by the model in a functional sense; this property is useful in certain applications (e.g., bias detection), but in other cases it may be confusing to users. Another concern with this approach is spreading credit among correlated features (Merrick and Taly, 2019; Kumar et al., 2020), because marginalizing out features with their conditional distribution guarantees that perfectly correlated features receive equal attributions (Covert et al., 2020). This property can be avoided by using a different feature removal approach (Janzing et al., 2020), but recent work also argues for sparsifying attributions using a variant of the Shapley value (Frye et al., 2020); interestingly, this is a summary technique (third dimension of our framework) that helps resolve an issue that arises from the feature removal strategy (first dimension of our framework).

The growing interest in black-box ML models has spurred a remarkable amount of model explanation research, and the past decade has yielded numerous publications proposing innovative new methods. However, as the field has matured we have also seen a growing number of unification theories that reveal underlying similarities and implicit relationships among state-of-the-art methods (Lundberg and Lee, 2017; Ancona et al., 2018; Covert et al., 2020). Our framework for removal-based explanations is the broadest unifying theory yet, and it bridges the gap between parts of the explainability literature that may have otherwise been viewed as disjoint. We believe that this work represents an important step towards making the field more organized, rigorous, and easier to navigate.

An improved understanding of the field presents new opportunities for both explainability practitioners and researchers. For practitioners, our framework enables more explicit reasoning about the trade-offs between available explanation tools. The unique advantages of different methods are difficult to understand when each approach is viewed as a monolithic algorithm, but disentangling their choices makes it simpler to reason about their strengths and weaknesses, and potentially develop hybrid methods (as in Section 3.11).

For researchers, our framework offers a new theoretical perspective that can guide and strengthen ongoing research. Using the tools we developed, future work will be better equipped to (i) specify the dimensions along which new methods differ from existing ones, (ii) distinguish the implicit objectives of new approaches from their approximations, (iii) resolve shortcomings in existing explanation methods using solutions along different dimensions of the framework, and (iv) rigorously justify new approaches in light of their connections with cooperative game theory, information theory and psychology. As the number of removal-based explanations continues to grow, we hope that our framework will serve as a strong foundation upon which future research can build.

Chapter 4

ROBUSTNESS OF REMOVAL-BASED EXPLANATIONS

4.1 Chapter Notes

This chapter presents joint work with Chris Lin (co-first author) and Su-In Lee that is currently a pre-print. The main contribution of this work is to build on the framework from Chapter 3 to provide a unified analysis of the robustness properties of many existing model explanation methods.

4.2 Introduction

In recent years, ML has shown great promise in a variety of real-world applications. An obstacle to its widespread deployment is the lack of transparency, an issue that has prompted a wave of research on interpretable or explainable machine learning (Simonyan et al., 2013; Zeiler and Fergus, 2014; Ribeiro et al., 2016; Sundararajan et al., 2017; Lundberg and Lee, 2017; Petsiuk et al., 2018). One popular way of explaining a machine learning model is feature attribution, which assigns importance scores to input features of the model (Simonyan et al., 2013; Ribeiro et al., 2016; Sundararajan et al., 2017; Lundberg and Lee, 2017; Petsiuk et al., 2018). Many feature attribution methods can be categorized as either *gradient-based* methods that compute gradients of model predictions with respect to input features (Ancona et al., 2018), or *removal-based* methods that remove features to quantify each feature's influence (Covert et al., 2021). While substantial progress has been made and current tools are widely used for model debugging and scientific discovery (Bhatt et al., 2020; DeGrave et al., 2021; Janizek et al., 2021; Novakovsky et al., 2022), some important concerns remain. Among them is *unclear robustness*: feature attributions are vulnerable to adversarial attacks, and even without an adversary appear unstable under small changes to the input or model.

As a motivating example, consider the work of Ghorbani et al. (2019), which shows that minor perturbations to an image can lead to substantially different feature attributions while preserving the original prediction. Similar to adversarial examples designed to alter model predictions (Szegedy et al., 2013; Goodfellow et al., 2014; Kurakin et al., 2018), this phenomenon violates the intuition that explanations should be invariant to imperceptible changes. A natural question, and the focus of this chapter, is therefore: *When can we guarantee that feature attributions are robust to small changes in the input or small changes in the model?* In answering this question, we find that previous theoretical work on this topic has primarily focused on gradient-based methods. Hence, in this work we provide a unified analysis to characterize the robustness of removal-based feature attributions under both input and model perturbation, considering a range of existing methods in combination with different approaches to simulate feature removal.

Related work Recent work has demonstrated that gradient-based attributions for neural networks are susceptible to small input perturbations that can lead to markedly different attributions (Dombrowski et al., 2019; Ghorbani et al., 2019; Kindermans et al., 2019). Theoretical analyses have established that this issue relates to model smoothness, and approaches like stochastic smoothing of gradients, weight decay regularization, smoothing activation functions, and Hessian minimization have been shown to generate more robust gradient-based attributions (Dombrowski et al., 2019, 2022; Wang et al., 2020b; Agarwal et al., 2021). As for the robustness of removal-based attributions under input perturbations, two works investigated the Lipschitz continuity properties of methods like RISE, LIME, SHAP and leave-one-out: Alvarez-Melis and Jaakkola (2018) take an empirical approach to study local Lipschitz continuity, and Khan et al. (2022) take a theoretical approach to characterize global Lipschitz continuity. Agarwal et al. (2022) study the robustness of explanation methods for graph neural networks, which output discrete node or edge masks rather than real-valued attributions. Our work on input perturbation is most similar to Khan et al. (2022), but we generalize the analysis along multiple dimensions by considering alternative feature removal

and summarization techniques (e.g., LIME's weighted least squares approach Ribeiro et al. 2016).

Other works have considered model perturbations and manipulated neural networks to produce targeted, arbitrary gradient-based attributions (Anders et al., 2020; Heo et al., 2019). Anders et al. (2020) explain this phenomenon through the insight that neural network gradients are underdetermined with many degrees of freedom to exploit, and they demonstrate better robustness when gradient-based attributions are projected onto the data manifold. As for the robustness of removal-based attributions under model perturbation, models can be modified to hide their reliance on sensitive features in LIME and SHAP (Slack et al., 2020; Dimanov et al., 2020); however, Frye et al. (2021) show that such hidden features can be discovered if features are removed using their conditional distribution, suggesting a key role for the feature removal approach in determining sensitivity to such attacks. Our work formalizes these results through the lens of robustness under model perturbation, and provides theoretical guarantees.

Contribution The main contribution of this work is to develop a comprehensive characterization of the robustness properties of removal-based feature attributions, focusing on two notions of robustness that have received attention in the literature: (i) stability under input changes, and (ii) stability under model changes. Our specific contributions are the following:

1. We provide theoretical guarantees for the robustness of model predictions with the removal of arbitrary feature subsets, under both input and model perturbations.

2. We analyze the robustness of techniques for summarizing each feature's influence (e.g., Shapley values) and derive best- and worst-case robustness within several classes of such techniques.

3. We combine the analyses above to prove unified attribution robustness properties to both input and model perturbations, where changes in removal-based attributions are controlled by the scale of perturbations in either input space or function space.

4. We validate our theoretical results and demonstrate practical implications using both synthetic and real-world datasets.

4.3 Background

Here, we review removal-based feature attributions and then introduce our problem formulation.

4.3.1 Removal-Based Feature Attributions

We focus here on an existing model f whose predictions we seek to explain. We consider that the model's output space is one-dimensional, or $f : \mathbb{R}^d \mapsto \mathbb{R}$, which is not restrictive because attributions are typically calculated for scalar predictions (e.g., the probability for a given class). We consider that the input variable $\boldsymbol{x}$ consists of d separate features, or $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$, and we assume that all explanations are generated for inputs that lie on the data manifold $\mathcal{X} \subseteq \mathbb{R}^d$, or where $p(x) > 0$.

Our work focuses on a class of methods known as *removal-based explanations*; see Chapter 3 for a detailed discussion. Intuitively, these are algorithms that remove subsets of inputs and summarize how each feature affects the model. This framework describes a large number of methods that are distinguished by two main implementation choices: (i) how feature information is removed from the model, and (ii) how the algorithm summarizes each feature's influence.[1] This perspective shows a close connection between methods like leave-one-out (Zeiler and Fergus, 2014), LIME (Ribeiro et al., 2016) and Shapley values (Lundberg and Lee, 2017), and as we show here, enables a unified analysis of their robustness properties. Below, we describe the two main implementation choices in detail.

Feature removal Most machine learning models require all feature values to generate predictions, so we must specify a convention for depriving the model of feature information.

[1]The framework also considers the choice to explain different *model behaviors* (e.g., the loss calculated over one or more examples), but we focus on methods that explain individual predictions.

We denote the prediction given partial inputs by $f(x_S)$. Many implementations have been discussed in the literature (Covert et al., 2021), but we focus here on three common choices. Given a set of observed values x_S, these techniques can all be viewed as averaging the prediction over a distribution $q(\boldsymbol{x}_{\bar{S}})$ for the held-out feature values:

$$f(x_S) \equiv \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}})} [f(x_S, \boldsymbol{x}_{\bar{S}})] = \int f(x_S, x_{\bar{S}}) q(x_{\bar{S}}) dx_{\bar{S}}. \tag{4.1}$$

Specifically, the three choices we consider are:

- (Baseline values) Given a baseline input $b \in \mathbb{R}^d$, we can set the held-out features to their corresponding values $b_{\bar{S}}$ (Sundararajan and Najmi, 2020). This is equivalent to letting $q(\boldsymbol{x}_{\bar{S}})$ be a Dirac delta centered at $b_{\bar{S}}$.

- (Marginal distribution) Rather than using a single replacement value, we can average across values sampled from the input's marginal distribution, or let $q(\boldsymbol{x}_{\bar{S}}) = p(\boldsymbol{x}_{\bar{S}})$ (Lundberg and Lee, 2017).

- (Conditional distribution) Finally, we can average across replacement values while conditioning on the available features, which is equivalent to setting $q(\boldsymbol{x}_{\bar{S}}) = p(\boldsymbol{x}_{\bar{S}} \mid x_S)$ (Frye et al., 2021).

The first two options are commonly implemented in practice because they are simplest to estimate, but the third choice is viewed by some work as more informative to users (Aas et al., 2021; Frye et al., 2021; Covert et al., 2021). The conditional distribution approach requires more complex and error-prone estimation procedures (see Chen et al. 2022 for a discussion), but we assume here that all three versions of $f(x_S)$ can be calculated exactly.

Summary technique Given a feature removal technique that allows us to query the model with arbitrary feature sets, we must define a convention for summarizing each feature's influence. This is challenging due to the exponential number of feature sets, or because

$|\mathcal{P}(d)| = 2^d$. Again, many techniques have been discussed in the literature (Covert et al., 2021), with the simplest option being comparing the prediction with all features included and with a single feature missing (Zeiler and Fergus, 2014). We take a broad perspective here, considering a range of approaches that define attributions as a linear combination of predictions with different feature sets. These methods include leave-one-out (Zeiler and Fergus, 2014), RISE (Petsiuk et al., 2018), LIME (Ribeiro et al., 2016), Shapley values (Lundberg and Lee, 2017), and Banzhaf values (Chen and Jordan, 2020), among other possible options.

All of these methods yield per-feature attribution scores $\phi_i(f, x) \in \mathbb{R}$ for $i \in [d]$. For example, the Shapley value calculates feature attribution scores as follows (Lundberg and Lee, 2017):

$$\phi_i(f, x) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \left(f(x_{S \cup i}) - f(x_S) \right). \tag{4.2}$$

The specific linear combinations for the other approaches we consider are shown in Table 4.3. The remainder of the chapter uses the notation $f(x_S)$ to refer to predictions with partial information, and $\phi(f, x) = [\phi_1(f, x), \ldots, \phi_d(f, x)] \in \mathbb{R}^d$ to refer to feature attributions. We do not introduce separate notation to distinguish between implementation choices, opting instead to make the relevant choices clear in each result.

4.3.2 Problem Formulation

The problem formulation here is straightforward: our goal is to understand the stability of feature attributions under input perturbations and model perturbations. Formally, we aim to study

1. whether $\|\phi(f, x) - \phi(f, x')\|$ is controlled by $\|x - x'\|$ (**input perturbation**), and

2. whether $\|\phi(f, x) - \phi(f', x)\|$ is controlled by $\|f - f'\|$ (**model perturbation**).

The following sections show how this can be guaranteed using certain distance metrics, and under certain assumptions about the model and/or the data distribution.

4.4 Preliminary Results

As an intermediate step towards understanding explanation robustness, we first provide results for sub-components of the algorithms. We begin by addressing the robustness of predictions with partial information to input and model perturbations, and we then discuss robustness properties of the summary technique. Proofs for all results are in Appendix B.

Before proceeding, we introduce two assumptions that are necessary for our analysis.

Assumption 4.1. *We assume that the model f is globally L-Lipschitz continuous, or that we have $|f(x) - f(x')| \leq L \cdot \|x - x'\|_2$ for all $x, x' \in \mathbb{R}^d$.*

Assumption 4.2. *We assume that the model f has bounded predictions, or that there exists a constant B such that $|f(x)| \leq B$ for all $x \in \mathbb{R}^d$.*

The first assumption holds for most deep learning architectures, because they typically compose a series of Lipschitz continuous layers (Virmaux and Scaman, 2018). The second assumption holds with $B = 1$ for classification models. These are therefore mild assumptions, but they are discussed further in Section 4.7.

We also define the notion of a functional norm, which is useful for several of our results.

Definition 4.1. *The L^p norm for a function $g : \mathbb{R}^d \mapsto \mathbb{R}$ is defined as $\|g\|_p \equiv (\int |g(x)|^p dx)^{1/p}$, where the integral is taken over $\mathbb{R}^d$. $\|g\|_{p,\mathcal{X}'}$ denotes the same integral taken over the domain $\mathcal{X}' \subseteq \mathbb{R}^d$.*

Our results adopt this to define a notion of distance between functions $\|g - g'\|_p$, focusing on the cases where $p = 1$ for the L^1 distance and $p = \infty$ for the Chebyshev distance.

4.4.1 Prediction Robustness to Input Perturbations

Our goal here is to understand how the prediction function $f(\boldsymbol{x}_S)$ for a feature set $\boldsymbol{x}_S$ behaves under small input perturbations. We first consider the case where features are removed using the baseline or marginal approach, and we find that Lipschitz continuity is inherited from the original model.

Lemma 4.1. *When removing features using either the* ***baseline*** *or* ***marginal*** *approaches, the prediction function $f(\boldsymbol{x}_S)$ for any feature set $\boldsymbol{x}_S$ is L-Lipschitz continuous:*

$$|f(x_S) - f(x'_S)| \leq L \cdot \|x_S - x'_S\|_2 \quad \forall\, x_S, x'_S \in \mathbb{R}^{|S|}.$$

Next, we consider the case where features are removed using the conditional distribution approach. We show that the continuity of $f(\boldsymbol{x}_S)$ depends not only on the original model, but also on the similarity of the conditional distribution for the held-out features.

Lemma 4.2. *When removing features using the* ***conditional*** *approach, the prediction function $f(\boldsymbol{x}_S)$ for a feature set $\boldsymbol{x}_S$ satisfies*

$$|f(x_S) - f(x'_S)| \leq L \cdot \|x_S - x'_S\|_2 + 2B \cdot d_{TV}\Big(p(\boldsymbol{x}_{\bar{S}} \mid x_S), p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big),$$

where the total variation distance is defined via the L^1 functional distance as

$$d_{TV}\Big(p(\boldsymbol{x}_{\bar{S}} \mid x_S), p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big) \equiv \frac{1}{2}\Big\|p(\boldsymbol{x}_{\bar{S}} \mid x_S) - p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big\|_1.$$

Lemma 4.2 does not immediately imply Lipschitz continuity for $f(\boldsymbol{x}_S)$, because we have not bounded the total variation distance in terms of $\|x_S - x'_S\|_2$. To address this, we require the following Lipschitz-like continuity property in the total variation distance.

Assumption 4.3. *We assume that there exists a constant M such that for all $S \subseteq [d]$, we have*

$$d_{TV}\Big(p(\boldsymbol{x}_{\bar{S}} \mid x_S), p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big) \leq M \cdot \|x_S - x'_S\|_2 \quad \forall x_S, x'_S \in \mathbb{R}^{|S|}.$$

Intuitively, this says that the conditional distribution $p(\boldsymbol{x}_{\bar{S}} \mid x_S)$ cannot change too quickly as a function of the observed features. This property does not hold for all data distributions $p(\boldsymbol{x})$, and it may hold in some cases only for large M; in such scenarios, the function $f(\boldsymbol{x}_S)$ is not guaranteed to change slowly. However, there are cases where it holds.

For example, we have $M = 0$ if the features are independent, and the following example shows a case where it holds with dependent features.

Example 4.1. *For a Gaussian random variable $\boldsymbol{x} \sim \mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d\times d}$, Assumption 4.3 holds with $M = \frac{1}{2}\sqrt{\lambda_{\max}(\Sigma^{-1}) - \lambda_{\min}(\Sigma^{-1})}$. If $\boldsymbol{x}$ is assumed to be standardized, this captures the case where independent features yield $M = 0$. Intuitively, it also means that if one dimension is roughly a linear combination of the others, or if $\lambda_{\max}(\Sigma^{-1})$ is large, the conditional distribution can change quickly as a function of the conditioning variables.*

Now, assuming that this property holds for $p(\boldsymbol{x})$, we show that we can improve upon Lemma 4.2.

Lemma 4.3. *Under Assumption 4.3, the prediction $f(\boldsymbol{x}_S)$ defined using the **conditional** approach is Lipschitz continuous with constant $L + 2BM$ for any feature set $\boldsymbol{x}_S$.*

Between Lemmas 4.1 and 4.3, we have established that the function $f(\boldsymbol{x}_S)$ remains Lipschitz continuous for any feature set $\boldsymbol{x}_S$, although in some cases with a larger constant that depends on the data distribution. These results are summarized in Table 4.1. In Appendix B.1 we show that our analysis can be extended to account for sampling in eq. 4.1 when the expectation is not calculated exactly.

Table 4.1: Lipschitz constants induced by each feature removal technique.

Baseline	Marginal	Conditional
L	L	$L + 2BM$

Table 4.2: Relevant functional distances for each feature removal technique.

Baseline	Marginal	Conditional
$\|f - f'\|_\infty$	$\|f - f'\|_\infty$	$\|f - f'\|_{\infty,\mathcal{X}}$

4.4.2 Prediction Robustness to Model Perturbations

Our next goal is to understand how the function $f(\boldsymbol{x}_S)$ for a feature set $\boldsymbol{x}_S$ behaves under small changes to the model. The intuition is that if two models make very similar predictions

with all features, they should continue to make similar predictions with a subset of features. We first derive a general result involving the proximity between two models f and f' within a subdomain.

Lemma 4.4. *For two models $f, f' : \mathbb{R}^d \mapsto \mathbb{R}$ and a subdomain $\mathcal{X}' \subseteq \mathbb{R}^d$, the prediction functions $f(\boldsymbol{x}_S), f'(\boldsymbol{x}_S)$ for any feature set x_S satisfy*

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|_{\infty,\mathcal{X}'} \cdot Q_{x_S}(\mathcal{X}') + 2B \cdot \big(1 - Q_{x_S}(\mathcal{X}')\big),$$

where $Q_{x_S}(\mathcal{X}')$ is the probability of imputed samples lying in $\mathcal{X}'$ based on the distribution $q(\boldsymbol{x}_{\bar{S}})$:

$$Q_{x_S}(\mathcal{X}') \equiv \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}})} \left[\mathbb{I}\left\{(x_S, \boldsymbol{x}_{\bar{S}}) \in \mathcal{X}'\right\}\right].$$

The difference in predictions therefore depends on the distance between the models only within the subdomain $\mathcal{X}'$, as well as the likelihood of imputed sampled lying in this subdomain. This result illustrates a point shown by Slack et al. (2020): that two models which are equivalent on a small subdomain, or even on the entire data manifold $\mathcal{X}$, can lead to different attributions if we use a removal technique that yields a low value for $Q_{x_S}(\mathcal{X}')$. In fact, when $Q_{x_S}(\mathcal{X}') \to 0$, Lemma 4.4 reduces to a trivial bound $|f(x_S) - f'(x_S)| \leq 2B$ that follows directly from Assumption 4.2.

The general result in Lemma 4.4 allows us to show two simpler ones. In both cases, we choose the subdomain $\mathcal{X}'$ and removal technique $q(\boldsymbol{x}_{\bar{S}})$ so that $Q_{x_S}(\mathcal{X}') = 1$.

Lemma 4.5. *When removing features using the **conditional** approach, the prediction functions $f(\boldsymbol{x}_S), f'(\boldsymbol{x}_S)$ for two models f, f' and any feature set x_S satisfy*

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|_{\infty,\mathcal{X}}.$$

The next result is similar but involves a potentially larger upper bound $\|f - f'\|_\infty \geq \|f - f'\|_{\infty,\mathcal{X}}$.

Lemma 4.6. *When removing features using the* ***baseline*** *or* ***marginal*** *approach, the prediction functions $f(\boldsymbol{x}_S), f'(\boldsymbol{x}_S)$ for two models f, f' and any feature set x_S satisfy*

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|_\infty.$$

Remark 1. Lemma 4.6 implies that if two models f, f' are functionally equivalent, or $f(x) = f(x')$ for all $x \in \mathbb{R}^d$, they remain equivalent with any feature subset $\boldsymbol{x}_S$. In Section 4.5, we will see that this implies equal attributions for the two models—a natural property that, perhaps surprisingly, is not satisfied by all feature attribution methods (Shrikumar et al., 2017; Montavon et al., 2019), and that has been described in the literature as an *implementation invariance* property (Sundararajan et al., 2017). This holds automatically for all removal-based methods.

Remark 2. In contrast, Lemma 4.5 implies the same for models that are equivalent *only on the data manifold* $\mathcal{X} \subseteq \mathbb{R}^d$. This is a less stringent requirement to guarantee equal attributions, and it is perhaps reasonable that models with equal predictions for all realistic inputs receive equal attributions. To emphasize this, we propose distinguishing between a notion of *weak implementation invariance* and *strong implementation invariance*, depending on whether equal attributions are guaranteed when we have $f(x) = f'(x)$ everywhere ($x \in \mathbb{R}^d$) or almost everywhere ($x \in \mathcal{X}$).

4.4.3 Summary Technique Robustness

We previously focused on the effects of the feature removal choice, so our goal is now to understand the role of the summary technique. Given a removal approach that lets us query the model with any feature set, the summary generates attribution scores $\phi(f, x) \in \mathbb{R}^d$, often using a linear combination of the outputs $f(x_S)$ for each $S \subseteq [d]$. We formalize this in the following proposition.

Proposition 4.1. *The attributions for each method can be calculated by applying a linear*

operator $A \in \mathbb{R}^{d \times 2^d}$ *to a vector* $v \in \mathbb{R}^{2^d}$ *representing the predictions with each feature set, or*

$$\phi(f, x) = Av,$$

where the linear operator A *for each method is listed in Table 4.3, and* v *is defined as* $v_S = f(x_S)$ *for each* $S \subseteq [d]$ *based on the chosen feature removal technique.*

In this notation, the entries of v are indexed as v_S for $S \subseteq [d]$, and the entries of A are indexed as A_{iS} for $i \in [d]$ and $S \subseteq [d]$. The operation Av can be understood as summing across all subsets, or $(Av)_i = \sum_S A_{iS} \cdot v_S$. Representing the attributions this way is convenient for our next results.

The entries for the linear operator in Table 4.3 are straightforward for the first four methods, but LIME depends on several implementation choices: these include the choice of weighting kernel, regularization, and intercept term. The first three methods are in fact special cases of LIME (Covert et al., 2021). The class of weighting kernel used in practice is difficult to characterize analytically, but its default parameters for tabular, image and text data approach limiting cases where LIME reduces to other methods (Appendix B.7). For simplicity, the remainder of this section focuses on the other methods.

Perturbing either the input or model induces a change in $v \in \mathbb{R}^{2^d}$, and we must consider how the attributions differ between the original vector v and an alternative v'. Proposition 4.1 suggests that we can bound the distance between attributions via the change in model outputs $\|v - v'\|$ and properties of the matrix A. We can even do this under different distance metrics, as we show in the next result.

Lemma 4.7. *The difference in attributions given the same summary technique* A *and different model outputs* v, v' *can be bounded as*

$$\|Av - Av'\|_2 \leq \|A\|_2 \cdot \|v - v'\|_2 \qquad \textit{or} \qquad \|Av - Av'\|_2 \leq \|A\|_{1,\infty} \cdot \|v - v'\|_\infty,$$

where $\|A\|_2$ *is the spectral norm, and the operator norm* $\|A\|_{1,\infty}$ *is the square root of the sum*

Table 4.3: Summary technique linear operators used by various removal-based explanations.

Summary	Method	A_{iS} $(i \in S)$	A_{iS} $(i \notin S)$	$\|A\|_{1,\infty}$	$\|A\|_2$
Leave-one-out	Occlusion (Zeiler and Fergus, 2014)	$\mathbb{I}\{\|S\| = d\}$	$-\mathbb{I}\{\|S\| = d-1\}$	$2\sqrt{d}$	$\sqrt{d+1}$
Shapley value	SHAP (Lundberg and Lee, 2017)	$\frac{(\|S\|-1)!(d-\|S\|)!}{d!}$	$-\frac{\|S\|!(d-\|S\|-1)!}{d!}$	$2\sqrt{d}$	$\sqrt{2/d}$
Banzhaf value	Banzhaf (Chen and Jordan, 2020)	$1/2^{d-1}$	$-1/2^{d-1}$	$2\sqrt{d}$	$1/2^{d/2-1}$
Mean when included	RISE (Petsiuk et al., 2018)	$1/2^{d-1}$	0	$\sqrt{d}$	$\sqrt{(d+1)/2^d}$
Weighted least squares	LIME (Ribeiro et al., 2016)	Depends on implementation choices (see Appendix B.3)			

of squared row 1-norms, with values for each A given in Table 4.3.

For both inequalities, smaller norms $\|A\|$ represent stronger robustness to perturbations. Neither bound is necessarily tighter, and the choice of which to use depends on how $v - v'$ is bounded. The first bound using the Euclidean distance $\|v - v'\|_2$ is perhaps more intuitive, but the second bound is more useful here because the results in the analysis above effectively relate to $\|v - v'\|_\infty$.

This view of the summary technique's robustness raises a natural question, which is whether the approaches used in practice have relatively good or bad robustness (Table 4.3). The answer is not immediately clear, because within the class of linear operators we can control the robustness arbitrarily: we can achieve $\|A\|_2 = \|A\|_{1,\infty} = 0$ by setting $A = 0$ (strong robustness), and we can likewise get $\|A\|_2, \|A\|_{1,\infty} \to \infty$ by setting entries of A to a large value (weak robustness). These results are not useful, however, because they come at the expense of generating meaningful attributions.

Rather than considering robustness within the class of *all* linear operators $A \in \mathbb{R}^{d \times 2^d}$, we restrict our attention to attributions that are in some sense meaningful. There are multiple ways to define this, but we consider solutions that satisfy one or more of the following properties, which are motivated by axioms from the literature on game-theoretic credit allocation (Shapley, 1953; Monderer et al., 2002).

Definition 4.2. *We define the following properties for linear operators $A \in \mathbb{R}^{d \times 2^d}$ that are used for removal-based explanations:*

- *(Boundedness) For all $v \in \mathbb{R}^{2^d}$, each feature's attribution is bounded by its smallest and largest marginal contributions, or $\min_{S \subseteq [d] \setminus i}(v_{S \cup i} - v_S) \leq (Av)_i \leq \max_{S \subseteq [d] \setminus i}(v_{S \cup i} - v_S)$ for all $i \in [d]$.*

- *(Symmetry) For all $v \in \mathbb{R}^{2^d}$, two features that make equal marginal contributions to all feature sets must have equal attributions, or $(Av)_i = (Av)_j$ if $v_{S \cup i} = v_{S \cup \{j\}}$ for all $S \subseteq [d]$.*

- *(Efficiency) The attributions sum to the difference in predictions for the complete and empty sets, or $\mathbf{1}^\top Av = v_{[d]} - v_\varnothing$ for all $v \in \mathbb{R}^{2^d}$.*

Among these properties, we prioritize boundedness because this leads to a class of solutions that are well-known in game theory, and which are called *probabilistic values* (Monderer et al., 2002). We now show that constraining the summary technique to satisfy different combinations of these properties yields clear upper and lower bounds for the robustness. We first address the operator norm $\|A\|_{1,\infty}$, which is the simpler case.

Lemma 4.8. *When the linear operator A satisfies the **boundedness** property, we have $\|A\|_{1,\infty} = 2\sqrt{d}$.*

The above result applies to several summary techniques shown in Table 4.3, including leave-one-out (Zeiler and Fergus, 2014), Shapley values (Lundberg and Lee, 2017) and Banzhaf values (Chen and Jordan, 2020). We now address the case of the spectral norm $\|A\|_2$, beginning with the case where boundedness and symmetry are satisfied.

Lemma 4.9. *When the linear operator A satisfies the **boundedness** and **symmetry** properties, the spectral norm is bounded as follows,*

$$\frac{1}{2^{d/2-1}} \leq \|A\|_2 \leq \sqrt{d+1},$$

with $1/2^{d/2-1}$ achieved by the Banzhaf value and $\sqrt{d+1}$ achieved by leave-one-out.

This shows that the Banzhaf value is the most robust summary within a class of linear operators, which are known as *semivalues* (Monderer et al., 2002). Its robustness is even better than the Shapley value, a result that has been discussed in recent work by Wang and Jia (2022). However, the Banzhaf value is criticized for failing to satisfy the efficiency property (Dubey and Shapley, 1979), so we now address the case where this property holds.

Lemma 4.10. *When the linear operator A satisfies the* ***boundedness*** *and* ***efficiency*** *properties, the spectral norm is bounded as follows,*

$$\sqrt{\frac{2}{d}} \leq \|A\|_2 \leq \sqrt{2 + 2 \cdot \cos\left(\frac{\pi}{d+1}\right)},$$

with $\sqrt{2/d}$ achieved by the Shapley value.

We therefore observe that the Shapley value is the most robust choice within a different class of linear operators, which are known as *random-order values* (Monderer et al., 2002). In comparison, the worst-case robustness is achieved by a method that has not been discussed in the literature, but which is a special case of a previously proposed method (Frye et al., 2020). The upper bound in Lemma 4.10 is approximately equal to 2, whereas the upper bound in Lemma 4.9 grows with the input dimensionality, suggesting that the efficiency property imposes a minimum level of robustness but limits robustness in the best case.

Finally, we can also trivially say that $\|A\|_2 = \sqrt{2/d}$ when boundedness, symmetry and efficiency are all satisfied, because the Shapley value is the only linear operator to satisfy all three properties (Shapley, 1953; Monderer et al., 2002).

Lemma 4.11. *When the linear operator A satisfies the* ***boundedness***, ***symmetry*** *and* ***efficiency*** *properties, the spectral norm is $\|A\|_2 = \sqrt{2/d}$.*

4.5 Main Results: Feature Attribution Robustness

We now shift our attention to the robustness properties of entire feature attribution algorithms. These results build on those presented in Section 4.4, combining them to provide

general results that apply both to existing methods and to new methods that combine their implementation choices arbitrarily.

Our first main result relates to the robustness to input perturbations.

Theorem 4.1. *The robustness of removal-based explanations to input perturbations is given by the following meta-formula,*

$$\|\phi(f,x) - \phi(f,x')\|_2 \leq g(\text{removal}) \cdot h(\text{summary}) \cdot \|x - x'\|_2,$$

where the factors for each method are defined as follows:

$$g(\text{removal}) = \begin{cases} L & \text{if removal} = \textbf{\textit{baseline}} \text{ or } \textbf{\textit{marginal}} \\ L + 2BM & \text{if removal} = \textbf{\textit{conditional}}, \end{cases}$$

$$h(\text{summary}) = \begin{cases} 2\sqrt{d} & \text{if summary} = \textbf{\textit{Shapley}}, \textbf{\textit{Banzhaf}}, \text{ or } \textbf{\textit{leave-one-out}} \\ \sqrt{d} & \text{if summary} = \textbf{\textit{mean when included}}. \end{cases}$$

We therefore observe that the explanation functions themselves are Lipschitz continuous, but with a constant that combines properties of the original model with implementation choices from the attribution method. The model's inherent robustness interacts with the removal choice to yield the $g(\text{removal})$ factor, and the summary technique is accounted for by $h(\text{summary})$.

Our second result takes a similar form, but relates to the robustness to model perturbations.

Theorem 4.2. *The robustness of removal-based explanations to model perturbations is given by the following meta-formula,*

$$\|\phi(f,x) - \phi(f',x)\|_2 \leq h(\text{summary}) \cdot \|f - f'\|,$$

where the functional distance and factor associated with the summary technique are specified

as follows:

$$\|f - f'\| = \begin{cases} \|f - f'\|_\infty & \textit{if removal} = \textbf{\textit{baseline}} \textit{ or } \textbf{\textit{marginal}} \\ \|f - f'\|_{\infty,\mathcal{X}} & \textit{if removal} = \textbf{\textit{conditional}}, \end{cases}$$

$$h(\text{summary}) = \begin{cases} 2\sqrt{d} & \textit{if summary} = \textbf{\textit{Shapley}}, \textbf{\textit{Banzhaf}}, \textit{ or } \textbf{\textit{leave-one-out}} \\ \sqrt{d} & \textit{if summary} = \textbf{\textit{mean when included}}. \end{cases}$$

Similarly, we see here that the attribution difference is determined by the strength of the model perturbation, as measured by a functional distance metric that depends on the feature removal technique. This represents a form of Lipschitz-like continuity with respect to the model f. These results address the case with either an input perturbation or model perturbation, but we can also account for simultaneous input and model perturbations. The following result is implied by Theorems 4.1 and 4.2.

Corollary 4.1. *The robustness of removal-based explanations to simultaneous input and model perturbations is given by the following meta-formula,*

$$\|\phi(f, x) - \phi(f', x')\|_2 \leq g(\text{removal}) \cdot h(\text{summary}) \cdot \|x - x'\|_2 + h(\text{summary}) \cdot \|f - f'\|,$$

where the functional distance and factors associated with the removal and summary technique are given in Theorems 4.1 and 4.2.

The remainder of this chapter empirically verifies these results and discusses their practical implications.

4.6 Experiments

This section conducts a range of experiments to provide empirical validation for our theoretical results, and to examine their practical implications for common machine learning models.

4.6.1 Synthetic Data With Input Perturbation

First, we empirically validate our results from Section 4.5 using synthetic data and a logistic regression classifier, where the model's Lipschitz constant L and the data distribution's total variation constant M can be readily computed for the bound in Theorem 4.1.

Setup For each input $\boldsymbol{x} \in \mathbb{R}^4$ in the synthetic dataset, features are generated from a multivariate Gaussian distribution with zero mean, $\boldsymbol{x}_3, \boldsymbol{x}_4$ independent from each other and from $\boldsymbol{x}_1, \boldsymbol{x}_2$, and with a correlation ρ between $\boldsymbol{x}_1, \boldsymbol{x}_2$. That is, the covariance matrix $\Sigma \in \mathbb{R}^{4\times4}$ has the following form:

$$\Sigma = \begin{pmatrix} 1 & \rho & 0 & 0 \\ \rho & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The binary label $\boldsymbol{y} \in \{0, 1\}$ for each input is sampled from a Bernoulli distribution with $p = \sigma(\beta^\top \boldsymbol{x})$, where $\sigma(\cdot)$ is the sigmoid function and $\beta^\top = [5, 0, 3, 1]$. Unless stated otherwise, we consider $\rho = 1$, zeros are used for baseline feature removal, and the exact Gaussian distributions are used for marginal and conditional feature removal with 20,000 samples.

A logistic regression classifier was trained with 500 samples, and removal-based feature attributions were computed for 50 test samples (i.e., explicands) with and without input perturbation. In our experiment, perturbations are randomly sampled from standard Gaussian distributions with ℓ_2 norms normalized to values ranging between 0 to 2 with a total of 50 different values. For each perturbation norm, 50 perturbations are generated and applied to each explicand, resulting in 2,500 total explicand-perturbation pairs. As illustrated in Figure 4.1, a line is drawn to map the maximum ℓ_2 difference between original and perturbed attributions at each perturbation norm. The slope of such a line estimates the dependency of the upper bound of attribution differences with respect to the perturbation norm. Theorem 4.1 guarantees a specific slope, and this experiment aims to verify this property.

Furthermore, note that the sigmoid function $\sigma(\cdot)$ is 1/4-Lipschitz because its largest

derivative is 1/4. A logistic regression classifier is a composition of a linear function and sigmoid function $\sigma(\cdot)$, so the model is L-Lipschitz with $L = \|\hat{\beta}\|_2/4$, where $\hat{\beta}$ is the trained model coefficients. Finally, because the input features follow a Gaussian distribution, the total variation constant M derived in Example 4.1 applies here. Hence, we have specific values for all terms in Theorem 4.1 bound, which allows a direct comparison between theoretical and empirical results.

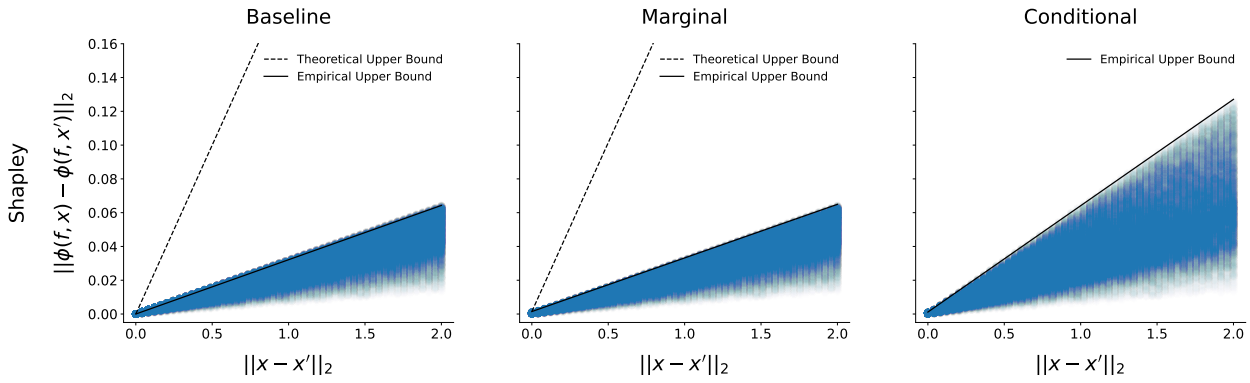


Figure 4.1: Shapley attribution differences under input perturbation with various perturbation norms in our synthetic data experiment. Lines bounding the maximum attribution difference at each perturbation norm are shown as empirical upper bounds. Theoretical upper bounds are included for baseline and marginal feature removal, but it is omitted for conditional feature removal because it is infinite for $\rho = 1$.

Results As shown in Figure 4.1, for all three feature removal approaches, the theoretical upper bounds for the Shapley value are indeed greater than the corresponding empirical upper bounds. We also observe that the empirical upper bound with conditional feature removal exceeds those with baseline and marginal feature removal, as expected due to our results involving the conditional distribution's total variation distance (Lemma 4.2).

Moreover, we vary the Lipschitz constant L of the trained logistic regression classifier by a factor of $\alpha = \{0.05, 0.1, 0.5, 1, 2\}$, which is done by multiplying the trained coefficients $\hat{\beta}$ by α. Figure 4.2 shows that the perturbation norm-dependent slope of the empirical bound increases linearly with α, aligning with our theoretical results. Also, Example 4.1 suggests

that increasing ρ would lead to a larger upper bound for conditional feature removal, which is observed in the empirical bound with $\rho = \{0, 0.25, 0.5, 0.75, 1\}$ (see Figure 4.2 right).

Overall, these empirical results validate our theoretical bound from Theorem 4.1. We note that gaps between our theoretical and empirical bounds exist, likely because we use the global Lipschitz constant to provide guarantee under all input perturbation strengths, instead of local constants for specific input regions and perturbation strengths, and also due to looseness in the total variation constant M. For example, our bound for multivariate Gaussian distributions in Example 4.1 suggests that $M = \infty$ when $\rho = 1$, which is far more conservative than our empirical results (see Figure 4.1 right); this is due to looseness in Pinsker's inequality (Appendix B.1). Similar results are observed for Occlusion, Banzhaf, RISE, and LIME attributions, and these are included in Appendix B.8.

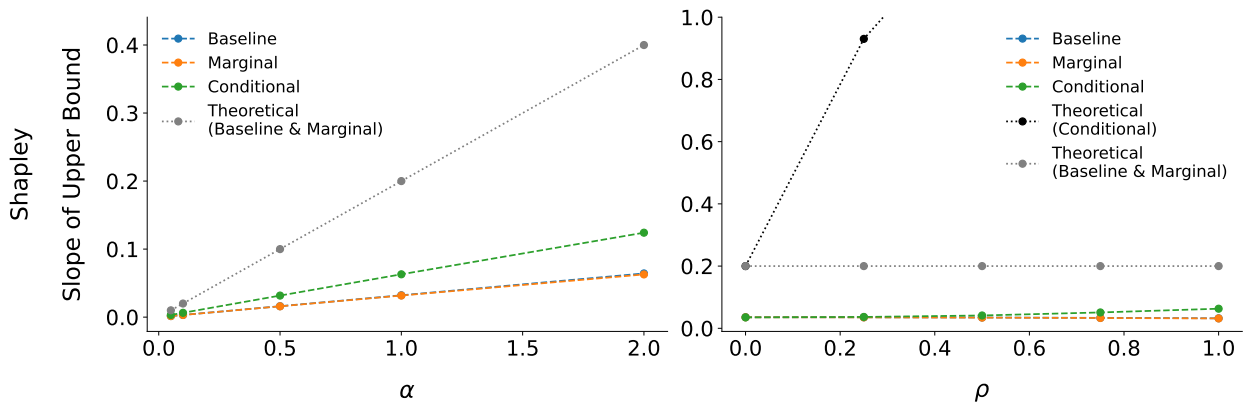


Figure 4.2: Slopes of empirical and theoretical upper bounds with respect to the input perturbation norm, for Shapley value attribution difference in the synthetic experiment. The parameter α varies the logistic regression Lipschitz constant, and ρ varies the correlation between the two synthetic features $\boldsymbol{x}_1, \boldsymbol{x}_2$.

4.6.2 Synthetic Data With Model Perturbation

Here, we construct a logistic regression classifier as well as a perturbed version such that their functional distance on the data manifold is small. We use this construction to empirically

verify that removing features with the conditional distribution leads to attributions that are more robust against model perturbation, as suggested by Theorem 4.2.

Setup The synthetic data generation follows the same process described above. A logistic regression classifier f is constructed with coefficients $[5, 0, 3, 1]$, and its perturbed counterpart f' has coefficients $[0, 5, 3, 1]$. Because the synthetic features $\boldsymbol{x}_1, \boldsymbol{x}_2$ are identical with the default value $\rho = 1$, we have the functional distance $\|f - f'\|_{\infty,\mathcal{X}} = 0$, so our theoretical bound suggests that $\|\phi(f, x) - \phi(f', x)\|_2 = 0$ for conditional feature removal. When we consider smaller values for ρ, we still expect better robustness for conditional rather than marginal removal due to Lemma 4.4, because significant probability mass lies in a subdomain where f and f' are functionally close (near the hyperplane where $\boldsymbol{x}_1 = \boldsymbol{x}_2$). Feature attributions are computed for 100 explicands with the same feature removal settings as above.

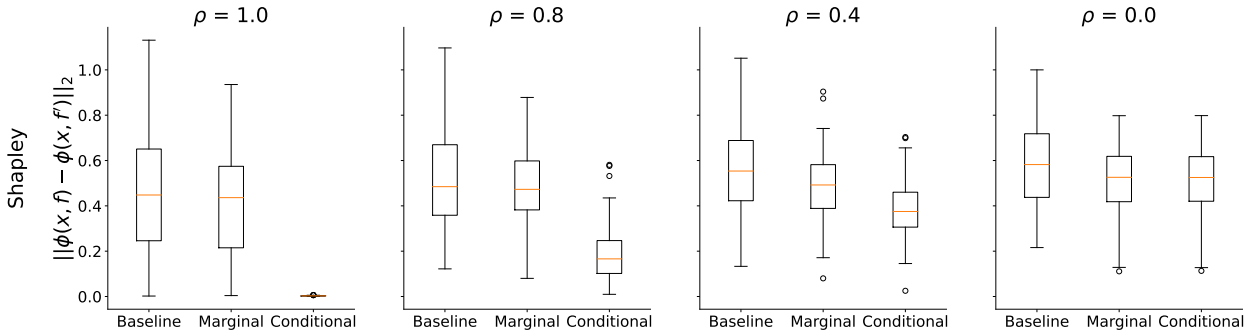


Figure 4.3: Shapley value attribution differences between the logistic regression classifiers f, f' with baseline, marginal, and conditional feature removal, and varying correlation ρ.

Results As suggested by our theoretical bound for model perturbation when $\rho = 1$, the Shapley attribution difference between f, f' is near zero when features are removed using their conditional distribution, whereas the attribution difference can be large with baseline and marginal feature removal (Figure 4.3). When we decrease ρ in the synthetic data generation, the models f, f' become more likely to generate different outputs, and we observe that the robustness gap between conditional vs. baseline and marginal feature removal reduces. These

empirical results corroborate Lemma 4.4. Similar results are observed for Occlusion, Banzhaf, RISE, and LIME attributions and are included in Appendix B.8. It is worth noting that the arbitrarily large bound for baseline and marginal feature removal underpins the adversarial attacks on LIME and SHAP demonstrated by Slack et al. (2020). Our theory and synthetic experiments for model perturbation are similar to those in Frye et al. (2021), but we provide a unified analysis for removal-based attribution methods instead of focusing on the Shapley value, and we consider cases where models disagree even on the data manifold (Lemma 4.4).

4.6.3 Implication I: Robust Attributions for Regularized Neural Networks

Our theoretical upper bound for input perturbation decreases when the model's Lipschitz constant becomes smaller (Theorem 4.1), suggesting that training a model with better Lipschitz regularity is one way to obtain more robust removal-based feature attributions. Weight decay is one way to achieve this, as it encourages lower Frobenius norm for the individual layers in a neural network, and the product of these Frobenius norms upper bounds the network's Lipschitz constant (Yoshida and Miyato, 2017). We therefore examine whether neural networks trained with weight decay produce attributions that are more robust against input perturbation.

Datasets The UCI white wine quality dataset (Cortez et al., 2009; Dua and Graff, 2017) consists of 4,898 samples of white wine, each with 11 input features for predicting the wine quality (whether the rating > 6). Two subsets of 500 samples are randomly chosen as the validation and test set. For a dataset with larger input dimension, we use MNIST (LeCun et al., 1998), which contains 28×28 grayscale images for digit classification. From the official training set, 10,000 images are randomly chosen as the validation set.

Setup Fully connected networks (MLPs) and convolutional networks (CNNs) are trained with a range of weight decay values for each dataset, where we use the values $\{0, 0.001, 0.0025, 0.005, 0.01\}$. Test accuracy does not differ much with these weight decay values (Figure B.12). Details

on model architectures and other hyperparameters are in Appendix B.10. Removal-based attributions are computed for the 500 test samples and all features in the wine quality dataset, and for 100 test images with 49 non-overlapping patches of size 4×4 as features in MNIST (see Appendix B.6 for a discussion of how our theory extends to the case with feature groups). For the wine quality dataset, removal-based attributions are computed exactly with all 2^{11} feature subsets. For MNIST, Shapley and Banzhaf values are computed with a least squares regression similar to KernelSHAP (Lundberg and Lee, 2017); Occlusion attributions are computed exactly, whereas Shapley, Banzhaf, LIME, and RISE are estimated with 10,000 random feature sets.

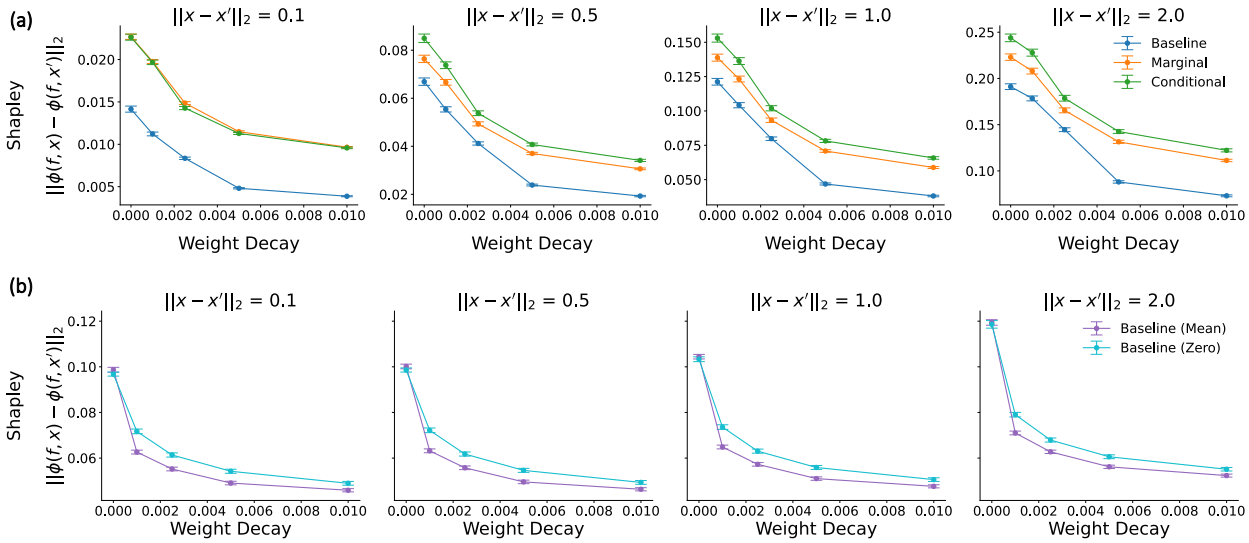


Figure 4.4: Shapley attribution difference for networks trained with increasing weight decay, under input perturbations with perturbation norms $\{0.1, 0.5, 1, 2\}$. The results include (a) the wine quality dataset with MLPs and baseline (replacing with training set minimums), marginal, and conditional feature removal; and (b) MNIST with CNNs and baseline feature removal with either training set means or zeros. Error bars show the mean and 95% confidence intervals across explicand-perturbation pairs.

When using baseline feature removal, features are replaced with the training set minimums for the wine quality dataset, and with the training set means or zeros for MNIST. The wine quality training set is used for removing features with their marginal distributions,

while a conditional variational autoencoder (CVAE) is trained following the procedure in Frye et al. (2021) to approximate feature removal with the conditional distribution (details in Appendix B.10). For both marginal and conditional feature removal in the wine quality dataset, 250 samples are used. Marginal and conditional feature removal are computationally expensive and less common for image data, so we use only baseline feature removal as is often done in practice. Finally, input perturbations are generated as in the previous experiment, with 10 random perturbations applied per explicand for the perturbation norms $\{0.1, 0.5, 1, 2\}$.

Results Figure 4.4 shows that Shapley value attribution differences decrease with increasing weight decay, across different feature removal techniques, for both the wine quality dataset and MNIST, and for all perturbation norms in $\{0.1, 0.5, 1, 2\}$. This demonstrates the practical implication that removal-based attributions can be made more robust against input perturbation by training with weight decay. Interestingly, weight decay has also been shown to improve the robustness of gradient-based attributions (Dombrowski et al., 2022). In Appendix B.9, similar results are observed for Occlusion, Banzhaf, RISE, and LIME attributions, and also when we measure explanation similarity using Pearson or Spearman rank correlation rather than ℓ_2 distance.

4.6.4 Implication II: Sanity Checks for Removal-Based Attributions

The robustness bound for model perturbation in Theorem 4.2 suggests that the attribution difference between an intact and perturbed model is guaranteed to be small if the model perturbation is small. Similarly, we expect that if the perturbation intensifies, the attribution differences may become larger. One way to perturb a neural network with increasing intensity is to sequentially randomize its parameters from the output to input layer, a procedure known as *cascading randomization* that was introduced by Adebayo et al. (2018) in the context of sanity checking feature attributions. Here, we study whether removal-based attributions indeed pass the parameter randomization check, as suggested by our theoretical bound.

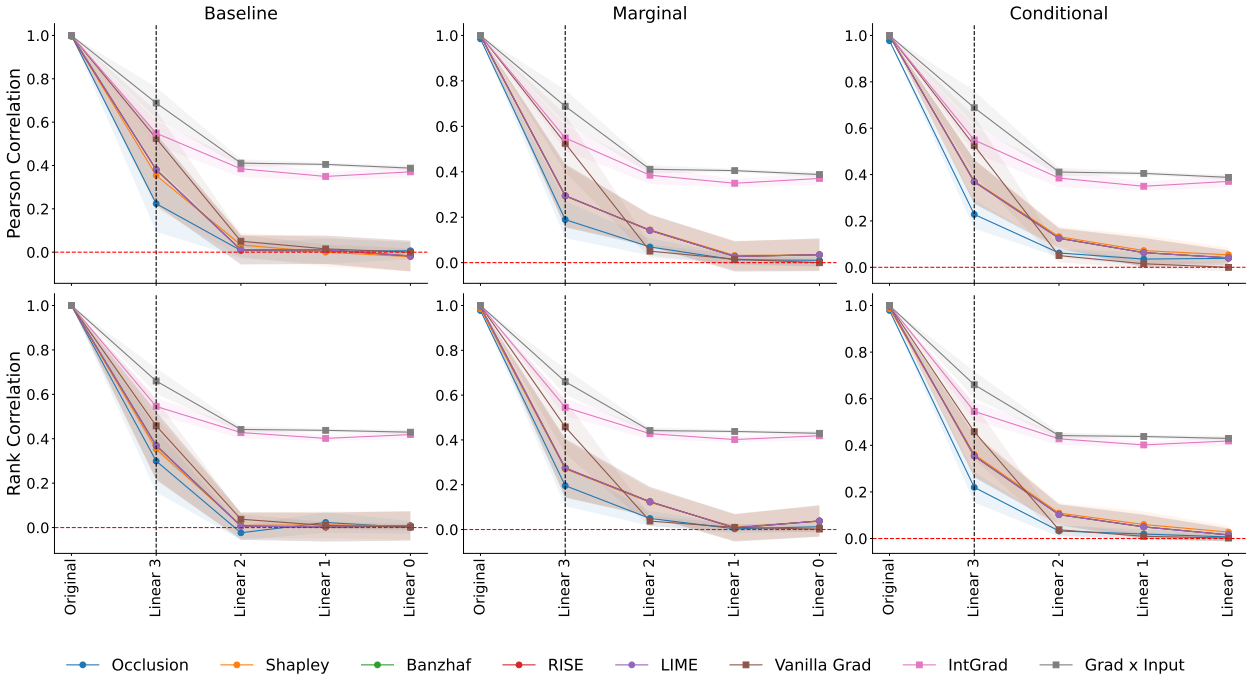


Figure 4.5: Sanity checks for attributions using cascading randomization for the MLP trained on the wine quality dataset. Attribution similarity is measured by Pearson correlation and Spearman rank correlation. We show the mean and 95% confidence intervals across 10 random seeds.

Datasets We use the same datasets and splits as above.

Setup The MLP and CNN without weight decay from the previous experiment are used here. The settings for removal-based attributions are the same as in the previous experiment. Parameters for each network are randomized in a cascading fashion from the output to input layer as in Adebayo et al. (2018), with attributions computed after randomizing each layer and compared to the original attributions using Pearson correlation and Spearman rank correlation. Randomization is performed using the layer's default initialization. Feature attributions with the original model are computed twice and compared to capture variations

due to sampling in feature removal and in attribution estimation using feature sets.[2] This experiment is run 10 times with different random seeds. Gradient-based attribution methods such as Vanilla Gradients (Simonyan et al., 2013), IntGrad (Sundararajan et al., 2017), and Grad $\times$ Input (Shrikumar et al., 2017) are included as references, as they were the focus of the original sanity checking work (Adebayo et al., 2018).

Results As suggested by our robustness bound for model perturbation, the original and perturbed removal-based attributions become dissimilar as more parameters are randomized, leading to near-zero correlations when the entire network is re-initialized (Figure 4.5). This shows that removal-based attributions reliably pass the model randomization sanity check. In contrast, we observe that for IntGrad and Grad $\times$ Input, the correlation between intact and perturbed attributions remains around 0.5 even when the entire network is randomized. Similarly, for the CNN trained on MNIST, the original and perturbed removal-based attributions have lower correlations as more parameters are randomized, whereas the correlations decrease only slightly for IntGrad and Grad $\times$ Input (Figure 4.6).

4.7 Discussion

Our analysis focused on the robustness properties of removal-based feature attributions under perturbations to both the model f and input $\boldsymbol{x}$. For input perturbations, Theorem 4.1 shows that the change in attribution is proportional to the input perturbation strength. The Lipchitz constant is determined by the attribution's removal and summary technique, as well as the model's inherent robustness. We find that both implementation choices play an important role, but our analysis in Section 4.4 reveals that the associated factors in our main result can only be so small for meaningful attributions, implying that attribution robustness in some sense reduces to model robustness. Fortunately, there are many existing works on improving a neural network's Lipschitz regularity (Virmaux and Scaman, 2018; Gouk et al.,

[2]We observed differences in the re-computed explanations for MNIST when using rank correlation. This is likely due to noise in the attribution scores that are closest to zero, because the differences are not detected when using Pearson correlation (Figure 4.6).

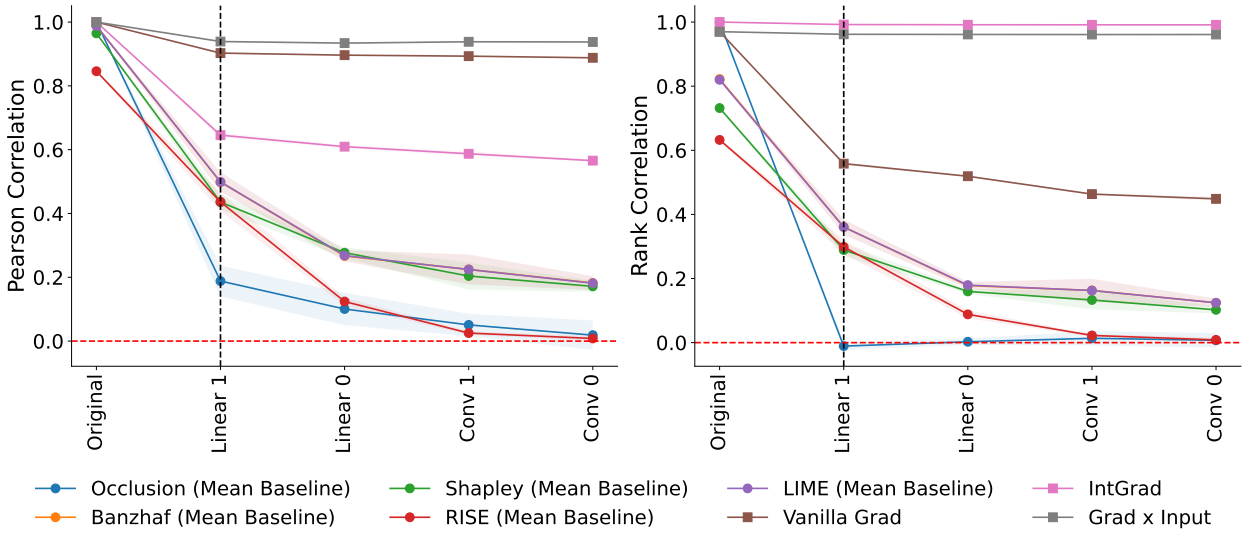


Figure 4.6: Sanity checks for attributions using cascading randomization for the CNN trained on MNIST. Attribution similarity is measured by Pearson correlation and Spearman rank correlation. We show the mean and 95% confidence intervals across 10 random seeds.

2021; Araujo et al., 2023). For model perturbations, Theorem 4.2 shows that the attribution difference is proportional to functional distance between the perturbed and original models, as measured by the Chebyshev distance (or infinity norm). Depending on the removal approach, the distance may depend on only a subdomain of the input space. One practical implication is that attributions should remain similar under minimal model changes, e.g., accurate model distillation or quantization. Another implication is that bringing the model closer to its correct form (e.g., the Bayes classifier) results in attributions closer to those of the correct model; several recent works have demonstrated this by exploring ensembled attributions (Dong and Rudin, 2020; Janizek et al., 2021; Ning et al., 2022; Gyawali et al., 2022), which are in some cases mathematically equivalent to attributions of an ensembled model.

Our robustness results provide a new lens to compare different removal-based attribution methods. In terms of the summary technique, our main proof technique relies on the associated operator norm, which does not distinguish between several approaches (Shapley,

leave-one-out, Banzhaf); however, our results for the spectral norm suggest that the Banzhaf value is in some sense most robust. This result echoes recent work (Wang and Jia, 2022) and has implications for other applications of game-theoretic credit allocation, but we also find that the Shapley value is most robust within a specific class of summary techniques. Our findings regarding the removal approach are more complex. The conditional distribution approach has been praised by some works for being more informative than baseline or marginal (Frye et al., 2021; Covert et al., 2021), but this comes at a cost of *worse* robustness to input perturbations. On the other hand, it leads to *improved* robustness to model perturbations, which has implications for fooling attributions with imperceptible off-manifold model changes (Slack et al., 2020).

Next, we make connections between current understanding of gradient-based methods and our findings for removal-based methods. Overall, the notion of Lipschitz continuity is important for robustness guarantees under input perturbations. Our results on removal-based attributions rely on Lipschitz continuity of the model itself, whereas gradient-based attributions rely on Lipschitz continuity of the model *gradient* (i.e., Lipschitz smoothness) (Dombrowski et al., 2022). Practically, both are computationally intensive to determine exactly for realistic networks (Virmaux and Scaman, 2018; Gouk et al., 2021). Empirically, weight decay is an effective approach for generating robust attributions, for both removal-based (our results) and gradient-based methods (see Dombrowski et al. 2022). Under model perturbations, removal-based attributions are more robust when features are removed such that $\boldsymbol{x}_S$ stay on the data manifold, whereas gradient-based attributions are more robust when gradients are projected on the tangent space of the data manifold (Dombrowski et al., 2019).

Finally, we consider limitations of our analysis. Future work may consider bounding attribution differences via different distance measures (e.g., ℓ_p norms for $p \neq 2$). The main limitation to our results is that they are conservative. By relying on global properties of the model, i.e., the Lipschitz constant and global prediction bounds, we arrive at worst-case bounds that are in many cases overly conservative, as reflected in our experiments. An

alternative approach can focus on localized versions of our bounds. Analogous to recent work on certified robustness (Katz et al., 2017; Tjeng et al., 2017; Raghunathan et al., 2018; Singh et al., 2019; Jordan et al., 2022), tighter bounds may be achievable with specific perturbation strengths, unlike our approach that provides bounds simultaneously for any perturbation strength.

Part II

SHAPLEY VALUE COMPUTATION

Chapter 5

PRACTICAL SHAPLEY VALUE ESTIMATION VIA LINEAR REGRESSION

5.1 Chapter Notes

This chapter presents joint work with Su-In Lee that was published in the International Conference on Artificial Intelligence and Statistics (AISTATS) in 2021. The content of this chapter is a minor variant of the published version of the paper. The main contribution of this work is to analyze and introduce several improvements for the Shapley value's weighted least squares estimator, which is commonly known as KernelSHAP (Lundberg and Lee, 2017).

5.2 Introduction

Shapley values are the basis of many model explanation methods (e.g., SHAP, IME, QII, Shapley Effects, Shapley Net Effects, SAGE) (Lundberg and Lee, 2017; Lundberg et al., 2020; Štrumbelj and Kononenko, 2014; Datta et al., 2016; Owen, 2014; Lipovetsky and Conklin, 2001; Covert et al., 2020). They were originally developed in cooperative game theory (Shapley, 1953), but recent work shows that Shapley values provide a powerful tool for explaining how ML models work when either individual features (Lundberg and Lee, 2017), neurons in a neural network (Ghorbani and Zou, 2020), or samples in a dataset (Ghorbani and Zou, 2019) are viewed as players in a cooperative game. They have become a go-to solution for allocating credit and quantifying contributions, largely due to to their appealing theoretical properties (Shapley, 1953; Monderer et al., 2002).

The main challenge when using Shapley values is calculating them efficiently. A naive calculation has computational complexity that is exponential in the number of players, so numerous approaches have been proposed to accelerate their calculation. Besides brute-

force methods (Lipovetsky and Conklin, 2001), other techniques include sampling-based approximations (Štrumbelj and Kononenko, 2014; Song et al., 2016; Chen et al., 2018b; Covert et al., 2020), model-specific approximations (e.g., TreeSHAP) (Ancona et al., 2019; Lundberg et al., 2020) and a linear regression-based approximation (KernelSHAP) (Lundberg and Lee, 2017). Among these, sampling- and regression-based approaches have the advantage of being applicable to any model type, which is useful for models like neural networks.

Here, we revisit the regression-based approach developed by Lundberg and Lee (2017), which is commonly known as KernelSHAP, to address several shortcomings. Recent work has questioned whether KernelSHAP is an unbiased estimator (Merrick and Taly, 2019), and unlike sampling-based estimators (Castro et al., 2009; Maleki et al., 2013; Covert et al., 2020), KernelSHAP does not provide uncertainty estimates. Furthermore, it provides no guidance on the number of samples required because its convergence properties are not well understood. We address each of these problems, in part by building on a newly proposed unbiased version of the regression-based approach. Our contributions include:

1. We derive an unbiased version of KernelSHAP, and we show that the original version incurs a negligible increase in bias in exchange for significantly lower variance.

2. We show how to detect KernelSHAP's convergence, automatically determine the number of samples required, and calculate uncertainty estimates for the final results.

3. We propose a variance reduction technique that accelerates KernelSHAP's convergence with no additional computational overhead.

4. We adapt the regression-based approach to stochastic cooperative games (Charnes and Granot, 1973) to provide fast new approximations for two global explanation methods, SAGE (Covert et al., 2020) and Shapley Effects (Owen, 2014).

With these new insights and tools, we aim to offer both an improved theoretical under-

standing and a more practical approach to Shapley value estimation via linear regression.[1]

5.3 Background

We now provide background information on the Shapley value, including the mathematical perspective that we build on with regression-based estimators.

5.3.1 Cooperative Games

As described in Chapter 2, a cooperative game is a function $v : \mathcal{P}(d) \mapsto \mathbb{R}$ that returns a value for each coalition $S \subseteq [d]$, where $[d] \equiv \{1, \ldots, d\}$ represents a set of players. Cooperative game theory has become increasingly important in ML, as many methods have been developed that frame model explanation problems in terms of cooperative games; see Chapter 3 for a detailed discussion. Notably, SHAP (Lundberg and Lee, 2017), IME (Štrumbelj and Kononenko, 2014) and QII (Datta et al., 2016) define cooperative games that represent an individual prediction's dependence on different features. For a model f and an input x, SHAP analyzes a cooperative game v_x defined as

$$v_x(S) = \mathop{\mathbb{E}}_{p(\boldsymbol{x}_{\bar{S}})}[f(x_S, \boldsymbol{x}_{\bar{S}})], \tag{5.1}$$

where $x_S \equiv \{x_i : i \in S\}$ represents a feature subset and $\boldsymbol{x}_S$ is the corresponding random variable, and we assume that features are removed using their marginal distribution (Lundberg and Lee, 2017). Two other methods, Shapley Effects (Owen, 2014) and SAGE (Covert et al., 2020), define cooperative games that represent a model's behavior across the entire dataset. Given a loss function ℓ and response variable $\boldsymbol{y}$, SAGE uses a cooperative game w that represents the model's predictive accuracy given a feature subset $\boldsymbol{x}_S$:

$$w(S) = -\mathop{\mathbb{E}}_{p(\boldsymbol{x}_S, \boldsymbol{y})}\big[\ell(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S], \boldsymbol{y})\big]. \tag{5.2}$$

[1] `https://github.com/iancovert/shapley-regression`

Several other techniques also frame model explanation in terms of cooperative games, where a target quantity (e.g., the model loss) varies as groups of players (e.g., input features) are removed, and the Shapley value is used to summarize each player's contribution (Covert et al., 2021).

5.3.2 Shapley Values

The Shapley value (Shapley, 1953) assumes that the *grand coalition* $[d]$ participates and seeks to provide each player $i \in [d]$ with a fair allocation of the total profit, which is represented by $v([d])$. Fair allocations must be based on each player's contribution to the profit, but a player's contribution is often difficult to define because it depends on the preceding players. The Shapley value resolves this problem by deriving a unique value based on a set of fairness axioms; see Chapter 2 for a discussion of these properties, or Shapley (1953); Monderer et al. (2002) for further details. It can be understood as a player's average marginal contribution across all possible player orderings, and each player's Shapley value $\phi_1(v), \ldots, \phi_d(v) \in \mathbb{R}$ for a game v is given by:

$$\phi_i(v) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \Big(v(S \cup i) - v(S) \Big). \tag{5.3}$$

Many model explanation methods can be understood in terms of ideas from cooperative game theory, but the Shapley value is especially popular and is also widely used in other fields (Aumann, 1994; Petrosjan and Zaccour, 2003; Tarashev et al., 2016).

5.3.3 Weighted Least Squares Characterization

We can understand the Shapley value in several ways, but the perspective most relevant for this work is that it is the solution to a weighted least squares problem. Many works have considered fitting simple models to cooperative games (Charnes et al., 1988; Hammer and Holzman, 1992; Grabisch et al., 2000; Ding et al., 2008, 2010; Marichal and Mathonet, 2011),

particularly additive models of the form

$$u(S) = \beta_0 + \sum_{i \in S} \beta_i.$$

Such additive models are known as *inessential games*, and although a game v may not be inessential, an inessential approximation can help summarize each player's average contribution. Several works model games by solving a weighted least squares problem using a weighting function $\mu(S)$ (Charnes et al., 1988; Hammer and Holzman, 1992; Ding et al., 2008), or by optimizing the following objective:

$$\min_{\beta_0,\ldots,\beta_d} \sum_{S \subseteq [d]} \mu(S)\big(u(S) - v(S)\big)^2.$$

Perhaps surprisingly, different weighting kernels μ lead to recognizable optimal regression coefficients $(\beta_1^*, \ldots \beta_d^*)$ (Covert et al., 2021). In particular, a carefully chosen weighting kernel yields regression coefficients equal to the Shapley values (Charnes et al., 1988; Lundberg and Lee, 2017). The Shapley kernel μ_{Sh} is given by

$$\mu_{\text{Sh}}(S) = \frac{d-1}{\binom{d}{|S|}|S|(d-|S|)},$$

where the values $\mu_{\text{Sh}}(\varnothing) = \mu_{\text{Sh}}([d]) = \infty$ effectively enforce constraints $\beta_0 = v(\varnothing)$ for the intercept and $\sum_{i \in [d]} \beta_i = v([d]) - v(\varnothing)$ for the sum of the coefficients. Lundberg and Lee (2017) used this Shapley value interpretation when developing an approach to approximate SHAP values via linear regression, which we discuss next.

5.4 Linear Regression Estimators

As described above, Shapley values are difficult to calculate because they require examining each player's marginal contribution to every possible subset (eq. 5.3). This leads to run-times that are exponential in the number of players, so efficient approximations are important in

practice and have been explored by many prior works (Štrumbelj and Kononenko, 2014; Song et al., 2016; Lundberg and Lee, 2017; Chen et al., 2018b; Ancona et al., 2019; Lundberg et al., 2020; Covert et al., 2020). Here, we revisit the regression-based approach presented by Lundberg and Lee (2017), and we then develop an unbiased version whose properties are simpler to analyze.

5.4.1 Optimization Objective

The least squares characterization of the Shapley value suggests that we can calculate the values $\phi_1(v), \ldots, \phi_d(v)$ by solving the following optimization problem:

$$\begin{aligned} &\min_{\beta_0,\ldots,\beta_d} \sum_{0<|S|<d} \mu_{\text{Sh}}(S)\big(\beta_0 + \sum_{i\in S}\beta_i - v(S)\big)^2 \\ &\text{s.t.} \quad \beta_0 = v(\varnothing), \quad \beta_0 + \sum_{i=1}^{d}\beta_i = v([d]). \end{aligned} \tag{5.4}$$

We introduce new notation to make the problem's solution easier to write. First, we denote the non-intercept coefficients as $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$. Next, we denote each subset $S \subseteq [d]$ using the corresponding binary vector $z \in \{0,1\}^d$, and with abuse of notation we write $v(z) \equiv v(S)$ and $\mu_{\text{Sh}}(z) \equiv \mu_{\text{Sh}}(S)$ for $S = \{i \in [d] : z_i = 1\}$. Lastly, we denote a distribution over z using $p(\boldsymbol{z})$, where we define $p(z) \propto \mu_{\text{Sh}}(z)$ when $0 < \mathbf{1}^\top z < d$ and $p(z) = 0$ otherwise. With this, we can rewrite the optimization problem as

$$\begin{aligned} &\min_{\beta} \sum_{z\in\{0,1\}^d} p(z)\big(v(\mathbf{0}) + z^\top\beta - v(z)\big)^2 \\ &\text{s.t.} \qquad \mathbf{1}^\top\beta = v(\mathbf{1}) - v(\mathbf{0}). \end{aligned} \tag{5.5}$$

5.4.2 Dataset Sampling

Solving the problem in eq. 5.5 requires evaluating the cooperative game v with all 2^d coalitions. Evaluating $v(\mathbf{0})$ and $v(\mathbf{1})$ is sufficient to ensure that the constraints are satisfied, but

we require all values $v(z)$ for z such that $0 < \mathbf{1}^\top z < d$ to fit the model exactly. KernelSHAP manages this challenge by subsampling a dataset and optimizing an approximate objective, and we refer to this approach as *dataset sampling*. Using n independent samples $z_i \sim p(\boldsymbol{z})$ and their values $v(z_i)$, KernelSHAP solves the following problem:

$$\begin{aligned} \min_{\beta} \ & \frac{1}{n}\sum_{i=1}^{n}\big(v(\mathbf{0}) + z_i^\top \beta - v(z_i)\big)^2 \\ \text{s.t.} \quad & \mathbf{1}^\top \beta = v(\mathbf{1}) - v(\mathbf{0}). \end{aligned} \tag{5.6}$$

The dataset sampling approach, also applied by LIME (Ribeiro et al., 2016), offers the flexibility to use only enough samples to accurately approximate the objective. Given a set of samples $(z_1, \ldots, z_n)$, solving this problem is straightforward. The Lagrangian with multiplier $\nu \in \mathbb{R}$ is given by:

$$\begin{aligned} \hat{\mathcal{L}}(\beta, \nu) = \beta^\top \Big(\frac{1}{n}\sum_{i=1}^{n} z_i z_i^\top\Big)\beta - 2\beta^\top \Big(\frac{1}{n}\sum_{i=1}^{n} z_i\big(v(z_i) - v(\mathbf{0})\big)\Big) \\ + \frac{1}{n}\sum_{i=1}^{n}\big(v(z_i) - v(\mathbf{0})\big)^2 + 2\nu\big(\mathbf{1}^\top\beta - v(\mathbf{1}) + v(\mathbf{0})\big). \end{aligned}$$

If we introduce the shorthand notation

$$\hat{A}_n = \frac{1}{n}\sum_{i=1}^{n} z_i z_i^\top \quad \text{and} \quad \hat{b}_n = \frac{1}{n}\sum_{i=1}^{n} z_i\big(v(z_i) - v(\mathbf{0})\big),$$

we can use the problem's KKT conditions (Boyd et al., 2004) to derive the following solution:

$$\hat{\beta}_n = \hat{A}_n^{-1}\Big(\hat{b}_n - \mathbf{1}\frac{\mathbf{1}^\top \hat{A}_n^{-1}\hat{b}_n - v(\mathbf{1}) + v(\mathbf{0})}{\mathbf{1}^\top \hat{A}_n^{-1}\mathbf{1}}\Big). \tag{5.7}$$

This method is commonly known as KernelSHAP (Lundberg and Lee, 2017), and the implementation in the SHAP repository[2] also allows for regularization terms in the approximate

[2] `http://github.com/slundberg/shap`

objective (eq. 5.6). While this approach is intuitive and simple to implement, the estimator $\hat{\beta}_n$ is not currently well understood (Merrick and Taly, 2019). As we show in Section 5.5, understanding its bias, variance and rate of convergence is not straightforward. We therefore derive an alternative approach that is simpler to analyze, which we discuss next.

5.4.3 An Exact Estimator

Consider the solution to the problem that uses all 2^d player coalitions (eq. 5.5). Rather than finding an *exact solution to an approximate problem* (eq. 5.7), we now derive an *approximate solution to the exact problem.* The full problem's Lagrangian is given by

$$\begin{aligned}\mathcal{L}(\beta, \nu) = \beta^\top \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top]\beta - 2\beta^\top \mathbb{E}\left[\boldsymbol{z}\big(v(\boldsymbol{z}) - v(\mathbf{0})\big)\right] \\ + \mathbb{E}\left[\big(v(\boldsymbol{z}) - v(\mathbf{0})\big)^2\right] + 2\nu\big(\mathbf{1}^\top\beta - v(\mathbf{1}) + v(\mathbf{0})\big),\end{aligned}$$

where we now consider $\boldsymbol{z}$ to be a random variable distributed according to $p(\boldsymbol{z})$. Using the shorthand notation

$$A = \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top] \quad \text{and} \quad b = \mathbb{E}\left[\boldsymbol{z}\big(v(\boldsymbol{z}) - v(\mathbf{0})\big)\right],$$

we can write the solution to the exact problem as:

$$\beta^* = A^{-1}\Big(b - \mathbf{1}\frac{\mathbf{1}^\top A^{-1} b - v(\mathbf{1}) + v(\mathbf{0})}{\mathbf{1}^\top A^{-1}\mathbf{1}}\Big). \tag{5.8}$$

Due to our setup of the problem, we have the property that $\beta_i^* = \phi_i(v)$. Unfortunately, we cannot evaluate this expression in practice without first evaluating v for all 2^d coalitions $S \subseteq [d]$. However, knowledge of $p(\boldsymbol{z})$ means that $A \in \mathbb{R}^{d\times d}$ can be calculated exactly and efficiently. To see this, note that $(\boldsymbol{z}\boldsymbol{z}^\top)_{ij} = \boldsymbol{z}_i\boldsymbol{z}_j = \mathbb{1}(\boldsymbol{z}_i = \boldsymbol{z}_j = 1)$. Therefore, to calculate A, we only need to calculate $p(\boldsymbol{z}_i = 1)$ for diagonal values A_{ii} and $p(\boldsymbol{z}_i = \boldsymbol{z}_j = 1)$ for off-diagonal values A_{ij}. See Appendix C.1 for their derivations, which are easy to compute in practice.

Since b cannot be calculated exactly and efficiently due to its dependence on v, this suggests that we should use A's exact form and approximate β^* by estimating *only* b. We propose the following estimator:

$$\bar{b}_n = \frac{1}{n}\sum_{i=1}^{n} z_i v(z_i) - \mathbb{E}[\boldsymbol{z}]v(\mathbf{0}).$$

Using this, we arrive at an alternative to the original KernelSHAP estimator, which we refer to as *unbiased KernelSHAP*:

$$\bar{\beta}_n = A^{-1}\Big(\bar{b}_n - \mathbf{1}\frac{\mathbf{1}^\top A^{-1}\bar{b}_n - v(\mathbf{1}) + v(\mathbf{0})}{\mathbf{1}^\top A^{-1}\mathbf{1}}\Big). \tag{5.9}$$

In the next section, we compare this to the original KernelSHAP estimator $\hat{\beta}_n$ both theoretically and empirically.

5.5 Estimator Properties

We now analyze the consistency, bias and variance properties of the Shapley value estimators described above, and we consider how to detect, forecast, and accelerate their convergence.

5.5.1 Consistency, Bias and Variance

A *consistent* estimator in this context is one that converges to the correct Shapley values β^* given a sufficient number of samples. If the game v has bounded value, then the strong law of large numbers implies that

$$\lim_{n\to\infty} \hat{A}_n = A \quad \text{and} \quad \lim_{n\to\infty} \hat{b}_n = \lim_{n\to\infty} \bar{b}_n = b,$$

and the convergence is almost sure. From this, we can see that both estimators are consistent:

$$\lim_{n\to\infty} \hat{\beta}_n = \lim_{n\to\infty} \bar{\beta}_n = \beta^*.$$

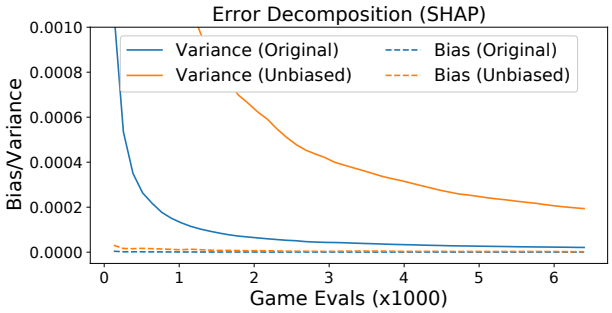


Figure 5.1: SHAP error decomposition for the original and unbiased KernelSHAP estimators. The bias and variance are calculated empirically across 250 runs.

Next, an *unbiased* estimator is one whose expectation is equal to the correct Shapley values β^*. This is difficult to verify for the KernelSHAP estimator $\hat{\beta}_n$ due to the multiplicative interaction between $\hat{A}_n$ and $\hat{b}_n$ (see eq. 5.7). Both $\hat{A}_n$ and $\hat{b}_n$ are unbiased, but terms like $\mathbb{E}[\hat{A}_n^{-1}\hat{b}_n]$ and $\mathbb{E}[\hat{A}_n^{-1}\mathbf{1}\mathbf{1}^\top\hat{A}_n^{-1}\hat{b}_n/(\mathbf{1}^\top\hat{A}_n^{-1}\mathbf{1})]$ are more difficult to characterize. When making claims about KernelSHAP's bias, we rely instead on empirical evidence.

In contrast, it is easy to see that the alternative estimator $\bar{\beta}_n$ is unbiased. Because of its linear dependence on $\bar{b}_n$ and the fact that $\mathbb{E}[\bar{b}_n] = b$, we can see that $\mathbb{E}[\bar{\beta}_n] = \beta^*$. We therefore conclude that the alternative estimator $\bar{\beta}_n$ is both consistent and unbiased, whereas the original KernelSHAP ($\hat{\beta}_n$) is only provably consistent. It is for this reason that we refer to $\bar{\beta}_n$ as *unbiased KernelSHAP*.

Regarding the estimators' variance, unbiased KernelSHAP is again easier to characterize. The values $\bar{\beta}_n$ are a function of $\bar{b}_n$, and the multivariate central limit theorem (CLT) (Van der Vaart, 2000) asserts that $\bar{b}_n$ converges in distribution to a multivariate Gaussian, or

$$\bar{b}_n\sqrt{n} \xrightarrow{D} \mathcal{N}(b, \Sigma_{\bar{b}}), \tag{5.10}$$

where $\Sigma_{\bar{b}} = \mathrm{Cov}\big(\boldsymbol{z}v(\boldsymbol{z})\big)$. This implies that for the estimator $\bar{\beta}_n$, we have the convergence

property

$$\bar{\beta}_n \sqrt{n} \xrightarrow{D} \mathcal{N}(\beta^*, \Sigma_{\bar{\beta}}), \tag{5.11}$$

where, due to its linear dependence on $\bar{b}_n$ (see eq. 5.9), we have the covariance $\Sigma_{\bar{\beta}}$ given by

$$\Sigma_{\bar{\beta}} = C \Sigma_{\bar{b}} C^\top \qquad \text{with} \qquad C = A^{-1} - \frac{A^{-1}\mathbf{1}\mathbf{1}^\top A^{-1}}{\mathbf{1}^\top A^{-1}\mathbf{1}}. \tag{5.12}$$

This allows us to reason about unbiased KernelSHAP's asymptotic distribution. In particular, we remark that $\bar{\beta}_n$ has variance that reduces at a rate of $\mathcal{O}(\frac{1}{n})$. Note that in comparison, the original KernelSHAP estimator $\hat{\beta}_n$ is difficult to analyze due to the interaction between the $\hat{A}_n$ and $\hat{b}_n$ terms. We can apply the CLT to each term individually, but reasoning about $\hat{\beta}_n$'s distribution remains challenging.

To facilitate our comparison between these two estimators, we present a simple experiment to analyze them empirically. We approximated the SHAP values for an individual prediction in the census income dataset (Lichman et al., 2013) and empirically calculated the mean squared error relative to the true SHAP values across 250 runs.[3] We then decomposed the error into bias and variance terms as follows:

$$\underbrace{\mathbb{E}\left[||\hat{\beta}_n - \beta^*||^2\right]}_{\text{Error}} = \underbrace{\mathbb{E}\left[||\hat{\beta}_n - \mathbb{E}[\hat{\beta}_n]||^2\right]}_{\text{Variance}} + \underbrace{\left|\left|\, \mathbb{E}[\hat{\beta}_n] - \beta^* \right|\right|^2}_{\text{Bias}}.$$

Figure 5.1 shows that the error for both estimators is dominated by variance rather than bias.[4] It also shows that KernelSHAP incurs virtually no bias in exchange for significantly lower variance. In Appendix C.4, we provide global measures of the bias and variance to confirm these observations across multiple examples and two other datasets. This suggests that although KernelSHAP is more difficult to analyze theoretically, it should be used in

[3]The true SHAP values use a sufficient number of samples to ensure convergence, as described later in this section.

[4]The unbiased approach appears to have higher bias due to estimation error, but its bias is provably equal to zero. This is therefore due to sampling error.

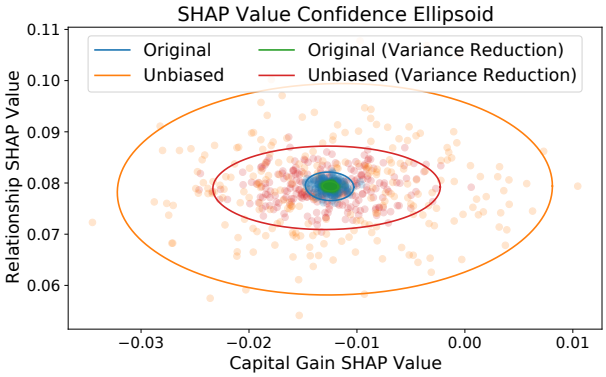


Figure 5.2: Gaussian 95% confidence ellipsoids for two SHAP values in a census income prediction (from 250 runs). The estimators use an equal number of samples.

practice because its bias is negligible and it converges faster.

5.5.2 Variance Reduction via Paired Sampling

Having analyzed each estimator's properties, we now consider whether their convergence can be accelerated. We propose a simple variance reduction technique that leads to significantly faster convergence in practice.

When sampling n subsets according to the distribution $z_i \sim p(\boldsymbol{z})$, we suggest using a *paired sampling* strategy where each sample z_i is paired with its complement[5] $\mathbf{1} - z_i$. To show why this approach accelerates convergence, we focus on unbiased KernelSHAP, which is easier to analyze theoretically.

When estimating b for unbiased KernelSHAP ($\bar{\beta}_n$), consider using the following modified estimator that combines z_i with $\mathbf{1} - z_i$:

$$\check{b}_n = \frac{1}{2n} \sum_{i=1}^{n} \big(z_i v(z_i) + (\mathbf{1} - z_i) v(\mathbf{1} - z_i) - v(\mathbf{0}) \big). \tag{5.13}$$

[5]We call $\mathbf{1} - z$ the *complement* because it is the binary vector for $[d] \setminus S$, where S corresponds to z. Such approaches are sometimes referred to as *antithetical sampling*.

Substituting this into unbiased KernelSHAP (eq. 5.9) yields a new estimator $\check{\beta}_n$ that preserves the properties of being both consistent and unbiased:

$$\check{\beta}_n = A^{-1}\Big(\check{b}_n - \mathbf{1}\frac{\mathbf{1}^\top A^{-1}\check{b}_n - v(\mathbf{1}) + v(\mathbf{0})}{\mathbf{1}^\top A^{-1}\mathbf{1}}\Big). \tag{5.14}$$

For games v that satisfy a condition that we describe below, we can guarantee that this sampling approach leads to $\check{\beta}_n$ having lower variance than $\bar{\beta}_n$, even when we account for $\check{b}_n$ requiring twice as many cooperative game evaluations $v(z)$ as $\bar{b}_n$ (see proof in Appendix C.1).

Theorem 5.1. *The difference between the covariance matrices for the estimators $\bar{\beta}_{2n}$ and $\check{\beta}_n$ is given by*

$$\mathrm{Cov}(\bar{\beta}_{2n}) - \mathrm{Cov}(\check{\beta}_n) = \frac{1}{2n} C G_v C^\top,$$

where G_v is a property of the game v, defined as

$$G_v = -\mathrm{Cov}\Big(\boldsymbol{z}v(\boldsymbol{z}), (\mathbf{1}-\boldsymbol{z})v(\mathbf{1}-\boldsymbol{z})\Big).$$

For sufficiently large n, $G_v \succeq 0$ guarantees that the Gaussian confidence ellipsoid $\bar{E}_{2n,\alpha}$ for $\bar{\beta}_{2n}$ contains the corresponding confidence ellipsoid $\check{E}_{n,\alpha}$ for $\check{\beta}_n$, or $\check{E}_{n,\alpha} \subseteq \bar{E}_{2n,\alpha}$, at any confidence level $\alpha \in (0,1)$.

Theorem 5.1 shows that $\check{\beta}_n$ is a more precise estimator than $\bar{\beta}_{2n}$ when the condition $G_v \succeq 0$ is satisfied (i.e., G_v is positive semi-definite). This may not hold in the general case, but in Appendix C.1 we show that a weaker condition holds for *all games*: the diagonal values of G_v satisfy $(G_v)_{ii} \geq 0$ for any game v. Geometrically, this weaker condition means that the confidence ellipsoid $\bar{E}_{2n,\alpha}$ extends beyond $\check{E}_{n,\alpha}$ in the axis-aligned directions.

Figure 5.2 illustrates the result of Theorem 5.1 by showing empirical 95% confidence ellipsoids for two SHAP values. Although a comparable condition is difficult to prove for the original KernelSHAP estimator ($\hat{\beta}_n$), we find that the paired sampling approach yields a similar reduction in variance. Our experiments provide further evidence that this approach accelerates convergence for both estimators (Section 5.7).

5.5.3 Convergence Detection and Forecasting

One of KernelSHAP's practical shortcomings is its lack of guidance on the number of samples required to obtain accurate estimates. We address this problem by developing an approach for convergence detection and forecasting. Previously, we showed that unbiased KernelSHAP ($\bar{\beta}_n$) has variance that reduces at a rate $\mathcal{O}(\frac{1}{n})$ (eq. 5.10). Furthermore, its variance is simple to estimate in practice: we require only an empirical estimate $\hat{\Sigma}_{\bar{b}}$ of $\Sigma_{\bar{b}}$ (defined above), which we can calculate using an online algorithm, such as Welford's well-known approach (Welford, 1962).

We also showed that the original KernelSHAP ($\hat{\beta}_n$) is difficult to characterize, but its variance is empirically lower than the unbiased version. Understanding its variance is useful for convergence detection, so we propose an approach to approximate it. Based on the results in Figure 5.1, we may hypothesize that KernelSHAP's variance reduces at the same rate of $\mathcal{O}(\frac{1}{n})$; in Appendix C.4, we examine this by plotting the product of the variance and the number of samples over the course of estimation. We find that the product is *constant* as the sample number increases, which suggests that the $\mathcal{O}(\frac{1}{n})$ rate holds in practice. This property is difficult to prove formally, but it can be used for simple variance approximation.[6]

When running KernelSHAP, we suggest estimating the variance by selecting an intermediate value m such that $m << n$ and calculating multiple independent estimates $\hat{\beta}_m$ while accumulating samples for $\hat{\beta}_n$. For any n, we can then approximate $\text{Cov}(\hat{\beta}_n)$ as

$$\text{Cov}(\hat{\beta}_n) \approx \frac{m}{n}\text{Cov}(\hat{\beta}_m),$$

where $\text{Cov}(\hat{\beta}_m)$ is estimated empirically using the multiple independent estimates $\hat{\beta}_m$. This online approach has a negligible impact on the algorithm's run-time, and the covariance estimate can be used to provide confidence intervals for the final results.

---

[6]The original version of this work did not mention the fact that KernelSHAP is an M-estimator (Van der Vaart, 2000), which implies that $\hat{\beta}_n$ is asymptotically unbiased and Gaussian. This point was mentioned in a later publication that is not included in this thesis (Chen et al., 2022).

Whether we use the original or unbiased version of KernelSHAP, the estimator's covariance at a given value of n lets us both detect and forecast convergence. For convergence detection, we propose stopping at the current value n when the largest standard deviation is a sufficiently small portion t (e.g., $t = 0.01$) of the gap between the largest and smallest Shapley value estimates. For unbiased KernelSHAP, this criterion is equivalent to:

$$\max_i \sqrt{\frac{1}{n}(\hat{\Sigma}_{\bar{\beta}})_{ii}} < t\big(\max_i\,(\bar{\beta}_n)_i - \min_i\,(\bar{\beta}_n)_i\big).$$

To forecast the number of samples required to reach convergence, we again invoke the property of the estimator's variance reducing at a rate of $\mathcal{O}(\frac{1}{n})$. Given a threshold value t and estimates $\bar{\beta}_n$ and $\hat{\Sigma}_{\bar{\beta}}$, the approximate number of samples $\hat{N}$ required is:

$$\hat{N} = \frac{1}{t^2}\Big(\frac{\max_i\,\sqrt{(\hat{\Sigma}_{\bar{\beta}})_{ii}}}{\max_i\,(\bar{\beta}_n)_i - \min_i\,(\bar{\beta}_n)_i}\Big)^2.$$

This allows us to forecast the time until convergence at any point during the algorithm. The forecast is expected to become more accurate as the estimated terms become more precise. Our approach is theoretically grounded for unbiased KernelSHAP, and the approximate approach for the standard version of KernelSHAP relies only on the assumption that $\mathrm{Cov}(\hat{\beta}_n)$ reduces at a rate of $\mathcal{O}(\frac{1}{n})$.

5.6 Stochastic Cooperative Games

We have focused so far on developing a regression-based approach to estimate Shapley values for any cooperative game. We now discuss how to adapt this approach to *stochastic cooperative games*, which leads to efficient estimators for two global explanation methods.

5.6.1 Stochastic Cooperative Games

Stochastic cooperative games return a random value for each coalition of participating players $S \subseteq [d]$. Such games are represented by a function V that maps coalitions to a *distribution*

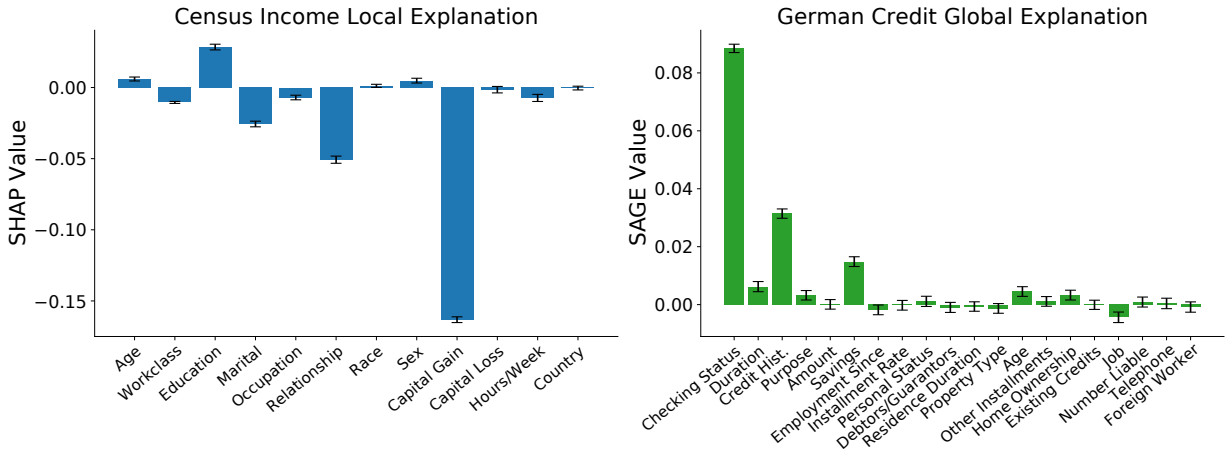


Figure 5.3: Shapley value-based explanations with 95% uncertainty estimates. Left: SHAP values for a single prediction with the census income dataset. Right: SAGE values for the German credit dataset.

of possible outcomes, so that $V(S)$ is a random variable (Charnes and Granot, 1973, 1976). To aid our presentation, we assume that the uncertainty in the game can be represented by an exogenous random variable $\boldsymbol{u}$. The game can then be denoted by $V(S, \boldsymbol{u})$, where $V(\cdot, u)$ is a deterministic function of S for any fixed value u.

Stochastic cooperative games provide a useful tool for understanding two global explanation methods, SAGE (Covert et al., 2020) and Shapley Effects (Owen, 2014). To see why, assume an exogenous variable $\boldsymbol{u} = (\boldsymbol{x}, \boldsymbol{y})$ that represents a random input-response pair, and consider the following game:

$$W(S, \boldsymbol{x}, \boldsymbol{y}) = -\ell\big(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S], \boldsymbol{y}\big). \tag{5.15}$$

The game W evaluates the (negated) loss with respect to the label $\boldsymbol{y}$ given a prediction that depends only on the features $\boldsymbol{x}_S$. The cooperative game used by SAGE can be understood as the expectation of this game, or $w(S) = \mathbb{E}[W(S, \boldsymbol{x}, \boldsymbol{y})]$ (see eq. 5.2). Shapley Effects is based on the expectation of a similar game, where the loss is evaluated with respect to the

full model prediction $f(\boldsymbol{x})$ (see Appendix C.1). As we show next, an approximation tailored to this setting yields significantly faster estimators for these methods.

5.6.2 Generalizing the Shapley Value

It is natural to assign values to players in stochastic cooperative games like we do for deterministic games. We propose a simple generalization of the Shapley value for games $V(S, \boldsymbol{u})$ that averages a player's marginal contributions over both (i) player orderings and (ii) values of the exogenous variable $\boldsymbol{u}$:

$$\phi_i(V) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \mathop{\mathbb{E}}_{p(\boldsymbol{u})} \Big[V(S \cup i, \boldsymbol{u}) - V(S, \boldsymbol{u}) \Big].$$

Due to the linearity property of Shapley values (Shapley, 1953; Monderer et al., 2002), the following values are all equivalent:

1. The Shapley values of the game's expectation $v(S) = \mathbb{E}_{\boldsymbol{u}}[V(S, \boldsymbol{u})]$, or $\phi_i(v)$.

2. The expected Shapley values of games with fixed u, or $\mathbb{E}_{\boldsymbol{u}}[\phi_i(v_{\boldsymbol{u}})]$ where $v_u(S) = V(S, u)/$

3. Our generalization of Shapley values to the stochastic cooperative game $V(S, \boldsymbol{u})$, or $\phi_i(V)$.

The first two items suggest ways of calculating the values $\phi_i(V)$ using tools designed for deterministic cooperative games. However, the expectation $\mathbb{E}_{\boldsymbol{u}}[V(S, \boldsymbol{u})]$ may be slow to evaluate (e.g., if it is across an entire dataset), and calculating Shapley values separately for each value of u would be intractable if $\boldsymbol{u}$ has many possible values. We therefore introduce a third approach.

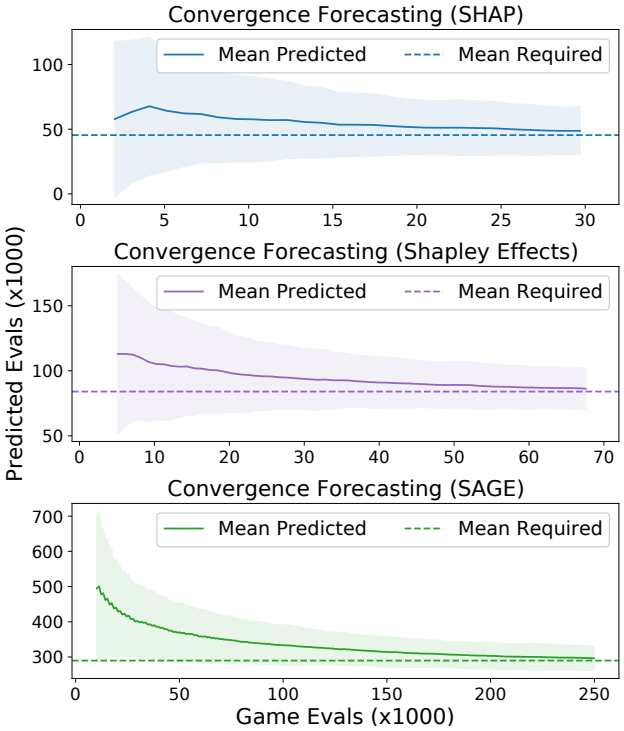


Figure 5.4: Convergence forecasting for SHAP, Shapley Effects and SAGE. The required number of samples is compared with the predicted number across 100 runs (with 90% confidence intervals displayed).

5.6.3 Shapley Value Approximation for Stochastic Cooperative Games

We now propose a fast, regression-based approach for calculating the generalized Shapley values $\phi_i(V)$ of stochastic cooperative games $V(S, \boldsymbol{u})$. Fortunately, it requires only a simple modification of the estimators described in Section 5.4. First, we must calculate the values $\mathbb{E}_{\boldsymbol{u}}[V(\mathbf{1}, \boldsymbol{u})]$ and $\mathbb{E}_{\boldsymbol{u}}[V(\mathbf{0}, \boldsymbol{u})]$ for the grand coalition and the empty coalition. Next, we replace our previous b estimators ($\hat{b}_n$ and $\bar{b}_n$) with estimators that use n pairs of independent samples $z_i \sim p(\boldsymbol{z})$ and $u_i \sim p(\boldsymbol{u})$. To adapt the original KernelSHAP to this setting, we use

$$\tilde{b}_n = \frac{1}{2}\sum_{i=1}^{n} z_i\big(V(z_i, u_i) - \mathbb{E}_{\boldsymbol{u}}[V(\mathbf{0}, \boldsymbol{u})]\big).$$

We then substitute this into the KernelSHAP estimator as follows:

$$\tilde{\beta}_n = \hat{A}_n^{-1}\Big(\tilde{b}_n - \mathbf{1}\frac{\mathbf{1}^\top \hat{A}_n^{-1}\tilde{b}_n - v(\mathbf{1}) + v(\mathbf{0})}{\mathbf{1}^\top \hat{A}_n^{-1}\mathbf{1}}\Big). \tag{5.16}$$

By the same argument used in Section 5.5, this approach estimates a solution to the weighted least squares problem whose true solution is the generalized Shapley values $\phi_i(V)$. This adaptation of KernelSHAP is consistent, and the analogous version of unbiased KernelSHAP is consistent and unbiased (see Appendix C.2). These can be run with our paired sampling approach, and we can also provide uncertainty estimates and detect convergence (Section 5.5).

5.7 Experiments

We conducted experiments with four datasets to demonstrate the advantages of our Shapley value estimation approach. We used the census income dataset (Lichman et al., 2013), the Portuguese bank marketing dataset (Moro et al., 2014), the German credit dataset (Lichman et al., 2013), and a breast cancer (BRCA) subtype classification dataset (Berger et al., 2018). To avoid overfitting with the BRCA data, we analyzed a random subset of 100 out of 17,814 genes (Appendix C.3). We trained a LightGBM model (Ke et al., 2017) for the census data, CatBoost (Prokhorenkova et al., 2018) for the credit and bank data, and logistic regression for the BRCA data.

To demonstrate local and global explanations with uncertainty estimates, we show examples of SHAP (Lundberg and Lee, 2017) and SAGE (Covert et al., 2020) values generated using our estimators (Figure 5.3). Both explanations used a convergence threshold of $t = 0.01$ and display 95% confidence intervals, which are features not previously offered by KernelSHAP. We used the dataset sampling approach for both explanations, and for SAGE we used the estimator designed for stochastic cooperative games. These estimators are faster than their unbiased versions, but the results are nearly identical.

To measure run-time differences between each estimator when calculating SHAP values, we compared the number of samples required to explain 100 instances for each dataset

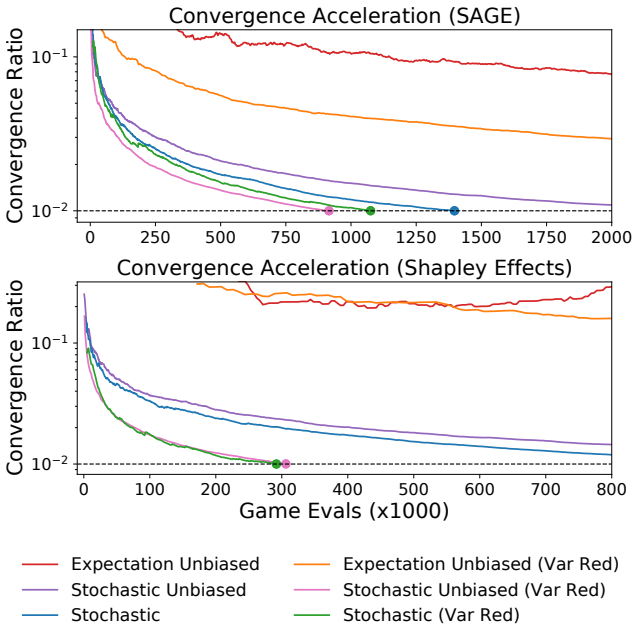


Figure 5.5: Convergence acceleration for SAGE and Shapley Effects. The ratio of the maximum standard deviation to the gap between the largest and smallest Shapley values is compared across six estimators.

(Table 5.1). Rather than reporting the exact number of samples, which is dependent on the chosen convergence threshold, we show the *ratio* between the number of samples required by each estimator; this ratio is independent of the convergence threshold when convergence is defined by the mean squared estimation error falling below a fixed value (Appendix C.4). Table 5.1 displays results based on 100 runs for each instance. The results show that the dataset sampling approach (the original version) is consistently faster than the unbiased estimator, and that paired sampling enables significantly faster convergence. In particular, we find that our paired sampling approach yields a 9× speedup on average over the original KernelSHAP.

To investigate the accuracy of our convergence forecasting method, we compared the predicted number of samples to the true number across 250 runs. The number of samples

Table 5.1: SHAP estimator run-time comparison. Each value represents the ratio of the average number of samples required relative to the fastest estimator for that dataset (lower is better).

	Census	Bank	Credit	BRCA
Unbiased	380.63	176.45	17437.44	90.40
Unbiased + Paired Sampling	128.60	90.61	422.17	40.44
Original (KernelSHAP)	12.74	7.41	13.74	2.49
Original + Paired Sampling	**1.00**	**1.00**	**1.00**	**1.00**

depends on the convergence threshold, and we used a threshold $t = 0.005$ for SHAP and $t = 0.02$ for Shapley Effects and SAGE. Figure 5.4 shows the results for SHAP (using the census data), Shapley Effects (using the bank data) and SAGE (using the BRCA data). In all three cases, the forecasts become more accurate with more samples, and they vary within an increasingly narrow range around the true number of required samples. There is a positive bias in the forecast, but the bias diminishes with more samples.

Finally, to demonstrate the speedup from our approach for stochastic cooperative games, we show that our stochastic estimator converges faster than a naive estimator based on the underlying game's expectation (see Section 5.6). We plotted the ratio between the maximum standard deviation and the gap between the smallest and largest values, which we used to detect convergence (using a threshold $t = 0.01$). Figure 5.5 shows that the stochastic approach dramatically speeds up both SAGE (using the BRCA data) and Shapley Effects (using the bank data), and that the paired sampling technique accelerates convergence for all estimators. The estimators based on the game's expectation are prohibitively slow and could not be run to convergence. The fastest estimators for both datasets are stochastic estimators using the paired sampling technique, and only these methods converged for both datasets in the number of samples displayed. As is the case for SHAP, the dataset sampling approach is often faster than the unbiased approach, but the latter is slightly faster for SAGE when

using paired sampling.

5.8 Discussion

This work describes several approaches for estimating Shapley values via linear regression. We first introduced an unbiased version of KernelSHAP, with properties that are simpler to analyze than the original version. We then developed techniques for detecting convergence, calculating uncertainty estimates, and reducing the variance of both the original and unbiased estimators. Finally, we adapted our approach to provide faster estimators for two global explanation methods based on stochastic cooperative games. Our work makes strides towards improving the practicality of Shapley value estimation by automatically determining the required number of samples, providing confidence intervals, and accelerating the estimation process.

More broadly, our work contributes to a mature literature on Shapley value estimation (Castro et al., 2009; Maleki et al., 2013) and to the growing field of model explanation (Owen, 2014; Štrumbelj and Kononenko, 2014; Datta et al., 2016; Lundberg and Lee, 2017; Covert et al., 2020). We focused on improving the regression-based approach to Shapley value estimation, and we leave to future work a detailed comparison with sampling-based (Štrumbelj and Kononenko, 2014; Song et al., 2016; Chen et al., 2018b; Covert et al., 2020) and model-specific approximations (Ancona et al., 2019; Lundberg et al., 2020). We also believe that certain insights from our work may be applicable to LIME, which is based on a similar dataset sampling approach; recent work has noted LIME's high variance when using an insufficient number of samples (Bansal et al., 2020), and an improved understanding of its convergence properties (Garreau and von Luxburg, 2020; Mardaoui and Garreau, 2020) may lead to approaches for automatic convergence detection and uncertainty estimation.

Chapter 6

AMORTIZED SHAPLEY VALUE ESTIMATION

6.1 Chapter Notes

This chapter presents joint work with Neil Jethani, Mukund Sudarshan (co-first authors), Su-In Lee and Rajesh Ranganath that was published in the International Conference on Learning Representations (ICLR) in 2022. The content of this chapter is a minor variant of the published version of the paper. The main contribution of this work is to introduce a method for amortized Shapley value estimation via a learned explainer model.

6.2 Introduction

With the proliferation of black-box models, Shapley values (Shapley, 1953) have emerged as a popular explanation approach due to their strong theoretical properties (Lipovetsky and Conklin, 2001; Štrumbelj and Kononenko, 2014; Datta et al., 2016; Lundberg and Lee, 2017). In practice, however, Shapley value-based explanations are known to have high computational complexity, with an exact calculation requiring an exponential number of model evaluations (Van den Broeck et al., 2021). Speed becomes a critical issue as models increase in size and dimensionality, and for the largest models in fields like computer vision and natural language processing, there is an unmet need for significantly faster Shapley value approximations that maintain high accuracy.

Recent work has addressed the computational challenges with Shapley values using two main approaches. First, many works have proposed *stochastic estimators* (Castro et al., 2009; Štrumbelj and Kononenko, 2014; Lundberg and Lee, 2017; Covert et al., 2020) that rely on sampling either feature subsets or permutations; though often consistent, these estimators require many model evaluations and impose an undesirable trade-off between run-time and

accuracy. Second, some works have proposed *model-specific approximations*, e.g., for trees (Lundberg et al., 2020) or neural networks (Shrikumar et al., 2017; Chen et al., 2018b; Ancona et al., 2019; Wang et al., 2021); while generally faster, these approaches can still require many model evaluations, often induce bias, and typically lack flexibility regarding the handling held-out features—a subject of ongoing debate in the field (Aas et al., 2019; Janzing et al., 2020; Frye et al., 2021; Covert et al., 2021).

Here, we introduce a new approach for efficient Shapley value estimation: to achieve the fastest possible run-time, we propose learning a separate explainer model that outputs precise Shapley value estimates in a single forward pass. Naively, such a learning-based approach would seem to require a large training set of ground truth Shapley values, which would be computationally intractable. Instead, our approach trains an explainer model by minimizing an objective function inspired by the Shapley value's weighted least squares characterization (Charnes et al., 1988), which enables efficient gradient-based optimization.

Contributions We introduce FastSHAP, an amortized approach for generating real-time Shapley value explanations.[1] We derive an objective function from the Shapley value's weighted least squares characterization and investigate several ways to reduce gradient variance during training. Our experiments show that FastSHAP provides accurate Shapley value estimates with an orders-of-magnitude speedup relative to non-amortized estimation approaches. Finally, we also find that FastSHAP generates high-quality image explanations that outperform gradient-based methods (e.g., IntGrad and GradCAM) on standard inclusion and exclusion metrics.

6.3 Background

In this section, we briefly review Shapley values and their weighted least squares characterization. We focus here on a classification model f trained to predict a response variable $\boldsymbol{y} \in [K]$ based on an input variable $\boldsymbol{x}$ that consists of d separate features, or $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$. We use

[1] `https://github.com/iancovert/fastshap/`

$\mathbf{1} \in \mathbb{R}^d$ to denote a vector of ones, predictions lie in the probability simplex $f(\boldsymbol{x}) \in \Delta^{K-1}$, and $f_y(\boldsymbol{x}) \in [0, 1]$ is the probability for the yth class.

6.3.1 Shapley Values

As described in Chapter 5, Shapley values are a credit allocation technique from cooperative game theory (Shapley, 1953), but they have been adopted to explain predictions from black-box machine learning models (Štrumbelj and Kononenko, 2014; Datta et al., 2016; Lundberg and Lee, 2017). For any coalitional game (or set function) $v : \mathcal{P}(d) \mapsto \mathbb{R}$, the Shapley values $\phi(v) \in \mathbb{R}^d$, or $\phi_i(v) \in \mathbb{R}$ for each feature $i \in [d]$, are given by the formula

$$\phi_i(v) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \big(v(S \cup i) - v(S)\big). \tag{6.1}$$

The difference $v(S \cup i) - v(S)$ represents the ith feature's contribution to the subset S, and the summation represents a weighted average across all subsets that do not include i. In the model explanation context, the coalitional game is chosen to represent how an individual prediction varies as different subsets of features are removed. For example, given an input-output pair (x, y), the prediction for the yth class can be represented by a coalitional game v_{xy} defined as

$$v_{xy}(S) = \mathop{\mathbb{E}}_{p(\boldsymbol{x}_{\bar{S}})} \left[f_y \left(x_S, \boldsymbol{x}_{\bar{S}} \right) \right], \tag{6.2}$$

where the held-out features $\boldsymbol{x}_{\bar{S}}$ are marginalized out using their joint marginal distribution $p(\boldsymbol{x}_{\bar{S}})$. Recent work has debated the properties of different coalitional game formulations, particularly the choice of how to remove features (Aas et al., 2019; Janzing et al., 2020; Frye et al., 2021; Covert et al., 2021).[2] However, regardless of the formulation, this approach to model explanation enjoys several useful theoretical properties due to its use of Shapley values: for example, the attributions are zero for irrelevant features, and they are guaranteed to sum to the model's prediction. We direct readers to Chapter 2 or prior work (Lundberg

[2]See Chapter 3 for a detailed discussion.

and Lee, 2017; Covert et al., 2021) for a detailed discussion of these properties.

Unfortunately, Shapley values also introduce computational challenges: the summation in eq. 6.1 involves an exponential number of subsets, which makes it infeasible to calculate for large d. Fast approximations are therefore required in practice, as we discuss next.

6.3.2 KernelSHAP

As described in detail in Chapter 5, KernelSHAP (Lundberg and Lee, 2017) is a popular Shapley value implementation that relies on an alternative mathematical interpretation of the Shapley value. Given a coalitional game $v_{xy}(S)$, eq. 6.1 shows that the values $\phi(v_{xy})$ are the features' weighted average contributions; equivalently, their weighted least squares characterization shows that they are the solution to an optimization problem over $\phi_{xy} \in \mathbb{R}^d$,

$$\phi(v_{xy}) = \underset{\phi_{xy}}{\arg\min} \underset{p(\boldsymbol{S})}{\mathbb{E}} \left[\left(v_{xy}(\boldsymbol{S}) - v_{xy}(\varnothing) - \boldsymbol{S}^\top \phi_{xy} \right)^2 \right] \tag{6.3}$$

$$\text{s.t.} \quad \mathbf{1}^\top \phi_{xy} = v_{xy}([d]) - v_{xy}(\varnothing), \qquad \text{(Efficiency constraint)}$$

where the inner product with $\boldsymbol{S} \subseteq [d]$ represents the sum of the corresponding vector entries, and the distribution $p(\boldsymbol{S})$ is defined as

$$p(S) \propto \frac{d-1}{\binom{d}{|S|} \cdot |S| \cdot (d - |S|)} \qquad \text{(Shapley kernel)}$$

for S such that $0 < |S| < d$ (Charnes et al., 1988). Based on this view of the Shapley value, Lundberg and Lee (2017) introduced KernelSHAP, a stochastic estimator that solves an approximate version of eq. 6.3 given a moderate number of subsets sampled from $p(\boldsymbol{S})$. Although the estimator is consistent and empirically unbiased (Covert and Lee, 2021), KernelSHAP often requires many samples to achieve an accurate estimate, and it must solve eq. 6.3 separately for each input-output pair (x, y). As a result, it is unacceptably slow for some use cases, particularly in settings with large, high-dimensional models. Our approach builds on KernelSHAP, leveraging the Shapley value's weighted least squares characterization

to design a faster, amortized estimation approach.

6.4 FastSHAP

We now introduce FastSHAP, a method that amortizes the cost of generating Shapley values across many data examples. FastSHAP has two main advantages over existing approaches: (i) it avoids solving separate optimization problems for each input to be explained, and (ii) it can use similar data points to efficiently learn the Shapley value function $\phi(v_{xy})$.

6.4.1 Amortizing Shapley Values

In our approach, we propose generating Shapley value explanations using a learned parametric function $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta) \in \mathbb{R}^d$. Once trained, the parametric function can generate explanations in a single forward pass, providing a significant speedup over methods that approximate Shapley values separately for each sample (x, y). Rather than using a dataset of ground truth Shapley values for training, we train $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ by penalizing its predictions according to the weighted least squares objective in eq. 6.3, or by minimizing the following objective function:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \left[\left(v_{\boldsymbol{xy}}(\boldsymbol{S}) - v_{\boldsymbol{xy}}(\varnothing) - \boldsymbol{S}^\top \phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta) \right)^2 \right]. \tag{6.4}$$

We use this loss function because if the model's predictions are forced to satisfy the Efficiency constraint, then given a large enough dataset and a sufficiently expressive model class, the global optimizer $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta^*)$ is a function that outputs exact Shapley values (see proof in Appendix D.1). Formally, the global optimizer satisfies the following:

$$\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta^*) = \phi(v_{\boldsymbol{xy}}) \;\text{ almost surely in }\; p(\boldsymbol{x}, \boldsymbol{y}). \tag{6.5}$$

We explore two approaches to address this constraint in practice. First, we can enforce efficiency by adjusting the Shapley value predictions using their *additive efficient normalization*

(Ruiz et al., 1998), which applies the following operation to the model's outputs:

$$\phi_{\text{fast}}(x, y; \theta) \leftarrow \phi_{\text{fast}}(x, y; \theta) + \frac{1}{d} \underbrace{\left(v_{xy}([d]) - v_{xy}(\varnothing) - \mathbf{1}^\top \phi_{\text{fast}}(x, y; \theta) \right)}_{\text{Efficiency gap}}. \tag{6.6}$$

The normalization step can be applied at inference time and optionally during training; in Appendix D.2, we show that this step is guaranteed to make the estimates closer to the true Shapley values. Second, we can relax the efficiency property by augmenting $\mathcal{L}(\theta)$ with a penalty on the efficiency gap (see eq. 6.6); the penalty requires a parameter $\gamma > 0$, and as we set $\gamma \to \infty$ we can expect efficiency to hold for the trained model (see Appendix D.1). Algorithm 1 summarizes our training approach.

Empirical considerations Optimizing $\mathcal{L}(\theta)$ using a single set of samples (x, y, S) is problematic because of high variance in the gradients, which can lead to poor optimization. We therefore consider several steps to reduce gradient variance. First, as is conventional in deep learning, we minibatch across multiple samples from $p(\boldsymbol{x})$. Next, when possible, we calculate the loss jointly across all classes $y \in [K]$. Then, we experiment with using multiple samples $S \sim p(\boldsymbol{S})$ for each input sample x. Finally, we explore *paired sampling*, where each sample S is paired with its complement $\bar{S}$, which has been shown to reduce KernelSHAP's variance (Covert and Lee, 2021). Appendix D.3 shows proofs that these steps are guaranteed to reduce gradient variance, and ablation experiments in Appendix D.4 demonstrate their improvement on FastSHAP's accuracy.

6.4.2 A Default Coalitional Game for FastSHAP

FastSHAP has the flexibility to work with any coalitional game formulation $v_{xy}(S)$. Here, we describe a default formulation that is useful for explaining predictions from a classification model.

The coalitional game's aim is to assess, for each subset S, the classification probability when only the features $\boldsymbol{x}_S$ are observed. Because most models $f(\boldsymbol{x})$ do not support making

Algorithm 1: FastSHAP training

Input: Coalitional game v_{xy}, learning rate α
Output: FastSHAP explainer $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$
initialize $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$
while *not converged* **do**
  sample $x, y \sim p(\boldsymbol{x}, \boldsymbol{y})$, $S \sim p(\boldsymbol{S})$
  predict $\hat{\phi} \leftarrow \phi_{\text{fast}}(x, y; \theta)$
  if *normalize* **then**
    set $\hat{\phi} \leftarrow \hat{\phi} + d^{-1}\left(v_{xy}([d]) - v_{xy}(\varnothing) - \mathbf{1}^\top \hat{\phi}\right)$
  end
  calculate $\mathcal{L} \leftarrow \left(v_{xy}(S) - v_{xy}(\varnothing) - S^\top \hat{\phi}\right)^2$
  update $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$
end

predictions without all the features, we require an approximation that simulates the inclusion of only $\boldsymbol{x}_S$ (Covert et al., 2021). To this end, we use a supervised surrogate model (Frye et al., 2021; Jethani et al., 2021a) to approximate marginalizing out the remaining features $\boldsymbol{x}_{\bar{S}}$ using their conditional distribution.

Separate from the original model $f(\boldsymbol{x})$, the *surrogate* model $g(\boldsymbol{x}_S; \beta) \in \Delta^{K-1}$ takes as input a vector of masked features x_S, where the mask replaces features x_i such that $i \notin S$ with a mask value that is not in the support of $\mathcal{X}$. Similar to prior work (Frye et al., 2021; Jethani et al., 2021a), the parameters β are learned by minimizing the following loss function:

$$\mathcal{L}(\beta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \left[D_{\text{KL}}\big(f(\boldsymbol{x}) \,||\, g(\boldsymbol{x}_{\boldsymbol{S}}; \beta)\big)\right]. \tag{6.7}$$

It has been shown that the global optimizer to eq. 6.7, or $g(\boldsymbol{x}_S; \beta^*)$, is equivalent to marginalizing out features from $f(\boldsymbol{x})$ with their conditional distribution (Covert et al., 2021):

$$g(x_S; \beta^*) = \mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S = x_S]. \tag{6.8}$$

The choice of distribution over subsets does not affect the global optimizer of eq. 6.7, but we

use a simple approach of sampling the cardinality uniformly at random, and then sampling the set members at random. We use the surrogate model as a default choice for two reasons. First, it requires a single prediction for each evaluation of $v_{xy}(S)$, which permits faster training than the common approach of averaging across many background samples (Lundberg and Lee, 2017; Janzing et al., 2020). Second, it yields explanations that reflect the model's dependence on the *information* communicated by each feature, rather than its *algebraic* dependence (Frye et al., 2021; Covert et al., 2021).

6.5 Related Work

Recent work on Shapley value explanations has largely focused on how to remove features (Aas et al., 2019; Frye et al., 2021; Covert et al., 2021) and how to approximate Shapley values efficiently (Chen et al., 2018b; Ancona et al., 2019; Lundberg et al., 2020; Covert and Lee, 2021). Model-specific approximations are relatively fast, but they often introduce bias and are entangled with specific feature removal approaches (Shrikumar et al., 2017; Ancona et al., 2019; Lundberg et al., 2020). In contrast, model-agnostic stochastic approximations are more flexible, but they trade off run-time and accuracy in the explanation. For example, KernelSHAP samples subsets to approximate the solution to a weighted least squares problem (Lundberg and Lee, 2017), while other approaches sample marginal contributions (Castro et al., 2009; Štrumbelj and Kononenko, 2014) or feature permutations (Illés and Kerényi, 2019; Mitchell et al., 2021). FastSHAP trains an explainer model to output an estimate that would otherwise require orders of magnitude more model evaluations, and, unlike other fast approximations, it is agnostic to the model class and feature removal approach.

Other methods have been proposed to generate explanations using learned explainer models. These are referred to as *amortized explanation methods* (Covert et al., 2021; Jethani et al., 2021a), and they include several approaches that are comparable to gradient-based methods in terms of compute time (Dabkowski and Gal, 2017; Chen et al., 2018a; Yoon et al., 2018; Schwab and Karlen, 2019; Schulz et al., 2020; Jethani et al., 2021a). Notably, one approach generates a training dataset of ground truth explanations and then learns an

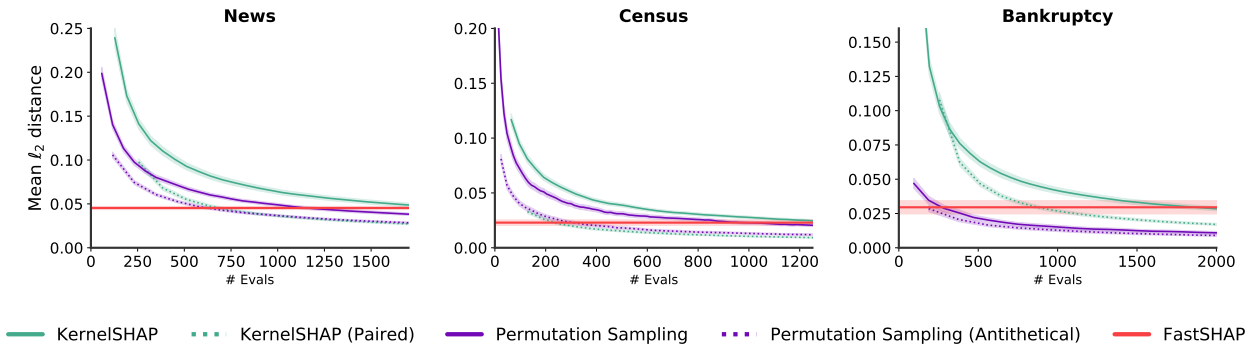


Figure 6.1: Comparison of Shapley value approximation accuracy across methods. Using three datasets, we measure the distance of each method's estimates to the ground truth as a function of the number of model evaluations. FastSHAP is represented by a horizontal line since it requires only a single forward pass. The baselines require 200−2000× model evaluations to achieve FastSHAP's level of accuracy.

explainer model to output explanations directly (Schwab and Karlen, 2019)—a principle that can be applied with any attribution method, at least in theory. However, for Shapley values, generating a large training set would be very costly, so FastSHAP sidesteps the need for a training set using a custom loss function based on the Shapley value's weighted least squares characterization (Charnes et al., 1988).

6.6 Experiments

We analyze FastSHAP's performance by comparing it to several well-known baselines. We first evaluate its accuracy on tabular datasets by comparing its outputs to the ground truth Shapley values. Then, to disentangle the benefits of amortization from the choice of coalitional game, we make the same comparisons using different formulations for $v_{xy}(\boldsymbol{S})$. Unless otherwise stated, we use the surrogate model formulation introduced in Section 6.4. Later, we test FastSHAP's ability to generate image explanations.

6.6.1 Tabular Datasets

To contextualize FastSHAP's accuracy, we compare it to several non-amortized stochastic estimators. First, we compare to KernelSHAP (Lundberg and Lee, 2017) and its accelerated version that uses paired sampling (Covert and Lee, 2021). Next, we compare to a permutation sampling approach and its accelerated version that uses antithetical sampling (Mitchell et al., 2021). As a performance metric, we calculate the proximity to Shapley values that were obtained by running KernelSHAP to convergence; we use these values as our ground truth because KernelSHAP is known to converge to the true Shapley values given infinite samples (Covert and Lee, 2021). These baselines were all run using an open-source implementation.[3]

We use either neural networks or tree-based models for each of $f(\boldsymbol{x})$ and $g(\boldsymbol{x}_S; \beta)$. The FastSHAP explainer model $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ is implemented via a network that outputs a vector of Shapley values for every $y \in [K]$; deep neural networks are ideal for FastSHAP because they have high representation capacity, they can provide many-to-many mappings, and they can be trained by stochastic gradient descent. Appendix D.4 contains more details about our implementation, including model classes, network architectures and training hyperparameters.

We also perform a series of experiments to determine several training hyperparameters for FastSHAP, exploring (i) whether or not to use paired sampling, (ii) the number of subset samples to use, and (iii) how to best enforce the efficiency constraint. Based on the results (see Appendix D.4), we use the following settings for our tabular data experiments: we use paired sampling, between 32 and 64 samples of $\boldsymbol{S}$ per $\boldsymbol{x}$ sample, additive efficient normalization during both training and inference, and we set $\gamma = 0$ (because the normalization step is sufficient to enforce efficiency).

Accuracy of FastSHAP explanations Here, we test whether FastSHAP's estimates are close to the ground truth Shapley values. Our experiments use data from a 1994 United

[3] `https://github.com/iancovert/shapley-regression/`

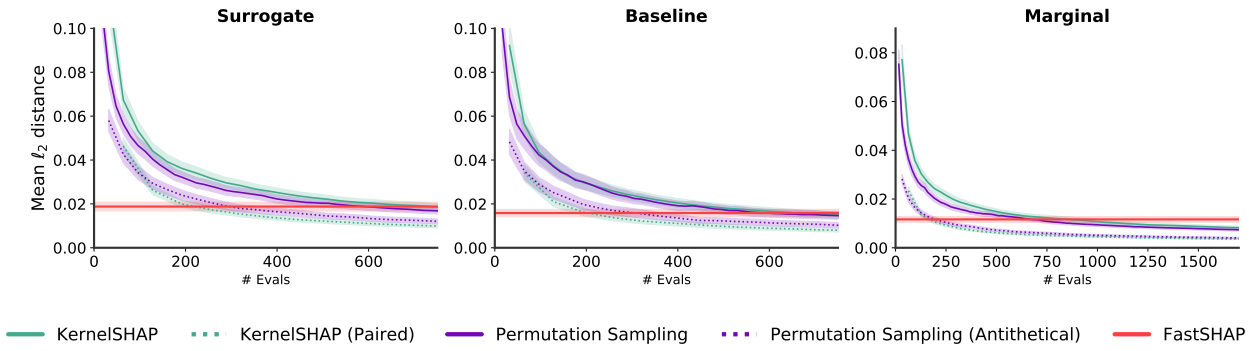


Figure 6.2: FastSHAP approximation accuracy for different coalitional games. Using the marketing dataset, we find that FastSHAP provides accurate Shapley value estimates regardless of the formulation (surrogate, marginal, baseline), with the baselines requiring $200-1000\times$ model evaluations to achieve FastSHAP's level of accuracy. Error bars represent 95% confidence intervals.

States census, a bank marketing campaign, bankruptcy statistics, and online news articles (Dua and Graff, 2017). The census data contains 12 input features, and the binary label indicates whether a person makes over \$50K a year (Kohavi et al., 1996). The marketing dataset contains 17 input features, and the label indicates whether the customer subscribed to a term deposit (Moro et al., 2014). The bankruptcy dataset contains 96 features describing various companies and whether they went bankrupt (Liang et al., 2016). The news dataset contains 60 numerical features about articles published on Mashable, and our label indicates whether the share count exceeds the median number (1,400) (Fernandes et al., 2015). The datasets were each split 80/10/10 for training, validation and testing.

In Figure 6.1, we show the distance of each method's estimates to the ground truth as a function of the number of model evaluations for the news, census and bankruptcy datasets. Figure 6.2 shows results for the marketing dataset with three different coalitional games (see more details below). For the baselines, each sample $\boldsymbol{S}$ requires evaluating the model given a subset of features, but since FastSHAP requires only a single forward pass of $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$, we show it as a horizontal line.

To reach FastSHAP's level of accuracy on the news, census and bankruptcy datasets,

KernelSHAP requires between 1,200−2,000 model evaluations. Like prior work (Covert and Lee, 2021), we find that paired sampling improves KernelSHAP's rate of convergence, helping reach FastSHAP's accuracy in 250−1,000 model evaluations. The permutation sampling baselines tend to be faster: the original version requires between 300−1,000 evaluations, and antithetical sampling takes 200−500 evaluations to reach an accuracy equivalent to FastSHAP. Across all four datasets, however, FastSHAP achieves its level of accuracy at least at least 600× faster than the original KernelSHAP, and 200× faster than the best non-amortized baseline.

Disentangling the choice of coalitional game In this experiment, we verify that FastSHAP produces accurate Shapley value estimates regardless of the choice of coalitional game. We use the marketing dataset for this experiment and test the following coalitional games:

1. (Surrogate/In-distribution) $v_{xy}(S) = g_y(x_S; \beta)$, which aims to marginalize out features using their conditional distribution (see Section 6.4)

2. (Marginal/Out-of-distribution) $v_{xy}(S) = \mathbb{E}_{p(\boldsymbol{x}_{\bar{S}})}\left[f_y\left(x_S, \boldsymbol{x}_{\bar{S}}\right)\right]$, which is a common alternative (Lundberg and Lee, 2017)

3. (Baseline removal) $v_{xy}(S) = f_y(x_S, b_{\bar{S}})$, where $b \in \mathcal{X}$ are fixed baseline values (the mean for continuous features and mode for discrete ones).

In Figure 6.2 we compare FastSHAP to the same non-amortized baseline methods, where each method generates Shapley value estimates using the coalitional games listed above. The results show that FastSHAP maintains the same computational advantage across all three cases: to achieve the same accuracy as FastSHAP's single forward pass, the baseline methods require at least 200 model evaluations, and in some cases up to nearly 1,000.

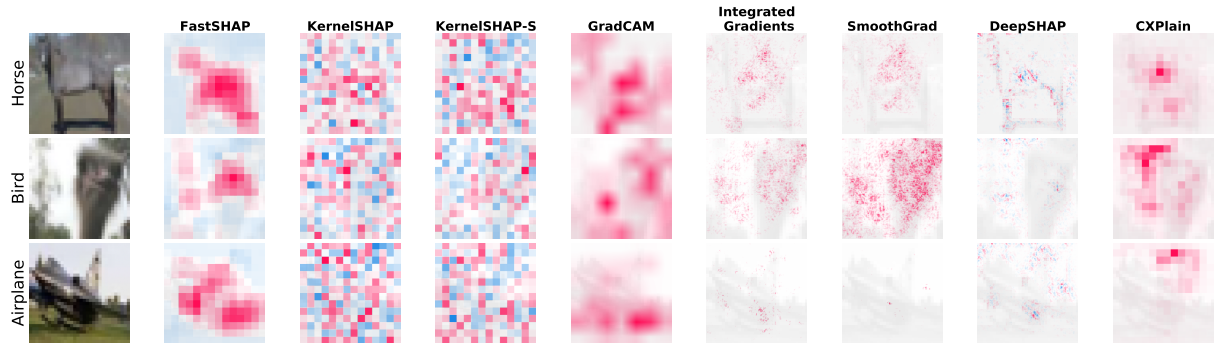


Figure 6.3: Explanations generated by each method for CIFAR-10 images.

6.6.2 Image Experiments

Images represent a challenging setting for Shapley values due to their high dimensionality and the computational cost of model evaluation. We therefore compare FastSHAP to KernelSHAP on two image datasets. We also consider several widely used gradient-based explanation methods, because they are the most commonly used methods for explaining image classifiers.

We consider two popular image datasets for our experiments. CIFAR-10 (Krizhevsky et al., 2009) contains 60,000 32×32 images across 10 classes, and we use 50,000 samples for training and 5,000 samples each for validation and testing. Each image is resized to 224×224 using bilinear interpolation to interface with the ResNet-50 architecture (He et al., 2016b). Figure 6.3 shows example CIFAR-10 explanations generated by each method. The ImageNette dataset (Howard and Gugger, 2020), a subset of 10 classes from the ImageNet dataset, contains 13,394 total images. Each image is cropped to keep the 224×224 central region, and the data is split 9,469/1,963/1,962. Example ImageNette explanations are shown in Figure 6.4.

We test three Shapley value estimators, FastSHAP, KernelSHAP, and DeepSHAP (Lundberg and Lee, 2017), where the last is an existing approximation designed for neural networks. We test KernelSHAP with the zeros baseline coalitional game, which we refer to

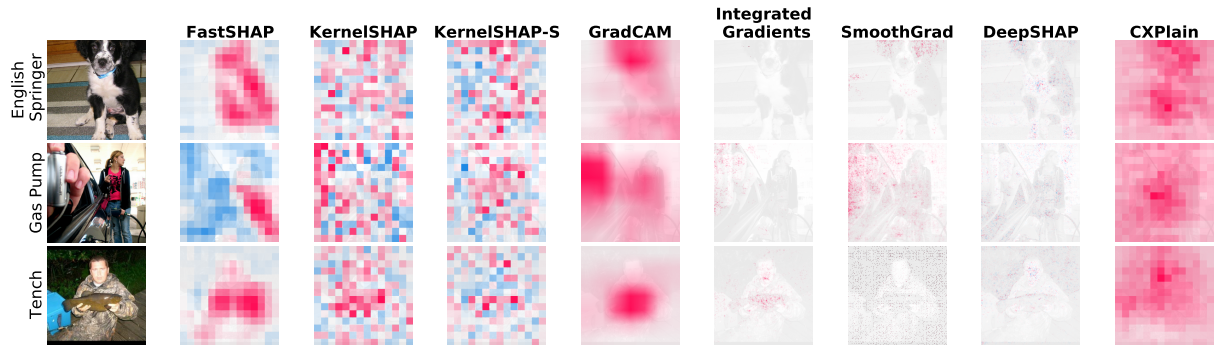


Figure 6.4: Explanations generated by each method for ImageNette images.

simply as KernelSHAP, and with the in-distribution surrogate formulation, which we refer to as KernelSHAP-S. We also compare these methods to the gradient-based explanation methods GradCAM (Selvaraju et al., 2017), SmoothGrad (Smilkov et al., 2017) and IntGrad (Sundararajan et al., 2017). Gradient-based methods are relatively fast and have therefore been widely adopted for explaining image classifiers. Finally, we also compare to CXPlain (Schwab and Karlen, 2019), an amortized explanation method that generates attributions that are not based on Shapley values.

The models $f(\boldsymbol{x})$ and $g(\boldsymbol{x}_S; \beta)$ are both ResNet-50 networks (He et al., 2016b) pre-trained on ImageNet and fine-tuned on the corresponding imaging dataset. FastSHAP, CXPlain, and KernelSHAP are all implemented to output 14×14 superpixel attributions for each class. For FastSHAP, we parameterize $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ to output superpixel attributions: we use an identical pre-trained ResNet-50 but replace the final layers with a 1×1 convolutional layer so that the output is $14 \times 14 \times K$ (see details in Appendix D.4). We use an identical network to produce attributions for CXPlain. For FastSHAP, we do not use additive efficient normalization, and we set $\gamma = 0$; we find that this relaxation of the Shapley value's efficiency property does not inhibit FastSHAP's ability to produce high-quality image explanations. KernelSHAP and KernelSHAP-S are implemented using the SHAP package's default param-

eters,[4] and GradCAM, SmoothGrad, and IntGrad are implemented using the `tf-explain` package's default parameters.[5]

Qualitative remarks Explanations generated by each method are shown in Figure 6.3 for CIFAR-10 and Figure 6.4 for ImageNette (see Appendix D.5 for more examples). While a qualitative evaluation is insufficient to draw conclusions about each method, we can offer several remarks on these examples. FastSHAP, and to some extent GradCAM, appear to reliably highlight the important objects, while the KernelSHAP explanations are noisy and fail to localize important regions. To a lesser extent, CXPlain occasionally highlights important regions. In comparison, the remaining methods (SmoothGrad, IntGrad and DeepSHAP) are granulated and highlight only small parts of the key objects. Next, we consider quantitative metrics that test these observations more systematically.

Metrics Evaluating the quality of Shapley value estimates requires access to ground truth Shapley values, which is computationally costly for images. Instead, we use two metrics that evaluate an explanation's ability to identify informative image regions. These metrics build on several recent proposals (Petsiuk et al., 2018; Hooker et al., 2019; Jethani et al., 2021a) and evaluate the model's classification accuracy after including or excluding pixels according to their estimated importance (sometimes referred to as *insertion* and *deletion* scores, see Chapter 7).

Similar to Jethani et al. (2021a), we begin by training a single evaluation model to approximate the original model's output given a subset of features; this serves as an alternative to training separate models on each set of features (Hooker et al., 2019) and offers a more realistic option than masking features with zeros (Schwab and Karlen, 2019). This procedure is analogous to the surrogate training procedure in Section 6.4.

Next, we analyze how the model's predictions change as we remove either important or

[4] `https://shap.readthedocs.io/en/latest/`

[5] `https://tf-explain.readthedocs.io/en/latest/`

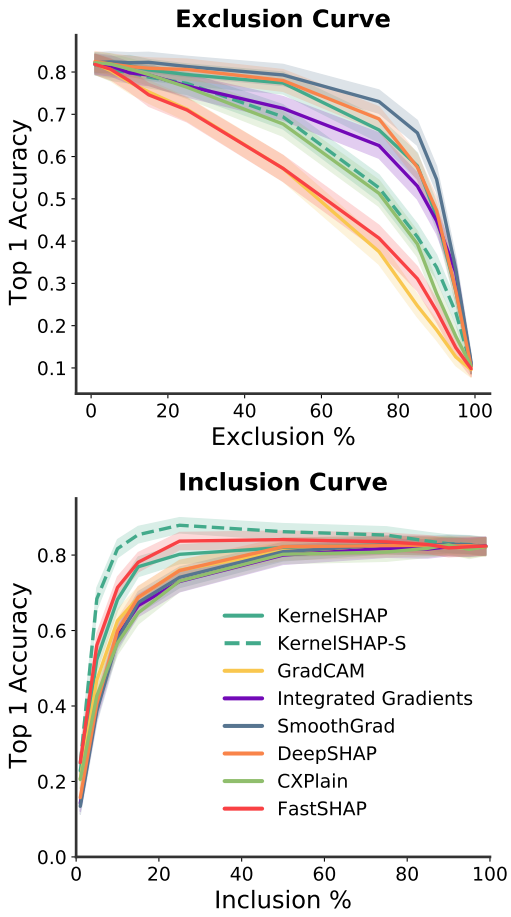


Figure 6.5: ImageNette inclusion and exclusion curves. The change in top-1 accuracy as an increasing percentage of the pixels estimated to be important are excluded (top) or included (bottom).

unimportant features according to each explanation. Using a set of 1,000 images, each image is first labeled by the original model $f(\boldsymbol{x})$ using the most likely predicted class. We then use explanations generated by each method to produce feature rankings and compute the top-1 accuracy (a measure of agreement with the original model) as we either include or exclude the most important features, ranging from $0-100\%$. The area under each curve (AUC) is termed the *Inclusion AUC* or *Exclusion AUC*.

These metrics match the idea that an accurate image explanation should (i) maximally degrade the performance when important features are excluded, and (ii) maximally improve

Table 6.1: Exclusion and Inclusion AUCs. Evaluation of each method on the basis of Exclusion AUC (lower is better) and Inclusion AUC (higher is better) calculated using top-1 accuracy. Parentheses indicate 95% confidence intervals, and the best methods are bolded in each column.

	CIFAR-10		ImageNette	
	Exclusion AUC	Inclusion AUC	Exclusion AUC	Inclusion AUC
FastSHAP	**0.42 (0.41, 0.43)**	0.78 (0.77, 0.79)	**0.51 (0.49, 0.52)**	**0.79 (0.78, 0.80)**
KernelSHAP	0.64 (0.63, 0.65)	0.78 (0.77, 0.79)	0.68 (0.67, 0.70)	0.77 (0.75, 0.78)
KernelSHAP-S	0.54 (0.52, 0.55)	**0.86 (0.85, 0.87)**	0.61 (0.60, 0.62)	**0.82 (0.80, 0.83)**
GradCAM	0.52 (0.51, 0.53)	0.76 (0.75, 0.77)	**0.52 (0.50, 0.53)**	0.74 (0.73, 0.76)
Integrated Gradients	0.55 (0.54, 0.56)	0.74 (0.73, 0.75)	0.65 (0.64, 0.67)	0.73 (0.71, 0.74)
SmoothGrad	0.70 (0.69, 0.71)	0.72 (0.71, 0.73)	0.72 (0.71, 0.73)	0.73 (0.72, 0.75)
DeepSHAP	0.65 (0.64, 0.66)	0.79 (0.78, 0.80)	0.69 (0.68, 0.71)	0.74 (0.73, 0.75)
CXPlain	0.56 (0.55, 0.57)	0.71 (0.70, 0.72)	0.60 (0.58, 0.61)	0.72 (0.71, 0.74)

the performance when important features are included (Petsiuk et al., 2018; Hooker et al., 2019). The explanations are evaluated by removing superpixels; for gradient-based methods, we coarsen the explanations using the sum total importance within each superpixel. In Appendix D.5, we replicate these metrics using log-odds rather than top-1 accuracy, finding a similar ordering among methods.

Table 6.1 shows the Inclusion and Exclusion AUC achieved by each method for both CIFAR-10 and ImageNette. In Figure 6.5, we also present the curves used to generate these AUCs for ImageNette. Lower Exclusion AUCs and higher Inclusion AUCs are better. These results show that FastSHAP outperforms all baseline methods when evaluated with Exclusion AUC: when the pixels identified as important by FastSHAP are removed from the images, the sharpest decline in top-1 accuracy is observed. Additionally, FastSHAP performs well when evaluated on the basis of Inclusion AUC, second only to KernelSHAP-S.

For ImageNette, GradCAM performs competitively with FastSHAP on Exclusion AUC and KernelSHAP-S marginally beats FastSHAP on Inclusion AUC. KernelSHAP-S also outperforms on Inclusion AUC with CIFAR-10, which is perhaps surprising given its high level of noise (Figure 6.3). However, KernelSHAP-S does not do as well when evaluated using

Table 6.2: Training and explanation run-times for 1,000 images (in minutes).

		CIFAR-10	ImageNette
Explain	FastSHAP	**0.04**	**0.04**
	KernelSHAP	453.69	1089.50
	KernelSHAP-S	460.10	586.12
	GradCAM	0.38	0.30
	IntGrad	0.91	0.92
	SmoothGrad	1.00	1.05
	DeepSHAP	5.39	6.01
	CXPlain	**0.04**	**0.04**
Train	FastSHAP	693.57	146.49
	KernelSHAP-S	362.03	73.22
	CXPlain	538.49	93.00

Exclusion AUC, and GradCAM does not do as well on Inclusion AUC. The remaining methods are in most cases not competitive on either metric (except DeepSHAP on CIFAR-10 Inclusion AUC). An accurate explanation should perform well on both metrics, so these results show that FastSHAP provides the most versatile explanations, because it is the only approach to excel at both Inclusion and Exclusion AUC.

Finally, we also test FastSHAP's robustness to limited training data. In Appendix D.5, we find that FastSHAP outperforms most baseline methods on Inclusion and Exclusion AUC when using just 25% of the ImageNette data, and that it remains competitive even when using just 10%.

Speed evaluation The image experiments were run using 8 cores of an Intel Xeon Gold 6148 processor and a single NVIDIA Tesla V100. Table 6.2 records the time required to explain 1,000 images. For FastSHAP, KernelSHAP-S and CXPlain, we also report the time required to train the surrogate and/or explainer models. The amortized explanation methods, FastSHAP and CXPlain, incur a fixed training cost but very low marginal cost for

each explanation. The gradient-based methods are slightly slower, but KernelSHAP requires significantly more time. These results suggest that FastSHAP is well suited for real-time applications where it is crucial to keep explanation times as low as possible. Further, when users need to explain a large quantity of data, FastSHAP's low explanation cost can quickly compensate for its training time.

6.7 Conclusion

This chapter introduced FastSHAP, a method for estimating Shapley values in a single forward pass using a learned explainer model. To enable efficient training, we sidestepped the need for a training dataset and derived a learning approach from the Shapley value's weighted least squares characterization. Our experiments demonstrate that FastSHAP can produce accurate Shapley value estimates while achieving a significant speedup over non-amortized approaches, as well as more accurate image explanations than popular gradient-based methods.

While Shapley values provide a strong theoretical grounding for model explanation, they have not been widely adopted for explaining large-scale models due to their high computational cost. FastSHAP can solve this problem, making fast and high-quality explanations possible in fields like computer vision and natural language processing. By casting model explanation as a learning problem, FastSHAP stands to benefit as the state of deep learning advances, and it opens a new direction of research for efficient Shapley value estimation.

Chapter 7

LEARNING TO ESTIMATE SHAPLEY VALUES WITH VISION TRANSFORMERS

7.1 Chapter Notes

This chapter presents joint work with Chanwoo Kim (co-first author) and Su-In Lee that was published in the International Conference on Learning Representations (ICLR) in 2023. The content of this chapter is a minor variant of the published version of the paper. The main contribution of this work is to test the amortized Shapley value estimation method from Chapter 6 in the context of vision transformers, where most current methods focus on analyzing attention values.

7.2 Introduction

Transformers were originally introduced for NLP (Vaswani et al., 2017), but in recent years they have been successfully adapted to a variety of other domains (Wang et al., 2020a; Jumper et al., 2021). In computer vision, transformer-based models are now used for problems including image classification, object detection and semantic segmentation (Dosovitskiy et al., 2020; Touvron et al., 2021; Liu et al., 2021b), and they achieve state-of-the-art performance in many tasks (Wortsman et al., 2022). The growing use of transformers in computer vision motivates the question of what drives their predictions: understanding a complex model's dependencies is an important problem in many applications, but the field has not settled on a solution for the transformer architecture.

Transformers are composed of alternating self-attention and fully-connected layers, where the self-attention operation associates attention values with every pair of tokens. In vision transformers (ViTs) (Dosovitskiy et al., 2020), the tokens represent non-overlapping image

patches, typically a total of $14 \times 14 = 196$ patches each of size 16×16. It is intuitive to view attention values as indicators of feature importance (Abnar and Zuidema, 2020; Ethayarajh and Jurafsky, 2021), but interpreting transformer attention in this way is potentially misleading. Recent work has raised questions about the validity of attention as explanation (Serrano and Smith, 2019; Jain and Wallace, 2019; Chefer et al., 2021), arguing that it provides an incomplete picture of a model's dependence on each token.

If attention is not a reliable indicator of feature importance, then what is? We consider the perspective that transformers are no different from any other architecture, and that we can explain their predictions using model-agnostic approaches that are currently used for other architectures. Among these methods, Shapley values are a theoretically compelling approach with feature importance scores that are designed to satisfy many desirable properties (Shapley, 1953; Lundberg and Lee, 2017).

The main challenge for Shapley values in the transformer context is calculating them efficiently, because a naive calculation has exponential running time in the number of patches. If Shapley values are poorly approximated, they are unlikely to reflect a model's true dependencies, but calculating them with high accuracy is currently too slow to be practical. Thus, our work aims to make Shapley values practical for transformers, and for ViTs in particular. Our contributions include:

1. We investigate several approaches for withholding input features from vision transformers, which is a key operation for computing Shapley values. We find that ViTs can accommodate missing image patches by masking attention values for held-out tokens, and that training with random masking is important for models to properly handle partial information.

2. We develop a learning-based approach to estimate Shapley values efficiently and accurately. Our approach involves fine-tuning an existing ViT using a loss function designed specifically for Shapley values (Jethani et al., 2021b), and we prove that our loss bounds the estimation error without requiring ground truth values during train-

ing. Once trained, our explainer model provides a significant speedup over methods like KernelSHAP (Lundberg and Lee, 2017).

3. Our experiments compare the Shapley value-based approach to several groups of competing methods: attention-based, gradient-based and removal-based explanations. We find that our approach provides the best overall performance, correctly identifying influential and non-influential patches for both target and non-target classes. We verify this using three image datasets, and the results are consistent across multiple metrics.

Overall, our work shows that Shapley values can be made practical for vision transformers, and that they provide a compelling alternative to current attention- and gradient-based approaches.

7.3 Related Work

Understanding neural network predictions is a challenging problem that has been actively researched for the last decade (Simonyan et al., 2013; Zeiler and Fergus, 2014; Ribeiro et al., 2016). We focus on feature attribution, or identifying the specific input features that influence a prediction, but prior work has also considered other notions of interpretability (Olah et al., 2017; Kim et al., 2018). The various techniques that have been developed can be grouped into several categories, which we describe below.

Attention-based explanations Transformers use self-attention to associate weights with each pair of tokens (Vaswani et al., 2017), and a natural idea is to assess which tokens receive the most attention (Clark et al., 2019; Rogers et al., 2020; Vig et al., 2020). There are several versions of this approach, including *attention rollout* and *attention flow* (Abnar and Zuidema, 2020), which analyze attention across multiple layers. Attention is a popular interpretation tool, but it is only one component in a sequence of nonlinear operations that provides an incomplete picture of a model's dependencies (Serrano and Smith, 2019; Jain and Wallace,

2019; Wiegreffe and Pinter, 2019), and direct usage of attention weights has not been shown to perform well in vision tasks (Chefer et al., 2021).

Gradient-based methods For other deep learning models such as CNNs, gradient-based explanations are a popular family of approaches. There are many variations on the idea of calculating input gradients, including methods that modify the input (e.g., SmoothGrad, IntGrad) (Smilkov et al., 2017; Sundararajan et al., 2017; Xu et al., 2020), operate at intermediate network layers (GradCAM) (Selvaraju et al., 2017), or design modified backpropagation rules (e.g., LRP, DeepLift) (Bach et al., 2015; Shrikumar et al., 2017; Chefer et al., 2021). Although they are efficient to compute for arbitrary network architectures, gradient-based explanations achieve mixed results in quantitative benchmarks, including for object localization and the removal of influential features (Petsiuk et al., 2018; Hooker et al., 2019; Saporta et al., 2021; Jethani et al., 2021b), and they have been shown to be insensitive to the randomization of model parameters (Adebayo et al., 2018).

Removal-based explanations Finally, *removal-based explanations* are those that quantify feature importance by explicitly withholding inputs from the model (Covert et al., 2021). For models that require all features to make predictions, several options for removing features include setting them to default values (Zeiler and Fergus, 2014), sampling replacement values (Agarwal and Nguyen, 2020) and blurring images (Fong and Vedaldi, 2017). These methods work with any model type, but they tend to be slow because they require making many predictions. The simplest approach of removing individual features (known as *leave-one-out*, Ethayarajh and Jurafsky 2021) is relatively fast, but the computational cost increases as we examine more feature subsets.

Shapley values (Shapley, 1953) are an influential approach within the removal-based explanation framework. By examining all feature subsets, they provide a nuanced view of each feature's influence and satisfy many desirable properties (Lundberg and Lee, 2017). They are approximated in practice using methods like TreeSHAP and KernelSHAP (Lundberg et al.,

2020; Covert and Lee, 2021), but these approaches either are not applicable or do not scale to large ViTs. Recent work highlighted a connection between attention flow and Shapley values (Ethayarajh and Jurafsky, 2021), but this approach is fundamentally different from SHAP (Lundberg and Lee, 2017): attention flow treats each feature's influence on the model as strictly additive, which is computationally convenient but fails to represent feature interactions. Our work instead focuses on the original formulation (Lundberg and Lee, 2017) and aims to make Shapley values based on feature removal practical for ViTs.

7.4 Background

We focus here on vision transformers trained for classification tasks, where the input image is $\boldsymbol{x} \in \mathbb{R}^{224\times224\times3}$ and the class is $\boldsymbol{y} \in [K]$. ViTs consist of a sequence of self-attention layers, where the tokens are image patches $\boldsymbol{x}_i \in \mathbb{R}^{16\times16\times3}$ and we have $d = 196$ patches in total. Our goal is to calculate Shapley values efficiently for ViTs; see Chapter 2 for an introduction to the Shapley value and Chapter 5 for a description of its weighted least squares characterization, which we build upon here. Briefly, Shapley values assign credit to coalitional games of the form $v : \mathcal{P}(d) \mapsto \mathbb{R}$, and the Shapley value $\phi_i(v)$ for each player $i \in [d]$ is defined as

$$\phi_i(v) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \big(v(S \cup i) - v(S)\big).$$

There are two main challenges when using Shapley values to explain individual predictions (Chen et al., 2022). The first is properly withholding feature information, and we explore how to address this challenge in the ViT context (Section 7.5). The second is calculating Shapley values efficiently, because their computation scales exponentially with the number of inputs d. Traditionally, they are approximated using sampling-based estimators like KernelSHAP (Castro et al., 2009; Štrumbelj and Kononenko, 2010; Lundberg and Lee, 2017), but we build on a more efficient learning-based approach (FastSHAP) introduced by Jethani et al. (2021b) (Section 7.6).

7.5 Evaluating Vision Transformers with Partial Information

The basic idea behind Shapley values, as well as other removal-based explanations (Covert et al., 2021), is to evaluate the model with partial feature information and analyze how a prediction changes. Most models need values for all the features to make predictions, so in practice we require a mechanism to represent feature removal. For example, we can set held-out image regions to zero, or we can average the prediction across randomly sampled replacement values.

With vision transformers, the options for removing features are slightly different. Recent work has demonstrated the robustness of ViTs to randomly zeroed pixel values (Naseer et al., 2021), but the self-attention operation enables a more elegant approach: we can simply ignore tokens for image patches we wish to remove (Jain et al., 2021). We achieve this by masking attention values at each self-attention layer, or setting them to a large negative value before applying the softmax operation (see Appendix E.1). This resembles causal attention masking in transformer language models like GPT-3 (Brown et al., 2020), but we use masking for a different purpose. Alternatively, we could use a unique token value as in masked language models such as BERT (Devlin et al., 2018), which would involve simply setting held-out tokens to the mask value.

Using this attention masking approach, we can evaluate a ViT model $f(\boldsymbol{x}; \eta)$ given subsets of image patches, denoted by $\boldsymbol{x}_S$. However, because these partial inputs represent off-manifold examples, the predictions with partial information may not behave as desired. We have two options to correct this: (i) we can ensure that the model is trained with random masking, or (ii) we can fine-tune the model to encourage sensible behavior with missing patches. The first option is more direct, but it does not allow us to explain models trained without masking. For the latter option, we can create an updated model denoted by $g(\boldsymbol{x}_S; \beta)$ that we fine-tune using the following loss,

$$\min_{\beta} \quad \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[D_{\mathrm{KL}}\big(f(\boldsymbol{x}; \eta) \;||\; g(\boldsymbol{x}_{\boldsymbol{S}}; \beta) \big) \Big], \tag{7.1}$$

where $p(\boldsymbol{S})$ is a distribution over subsets. In practice, we sample the cardinality uniformly at random, and then sample the members at random. Intuitively, eq. 7.1 encourages $g(\boldsymbol{x}_S; \beta)$ to preserve the original model's predictions even with missing features. We use this loss because it satisfies the desirable property that the optimal model $g(\boldsymbol{x}_S; \beta^*)$ outputs the expected prediction given the available information (Covert et al., 2021), or

$$g(x_S; \beta^*) = \mathbb{E}[f(\boldsymbol{x}; \eta) \mid \boldsymbol{x}_S = x_S]. \tag{7.2}$$

Note that this represents a best-effort prediction, because if $f(\boldsymbol{x}; \eta) = p(\boldsymbol{y} \mid \boldsymbol{x})$ then we have $g(\boldsymbol{x}_S; \beta^*) = p(\boldsymbol{y} \mid \boldsymbol{x}_S)$. Similarly, in the case where $f(\boldsymbol{x}_S; \eta)$ is trained directly with random masking, the training process estimates $f(\boldsymbol{x}_S; \eta) \approx p(\boldsymbol{y} \mid \boldsymbol{x}_s)$ (see Appendix E.2). We refer to the fine-tuned model $g(\boldsymbol{x}_{\boldsymbol{S}}; \beta)$ as a *surrogate*, following the naming in prior work (Frye et al., 2021). Whether we use the original model or a version fine-tuned with random masking, our attention masking approach enables us to probe how individual predictions change as we remove groups of image patches.

7.6 Learning to Estimate Shapley Values

Given our approach for evaluating ViTs with partial information, we can use Shapley values to identify influential image patches for an input x and class y. This involves evaluating the model with many feature subsets x_S, so we define a coalitional game $v_{xy}(S) = g_y(x_S; \beta)$. Alternatively, if we use a model trained with masking, we can define the coalitional game as $v_{xy}(S) = f_y(x_S; \eta)$. Common Shapley value approximations are based on sampling feature permutations (Castro et al., 2009; Štrumbelj and Kononenko, 2010) or fitting a weighted least squares model (Lundberg and Lee, 2017; Covert and Lee, 2021), but these can require hundreds or thousands of model evaluations to explain a single prediction.[1] Instead, we develop a learning-based estimation approach for ViTs.

[1]The number of model evaluations depends on how fast the estimators converge, and we find that KernelSHAP requires >100,000 samples to converge for ViTs (Appendix E.8).

Our goal is to obtain an explainer model that estimates Shapley values directly. To do so, we train a new vision transformer $\phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta) \in \mathbb{R}^d$ that outputs approximate Shapley values for an input-output pair (x, y) in a single forward pass. Crucially, rather than training the model using a dataset of ground truth Shapley value explanations, we train it by minimizing the following objective,

$$\begin{aligned} \mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p_{\text{Sh}}(\boldsymbol{S})} \left[\left(v_{\boldsymbol{xy}}(\boldsymbol{S}) - v_{\boldsymbol{xy}}(\varnothing) - \boldsymbol{S}^\top \phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta) \right)^2 \right] \\ \text{s.t.} \quad \mathbf{1}^\top \phi_{\text{ViT}}(x, y; \theta) = v_{xy}([d]) - v_{xy}(\varnothing) \quad \forall\, (x, y), \end{aligned} \tag{7.3}$$

where $p_{\text{Sh}}(\boldsymbol{S})$ is a distribution defined as $p_{\text{Sh}}(S) \propto (|S| - 1)!(d - |S| - 1)!$ for $0 < |S| < d$ and $p_{\text{Sh}}([d]) = p_{\text{Sh}}(\varnothing) = 0$, and the inner product with $S \subseteq [d]$ represents the sum of the corresponding vector entries. Intuitively, eq. 7.3 encourages the explainer model to output feature scores that provide an additive approximation for the predictions with partial information, where the predictions are represented by $v_{\boldsymbol{xy}}(\boldsymbol{S})$ and the additive approximation by $v_{\boldsymbol{xy}}(\varnothing) + \boldsymbol{S}^\top \phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$.

The loss in eq. 7.3 was introduced by Jethani et al. (2021b) and is derived from an optimization-based characterization of the Shapley value (Charnes et al., 1988). To rigorously justify this training approach, we derive new results that show how this objective controls the Shapley value estimation error. Proofs are in Appendix E.4. First, we show that the explainer's loss for a single input is strongly convex in the prediction, a result that implies the existence of unique optimal predictions.

Lemma 7.1. *For a single input-output pair (x, y), the expected loss under eq. 7.3 for the prediction $\phi_{\text{ViT}}(x, y; \theta)$ is μ-strongly convex with $\mu = H_{d-1}^{-1}$, where H_{d-1} is the $(d-1)$th harmonic number.*

Next, we utilize the strong convexity property from Lemma 7.1 to prove our main result: that the explainer model's loss function upper bounds the distance between the exact and approximated Shapley values. This is notable because we do not utilize ground truth values

during training.

Theorem 7.1. *For a model $\phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ whose predictions satisfy the constraint in eq. 7.3, the objective value $\mathcal{L}(\theta)$ upper bounds the Shapley value estimation error as follows,*

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\big|\big|\phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta) - \phi(v_{\boldsymbol{xy}})\big|\big|_2\Big] \leq \sqrt{2H_{d-1}\Big(\mathcal{L}(\theta) - \mathcal{L}^*\Big)},$$

where $\mathcal{L}^*$ represents the loss achieved by the exact Shapley values.

This shows that our objective is a viable approach for training without exact Shapley values, because optimizing eq. 7.3 minimizes an upper bound on the estimation error. In other words, if we can iteratively optimize the explainer model so that its loss approaches the optimum obtained by the exact Shapley values ($\mathcal{L}(\theta) \to \mathcal{L}^*$), our estimation error will go to zero.

In practice, we train the explainer model $\phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ using stochastic gradient descent, and several other steps are important during training. First, we normalize the explainer's unconstrained predictions in order to satisfy the objective's constraint in eq. 7.3; this ensures that the Shapley value's *efficiency* property holds (Shapley, 1953). Next, rather than training the explainer from scratch, we fine-tune an existing model that can be either the original classifier or a ViT pre-trained on a different supervised or self-supervised learning task (Touvron et al., 2021; He et al., 2021); ViTs are more difficult to train than convolutional networks, and we find that fine-tuning is important to train the explainer effectively (Table E.1). Finally, we simplify the architecture by estimating Shapley values for all classes simultaneously. Our training approach is described in more detail in Appendix E.3.

By using a ViT to estimate Shapley values, we model the true explanation function and learn rich representations that capture not only which class is represented, but where key information is located. And by fine-tuning an existing model, we allow the explainer to re-use visual features that were informative for other challenging tasks. Ultimately, the explainer cannot guarantee exact Shapley values, but no approximation algorithm can; instead, it

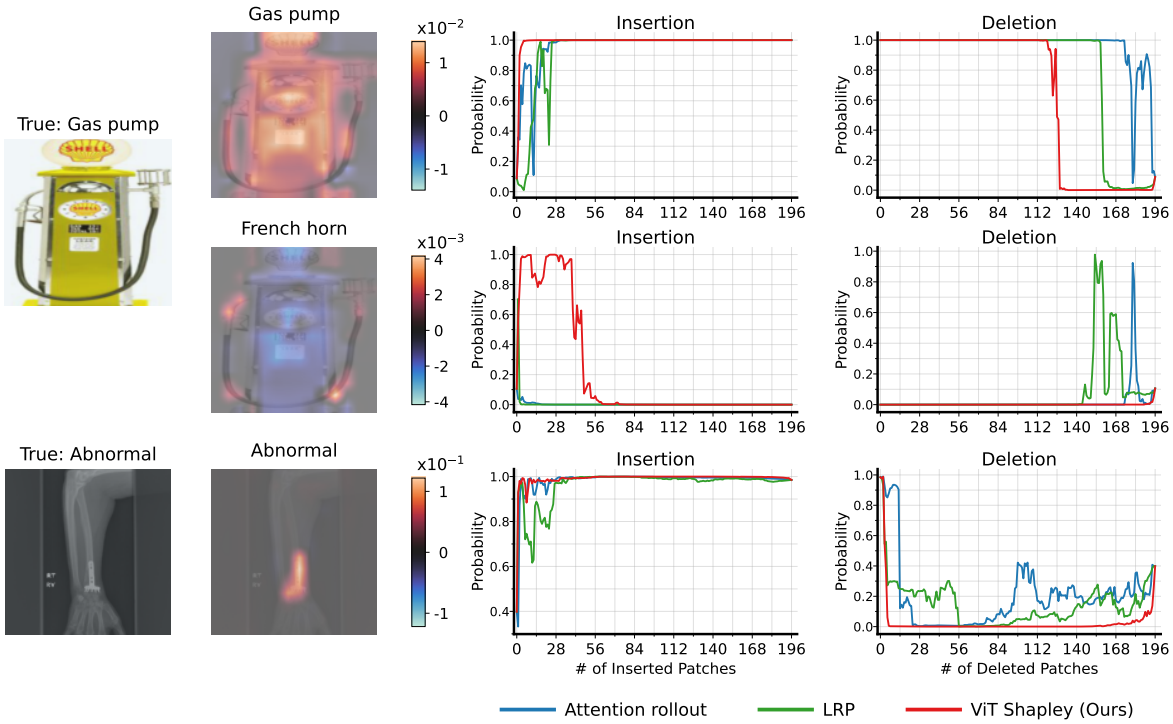


Figure 7.1: Explanations where our approach identifies relevant information for target and non-target classes. Left: original images from the ImageNette and MURA datasets. Middle left: explanations generated by ViT Shapley for specific classes. Right: probability of the class being explained after the insertion or deletion of important patches (higher is better for insertion, lower for deletion).

offers a favorable trade-off between accuracy and efficiency, and we find empirically that this approach offers a powerful alternative to the methods currently used for ViTs.

7.7 Experiments

We now demonstrate the effectiveness of our approach, termed *ViT Shapley*.[2] First, we evaluate attention masking for handling held-out patches in ViTs. Next, we compare explanations from ViT Shapley to several existing methods. Our baselines include attention-, gradient- and removal-based explanations, and we compare these methods via several metrics for ex-

[2] `https://github.com/suinleelab/vit-shapley`

planation quality, including insertion/deletion of important features (Petsiuk et al., 2018), sensitivity-n (Ancona et al., 2018), faithfulness (Bhatt et al., 2021) and ROAR (Hooker et al., 2019).

Our experiments are based on three image datasets: ImageNette, a natural image dataset consisting of ten ImageNet classes (Howard and Gugger, 2020; Deng et al., 2009), MURA, a medical image dataset of musculoskeletal radiographs classified as normal or abnormal (Rajpurkar et al., 2017), and the Oxford IIIT-Pets dataset, which has 37 classes (Parkhi et al., 2012). See Figure 7.1 for example images. We show results for ImageNette and MURA here, and the results for Pets are shown in Appendix E.8. We use ViT-Base models (Wightman, 2019) as classifiers for all datasets, unless otherwise specified.

7.7.1 Evaluating Image Patch Removal

Our initial experiments test whether attention masking is effective for handling held-out image patches. We fine-tuned the classifiers for each dataset following the procedure described in Section 7.5, and we also tested several approaches without performing any fine-tuning: attention masking, attention masking applied after the softmax operation (how dropout is often implemented for ViTs, Wightman 2019), setting input patches to zero (Naseer et al., 2021), setting token embeddings to zero, and replacing with random patches from the dataset. Finally, we performed identical fine-tuning while replacing input patches with zeros, which is equivalent to introducing a fixed mask token.

As a measure of how well missing patches are handled, we calculated the KL divergence relative to the full-image predictions as random patches are removed. This can be interpreted as a divergence measure between the masked predictions and the predictions with patches marginalized out (see Appendix E.2), or how close we are to correctly removing patch information. The metric is calculated with randomly generated patch subsets, and it represents whether the model makes reasonable predictions given partial inputs. Similar results for top-1 accuracy are in Appendix E.8.

Figure 7.2 shows the results. Most methods perform well with $<25\%$ of patches missing,

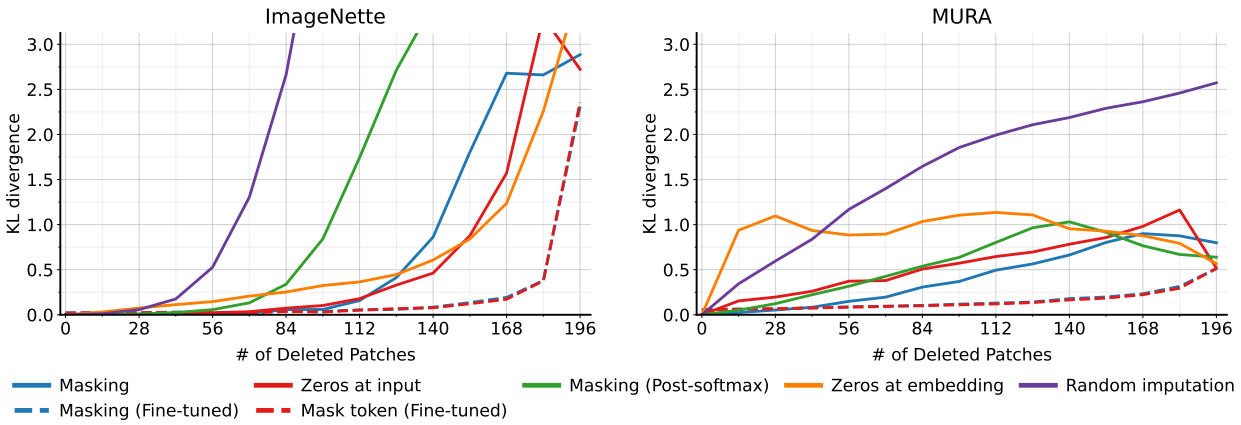


Figure 7.2: ViT predictions given partial information. We delete patches at random using several removal mechanisms, and then measure the quality of the resulting predictions via the KL divergence relative to the original, full-image predictions (lower is better).

leading to only small increases in KL divergence. This is especially true for ImageNette, where large objects make the model more robust to missing patches. However, the methods with no fine-tuning begin to diverge as larger numbers of patches are removed and the partial inputs become increasingly off-manifold. Thus, fine-tuning becomes necessary to properly account for partial inputs as more patches are removed.

For all datasets, we find that fine-tuning with either attention masking or input patches set to zero provide comparable performance, and that these perform best across all numbers of patches. This means that fine-tuning makes attention masking significantly more effective for marginalizing out missing patches, and these results suggest that training ViTs with held-out tokens may be necessary to enable robustness to partial information. As prior work suggests, properly handling held-out information is crucial for generating informative explanations (Frye et al., 2021; Covert et al., 2021), so the remainder of our experiments proceed with the fine-tuned attention masking approach.

7.7.2 Evaluating Explanation Accuracy

Next, we implemented ViT Shapley by training explainer models for both datasets. We used the fine-tuned classifiers from above to handle partial information, and we used the ViT-Base architecture with extra output layers to generate Shapley values for all patches. The explainer models were trained by optimizing eq. 7.3 using stochastic gradient descent (see details in Appendix E.3), and once trained, the explainer outputs approximate Shapley values in a single forward pass (Figure 7.1).

As comparisons for ViT Shapley, we considered a large number of baselines. For attention-based methods, we use attention rollout and the last layer's attention directed to the class token (Abnar and Zuidema, 2020). Similar to prior work (Chefer et al., 2021), we did not use attention flow due to the computational cost. Next, for gradient-based methods, we use Vanilla Gradients (Simonyan et al., 2013), IntGrad (Sundararajan et al., 2017), SmoothGrad (Smilkov et al., 2017), VarGrad (Hooker et al., 2019), LRP (Chefer et al., 2021) and GradCAM (Selvaraju et al., 2017). For removal-based methods, we use the leave-one-out approach (Zeiler and Fergus, 2014) and RISE (Petsiuk et al., 2018). Appendix E.6 describes the baselines in more detail, including how several were modified to provide patch-level results, and Appendix E.8 shows the running time for each method.

Given our set of baselines, we used several metrics to evaluate ViT Shapley. Evaluating explanation accuracy is difficult when the true importance is not known a priori, so we rely on metrics that test how removing (un)important features affects a model's predictions. Intuitively, removing influential features for a particular class should reduce the class probability, and removing non-influential features should not affect or even increase the class probability. Removal-based explanations are implicitly related to such metrics (Covert et al., 2021), but attention- and gradient-based methods may be hoped to provide strong performance with lower computational cost.

First, we implemented the widely used *insertion* and *deletion* metrics (Petsiuk et al., 2018). For these, we generate predictions while inserting/removing features in order of most

Table 7.1: Evaluating ViT Shapley using standard explanation metrics, with explanations calculated for the target class only. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	ImageNette			MURA		
	Ins. (↑)	Del. (↓)	Faith. (↑)	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.962 (0.004)	0.793 (0.013)	**0.694 (0.015)**	0.890 (0.010)	0.592 (0.013)	0.635 (0.016)
Attention rollout	0.938 (0.005)	0.880 (0.010)	**0.704 (0.015)**	0.845 (0.011)	0.692 (0.014)	0.618 (0.016)
GradCAM	0.914 (0.006)	0.937 (0.008)	0.680 (0.015)	0.899 (0.009)	0.681 (0.015)	0.631 (0.016)
IntGrad	0.967 (0.004)	0.930 (0.008)	0.403 (0.024)	0.897 (0.010)	0.796 (0.015)	0.201 (0.022)
Vanilla	0.950 (0.004)	0.808 (0.013)	**0.703 (0.015)**	0.890 (0.010)	0.537 (0.014)	0.629 (0.016)
SmoothGrad	0.947 (0.005)	0.942 (0.006)	**0.703 (0.015)**	0.870 (0.010)	0.813 (0.011)	0.617 (0.016)
VarGrad	0.949 (0.005)	0.946 (0.005)	**0.700 (0.015)**	0.857 (0.011)	0.823 (0.011)	0.615 (0.016)
LRP	0.967 (0.004)	0.779 (0.014)	**0.705 (0.015)**	0.900 (0.009)	0.551 (0.013)	0.646 (0.016)
Leave-one-out	0.969 (0.002)	0.917 (0.010)	0.140 (0.040)	0.926 (0.008)	0.694 (0.017)	0.308 (0.032)
RISE	0.977 (0.001)	0.860 (0.014)	**0.704 (0.015)**	0.957 (0.004)	0.573 (0.018)	0.618 (0.016)
ViT Shapley	**0.985 (0.002)**	**0.691 (0.014)**	**0.711 (0.015)**	**0.971 (0.002)**	**0.307 (0.013)**	**0.707 (0.013)**
Random	0.951 (0.005)	0.951 (0.005)	-	0.849 (0.010)	0.847 (0.010)	-

to least important, and we then evaluate the area under the curve of prediction probabilities (see Figure 7.1). Here, we average the results across 1,000 images for their true class. We use random test set images for ImageNette, and for MURA we use test examples that were classified as abnormal because these are more important in practice. When removing information, we use the fine-tuned classifier because this represents the closest approximation to properly removing information from the model (Section 7.5).

Table 7.1 displays the results, and we find that ViT Shapley offers the best performance on both datasets. RISE and LRP tend to be the most competitive baselines, and perhaps surprisingly, certain other methods fail to outperform a random baseline (GradCAM, SmoothGrad, VarGrad). The baselines are sometimes competitive with ViT Shapley on insertion, but the gap for the deletion metric is larger. Practically, this means that ViT Shapley identifies important features that quickly drive the prediction towards a given class, and that quickly reduce the prediction probability when deleted.

Next, we modified these metrics to address a common issue with model explanations:

Table 7.2: Evaluating ViT Shapley for explaining non-target classes. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	ImageNette		
	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	-	-	-
Attention rollout	-	-	-
GradCAM	0.021 (0.002)	0.005 (0.000)	-0.672 (0.015)
IntGrad	0.008 (0.001)	0.004 (0.000)	0.294 (0.022)
Vanilla	0.006 (0.001)	0.020 (0.001)	-0.682 (0.015)
SmoothGrad	0.006 (0.001)	0.006 (0.001)	-0.683 (0.015)
VarGrad	0.006 (0.001)	0.006 (0.001)	-0.680 (0.015)
LRP	0.004 (0.001)	0.022 (0.001)	-0.680 (0.015)
Leave-one-out	0.013 (0.002)	0.003 (0.000)	-0.017 (0.028)
RISE	0.023 (0.003)	0.002 (0.000)	-0.681 (0.015)
ViT Shapley	**0.093 (0.004)**	**0.001 (0.000)**	**0.672 (0.014)**
Random	0.005 (0.001)	0.005 (0.001)	-

that their results are not specific to each class (Rudin, 2019). ViT Shapley produces separate explanations for each class, so it can identify relevant patches even for non-target classes (see Figure 7.1). Table 7.2 shows insertion/deletion results averaged across all non-target classes for ImageNette. The attention-based methods do not produce class-specific explanations, and the remaining baselines generally provide poor results. Empirically, this is because the explanations are often similar across classes (see Appendix E.9). ViT Shapley performs best, particularly on insertion, and RISE is the best-performing baseline. Appendix E.8 shows results for MURA, as well as the curves used to calculate these results.

The insertion and deletion metrics only test importance rankings, so we require other metrics to test the specific attribution values. *Sensitivity-n* (Ancona et al., 2018) was proposed for this purpose, and it measures whether attributions correlate with the impact on a model's prediction when a feature is removed. The correlation is typically calculated across subsets of a fixed size, and then averaged across many predictions. *Faithfulness* (Bhatt et al., 2021) is a similar metric where the correlation is calculated across subsets of all sizes.

Table 7.1 and Table 7.2 show faithfulness results. Among the baselines, RISE and LRP remain most competitive, but ViT Shapley again performs best for both datasets. Figure 7.3

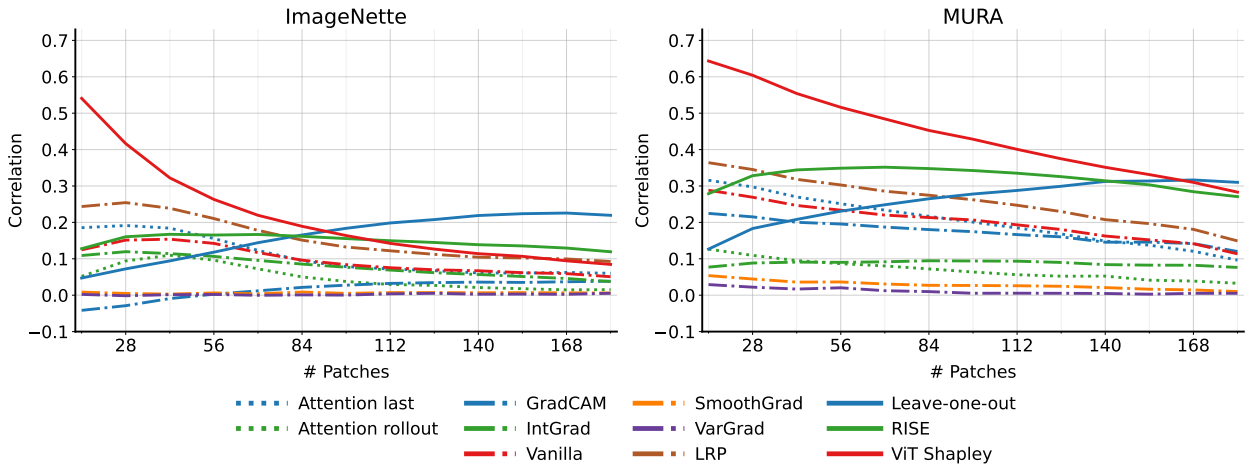


Figure 7.3: Sensitivity-n evaluation for different subset sizes. The metric is generated separately for a range of subset sizes, whereas faithfulness is calculated jointly over subsets of all sizes.

shows sensitivity-n results calculated across a range of subset sizes. Leave-one-out naturally performs best for large subset sizes, but ViT Shapley performs the best overall, particularly with smaller subsets. The sensitivity-n results focus on the target class, but Appendix E.8 shows results for non-target classes where ViT Shapley's advantage over many baselines (including LRP) is even larger.

Finally, we performed an evaluation inspired by ROAR (Hooker et al., 2019), which tests how a model's accuracy degrades as important features are removed. ROAR suggests retraining with masked inputs, but this is unnecessary here because the fine-tuned classifier is designed to handle held-out patches. We therefore generated multiple versions of the metric. First, we evaluated accuracy while using the fine-tuned classifier to handle masked patches. Second, we repeated the evaluation using a separate evaluator model trained directly with held-out patches, similar to EVAL-X (Jethani et al., 2021a). Third, we performed masked retraining as described by ROAR. The first version represents the original classifier's best-effort prediction, and the second is a best-effort prediction disconnected from the original model; masked retraining is similar, but the retrained model can exploit information communicated

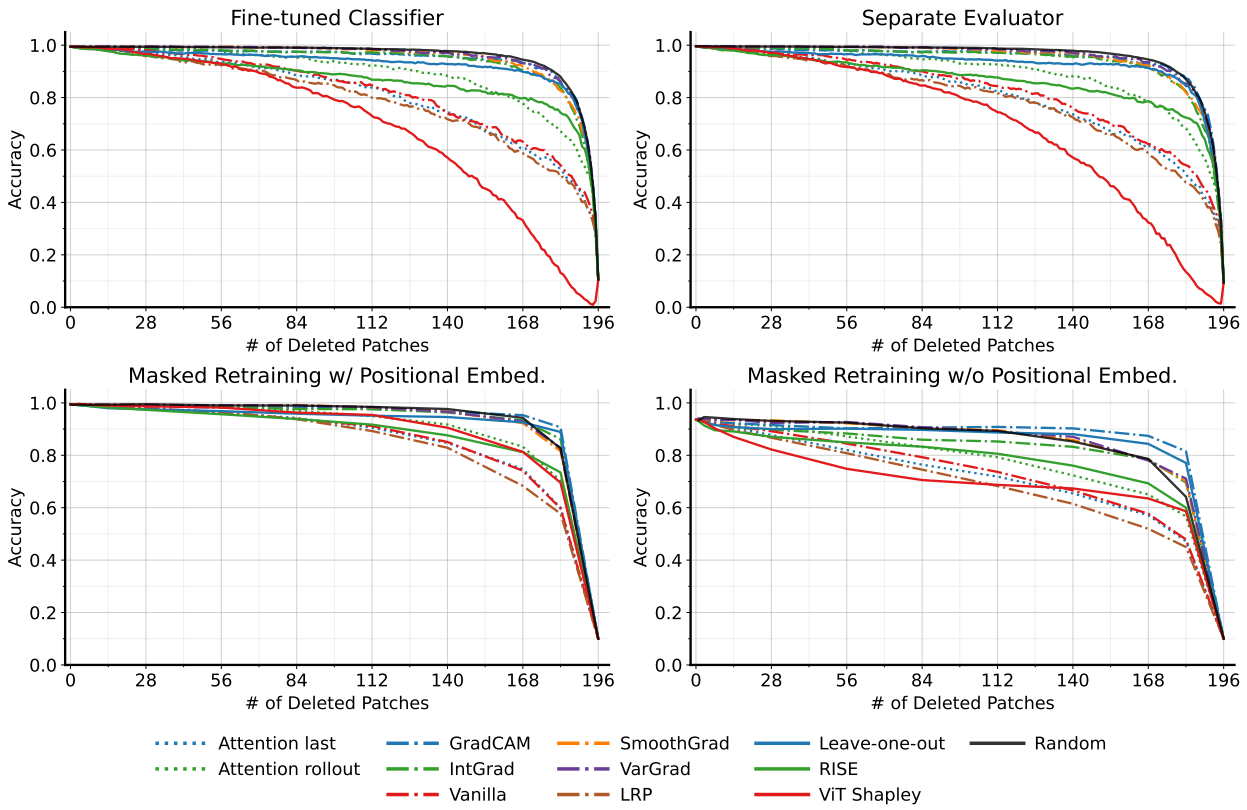


Figure 7.4: ImageNette accuracy when removing features in order of their importance, run with four evaluation strategies: fine-tuned classifier, separate evaluator, masked retraining, and masked retraining without positional embeddings.

by the masking, such as the shape and position of the removed object.

Figure 7.4 shows the results when removing important patches. ViT Shapley consistently outperforms the baselines across the first two versions of the metric, yielding faster degradation when important patches are removed. ViT Shapley also performs best when inserting important patches, yielding a faster increase in accuracy (Appendix E.8). It is outperformed by several baselines with masked retraining in the deletion direction (Figure 7.4 bottom left), but we find that this is likely due to spatial information leaked by ViT Shapley's deleted patches; indeed, when we retrained *without positional embeddings*, we found that ViT Shapley achieved the fastest degradation with a small number of deleted patches (Figure 7.4 bottom

right). Interestingly, positional embeddings in ViTs offer a unique approach to alleviate ROAR's known information leakage issue (Jethani et al., 2021a).

In addition to these experiments, we include many further results in the supplement (Appendix E.8). First, we observe similar benefits for ViT Shapley when using the Oxford-IIIT Pets dataset. Next, regarding the choice of architecture, we observe consistent results when replacing ViT-Base with ViT-Tiny, ViT-Small or ViT-Large. We also replicate our results when using a classifier trained directly with random masking, an approach discussed in prior work to accommodate partial input information (Covert et al., 2021). We then generated metrics comparing ViT Shapley's approximation quality with KernelSHAP (Lundberg and Lee, 2017), and we found that ViT Shapley's accuracy is equivalent to running KernelSHAP for roughly 120,000 model evaluations (Appendix E.8). Lastly, we provide qualitative examples in Appendix E.9, including comparisons with the baselines. Overall, these results show that ViT Shapley is a practical and effective approach for explaining ViT predictions.

7.8 Conclusion

In this work, we developed a learning-based approach to generate Shapley values for ViTs. Our approach involves training a separate explainer model using an objective that does not require ground truth supervision, and ViT Shapley outperforms a variety of attention- and gradient-based methods across a range of accuracy metrics. Our experiments provide empirical support for Shapley values as an alternative to attention-based approaches, which remain popular for transformer models. Future directions involve extending ViT Shapley to NLP models, operating with arbitrary token groups or superpixels, and accelerating or otherwise improving the explainer model's training.

Part III

FEATURE SELECTION WITH DEEP LEARNING

Chapter 8

GENE SELECTION FOR SPATIAL TRANSCRIPTOMICS

8.1 Chapter Notes

This chapter presents joint work with Rohan Gala, Tim Wang, Karel Svoboda, Uygar Sümbül and Su-In Lee that was published in Nature Communications in 2023. The content of this chapter is a minor variant of the published version, which was written in the style of a biology journal paper. The main contribution of this work is to develop a differentiable gene selection method for the context of spatial transcriptomics, which often relies on technologies that detect expression levels for only a small number of genes.

8.2 Introduction

Cell type classification has been revolutionized by recent advances in single-cell genomics. It is now possible to profile the entire mRNA repertoire (i.e., the transcriptome) of tens, or even hundreds, of thousands of individual cells (scRNA-seq) in a single experiment. Large-scale scRNA-seq studies have provided high-resolution taxonomies of the transcriptomic cell types in many tissues across several species, and leveraging data from these scRNA-seq studies, spatial transcriptomic methods can examine molecularly defined cells in their native context (Femino et al., 1998; Wang et al., 2012; Codeluppi et al., 2018; Chen et al., 2015a; Wang et al., 2018; Shah et al., 2016; Eng et al., 2019; Qian et al., 2020; Zhang et al., 2021; Sun et al., 2021). This has led to their use in large consortia such as the Human Cell Atlas and Brain Initiative Cell Census Network, and widespread recognition of their promise (Marx, 2021).

In particular, fluorescence *in situ* hybridization (FISH) is a prominent spatial transcriptomics approach, and it is the basis of many recently developed technologies (Codeluppi

et al., 2018; Chen et al., 2015a; Shah et al., 2016; Eng et al., 2019; Zhang et al., 2021; Close et al., 2021). By revealing the spatial organization of cells within tissues, FISH is playing a central role in uncovering the fundamental principles of brain organization (Close et al., 2021). In conjunction with other experimental modalities (e.g., morphological, connectivity, or electrophysiological studies), FISH can also link transcriptomic identity with other data to provide a better understanding of the functional role of individual cell types (Liu et al., 2021a; Gouwens et al., 2020; Scala et al., 2021; Zhang et al., 2021; Yao et al., 2021).

Whereas scRNA-seq detects genes in a largely unbiased manner, FISH-based technologies assay a pre-defined list of genes. In routine FISH experiments, only a small fraction of the transcriptome is targeted (Shah et al., 2016; Codeluppi et al., 2018; Close et al., 2021; Zhang et al., 2021); this is in part because the complexity and duration of FISH experiments increases sharply with the number of target genes, and also because highly specialized methods capable of probing thousands of genes are applicable only to thin tissue sections and cultured cells (Eng et al., 2019; Fang et al., 2022). Thus, judicious selection of a small number of highly informative target genes (i.e., a gene panel) is key for most FISH experiments. Experimentalists often rely on *ad hoc* approaches to gene selection, most commonly choosing markers based on prior knowledge or very high expression in a limited subset of cells (Zhang et al., 2021; Condylis et al., 2022). Such methods are suboptimal as they tend to overlook genes with more complex expression patterns and rarely account for correlated expression between selected genes, which yields redundant information. Here, we frame the identification of markers as a feature selection problem and seek to address it in a principled manner using tools from machine learning.

Feature selection problems arise in many domains (Subramanian et al., 2017; Abid et al., 2019; Aevermann et al., 2020; Song et al., 2021; Heydari et al., 2022), but spatial transcriptomics studies present unique challenges and thus demand a specialized solution. Importantly, because reference datasets consisting of spatially resolved mRNA detection measurements for thousands of genes are unavailable, scRNA-seq datasets are used instead to guide the selection process. The use of a surrogate dataset presents new obstacles: the expression

counts observed by scRNA-seq and spatial transcriptomics technologies may differ significantly, with a relationship that is nonlinear and noisy (Shah et al., 2018; Lopez et al., 2019; Liu et al., 2022). Hence, a gene panel selected without considering the difference between the datasets is unlikely to perform as well as intended, as we demonstrate with several existing methods.

Moreover, the optimal gene panel should account for specifics of the target experiment, which may demand tuning of the selection criterion. For instance, linking of spatial characterization of gene expression through spatial transcriptomics and the electrical properties of neurons (Gouwens et al., 2020; Scala et al., 2021) may require a gene panel that relates to membrane excitability. Another scenario could involve investigating a specific subclass of cells; for example, exploring neurons expressing a specific marker gene in a particular brain region demands a gene panel based on reference data from this molecularly and spatially constrained population. Furthermore, certain spatial transcriptomics methods might require the target genes to have either relatively high or low expression levels; for example, when using low-resolution detection methods, it may be preferable to prioritize highly expressed genes.

We address these challenges by introducing ***p****redictiv****e*** *and* ***r****obust gene* ***s****electi****o****n for* ***s****patial* ***t****ranscriptomics* (PERSIST), an algorithm to select genes that can serve as valuable targets in spatial transcriptomics studies. PERSIST uses scRNA-seq data and deep learning to find a small number of highly informative genes whose expression can predict the genome-wide expression profile. In doing so, PERSIST trains a reconstruction model with a loss function that accounts for noisy gene dropouts in scRNA-seq; incorporates expert knowledge by pre-selecting or pre-filtering genes; scales to very large datasets using minibatched training; and quantizes gene expression levels to account for the domain shift between scRNA-seq and spatial transcriptomics. Furthermore, our deep learning-based selection mechanism is flexible: by changing the prediction target, PERSIST can also operate in a supervised rather than unsupervised fashion to address specific experimental aims, such as cell type classification or electrophysiological characterization. Our work focuses primarily

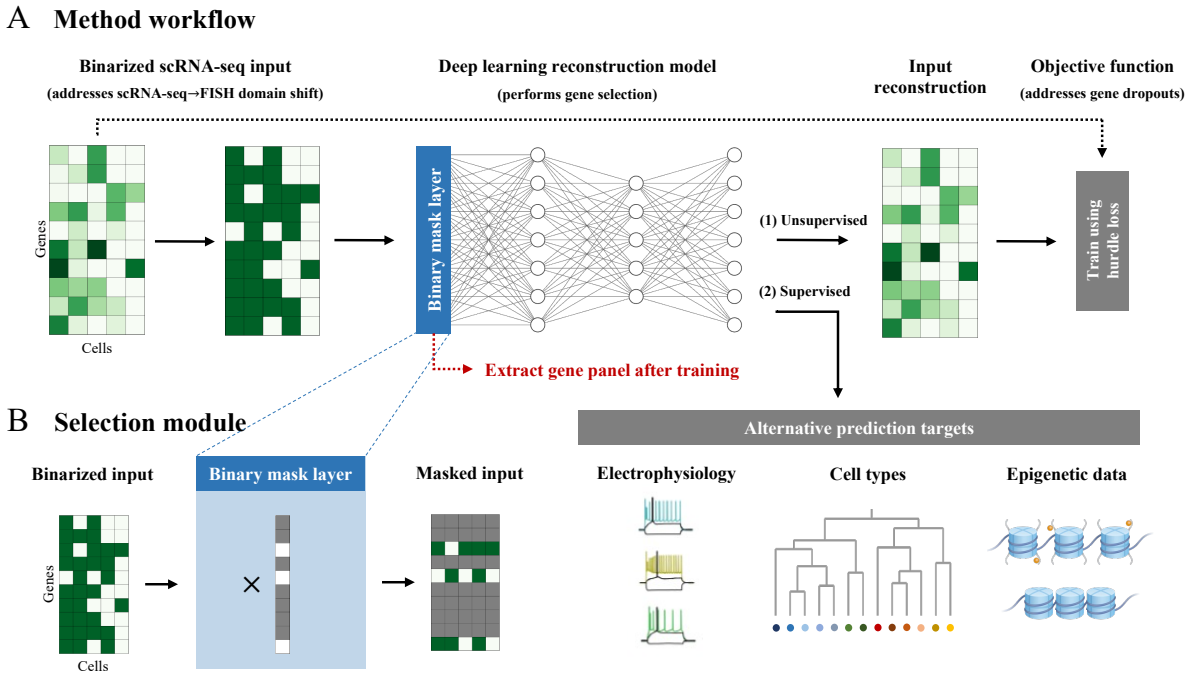


Figure 8.1: Overview of *predictive and robust gene selection for spatial transcriptomics* (PERSIST). PERSIST selects genes using a deep learning model trained to reconstruct the genome-wide expression profile. **A**, The model is trained using scRNA-seq data, which is binarized to address the domain shift relative to FISH measurements. By default, the model aims to reconstruct the original scRNA-seq gene expression levels, and the objective function is a hurdle loss designed to account for noisy gene dropouts. Alternatively, one can use a supervised prediction target to address specific experimental aims, such as cell type classification. After training, the selected gene panel is extracted from the model's binary mask layer. **B**, The binary mask layer selects genes by multiplying the input with a learned binary mask, which controls the specific input features that are used to make predictions.

on FISH-based studies, but many of the challenges identified above are common to a larger class of spatial transcriptomic methods (Qian et al., 2020; Sun et al., 2021), thus suggesting broader applicability of our method.

We validate our approach using reference datasets from different technologies (plate-based SmartSeq and droplet-based 10X), multiple brain regions (V1, ALM, MOp) and different species (mouse, human) on classification and reconstruction tasks. We then highlight PERSIST's flexibility and show how to incorporate a different data modality (electrophysiology)

with FISH experiments using a large Patch-seq dataset (Gouwens et al., 2020). Finally, we devise an evaluation procedure to showcase the effectiveness of our robust inference approach based on gene quantization using a recent MERFISH dataset (Zhang et al., 2021), which we show allows predictive models to transfer across technologies despite the measurement differences. Through our comprehensive set of experiments and comparisons with other methods, we provide strong evidence that PERSIST can identify valuable gene targets for spatial transcriptomics studies.

8.3 Results

8.3.1 Selecting genes using deep learning

Given scRNA-seq data from a cell population to be profiled using spatial transcriptomics, PERSIST selects a small panel of genes that can optimally reconstruct the entire scRNA-seq expression profile. Intuitively, such gene panels are useful for a variety of downstream tasks because they sacrifice minimal information. Our approach is inspired by classical dimension-reduction techniques like principal components analysis (PCA) (Jolliffe, 1986), but PERSIST selects a discrete set of genes rather than finding linear combinations. Additionally, it reconstructs the original data using a non-linear model and with a quality measure more appropriate for scRNA-seq data. PCA measures reconstruction quality using a mean squared error (MSE) loss, which recent work has found to be ill-suited for scRNA-seq (Lopez et al., 2018; Eraslan et al., 2019), so PERSIST instead uses a *hurdle loss function* to account for noisy gene dropouts (McDavid, 2016). 'Dropouts' refer to the failure to detect mRNA transcripts due to inefficiencies in cDNA library preparation, which is prevalent when using lower resolution (and typically higher throughput) platforms such as 10X (Clivio et al., 2019; Qiu, 2020; Sarkar and Stephens, 2021). The hurdle loss used by PERSIST therefore involves separately predicting each gene's expression level and whether it is actually expressed, which lets the model explicitly represent dropout noise in its predictions (see Methods).

PERSIST uses a deep learning model with a custom layer designed to pinpoint a small

number of useful input features (Figure 8.1). This approach is inspired by recent work on *differentiable feature selection*, which enables neural networks to select features using gradient-based optimization (Maddison et al., 2016; Jang et al., 2016; Chang et al., 2017; Abid et al., 2019). The selection layer applies a learned binary mask that sparsifies over the course of the optimization process; information initially flows through the model from all genes, but the relevant inputs are gradually reduced down to a user-specified number. The model's memory usage can be managed via the minibatch size used for training, and when necessary, the computational cost can be further reduced by performing a preliminary filtering step (see Methods).

By default, PERSIST operates in an *unsupervised* manner by reconstructing the full scRNA-seq expression profile, which removes the need for any labels or manual annotation. However, PERSIST can also operate in a *supervised* manner by incorporating cell-level annotations as the model's prediction target, such as cell type labels or complementary epigenetic data like chromatin accessibility and methylation (Figure 8.1A). As we show in our experiments, this gives PERSIST a versatility that is not shared by other methods, and which lets practitioners select genes that are tailored to meet specific biological questions and objectives. While spatial transcriptomics studies often have specific goals like classifying cell types (Codeluppi et al., 2018; Eng et al., 2019; Wang et al., 2018; Zhang et al., 2021), enabling PERSIST to operate in an unsupervised manner is important because reference cell type clusterings are not always available, consensus definitions of cell types are still evolving (Zeng, 2022), and focusing on gene expression enables unbiased characterization of complex tissues and specific brain regions.

The goal of our evaluation is to demonstrate that genes identified by PERSIST can serve as valuable targets in spatial transcriptomics studies. Showing this is not straightforward, both due to the cost of running multiple FISH studies with panels selected by different criteria, as well as the difficulty of providing an unbiased comparison through studies conducted on different tissues. We therefore evaluate the PERSIST gene panels by simulating their use in FISH studies, and in particular via prediction tasks that would be of interest

to practitioners in such studies. scRNA-seq and FISH have very different noise sources and detection issues, and the number of transcript counts observed by each technology can differ substantially for the same cell, so our simulated prediction tasks binarize gene expression levels. This pre-processing step allows models to transfer across technologies despite the domain shift, as we demonstrate in an experiment with MERFISH data.

Our experiments compare PERSIST to several widely used and state-of-the-art gene selection methods. We tested the Seurat (Stuart et al., 2019) and Cell Ranger (Zheng et al., 2017) gene selection procedures, which are based on per-gene variance and dispersion levels and are implemented in the popular ScanPy package (Wolf et al., 2018). These methods are designed primarily to reduce computation and inform clustering studies that help determine marker genes, but they are not intended to directly select gene panels; therefore, their performance is not expected to be competitive with methods designed explicitly for selecting small gene sets. Next, we tested the recently proposed GeneBasis method (Missarova et al., 2021) that selects genes using a greedy algorithm to preserve the data manifold. Finally, we considered three methods that aim to differentiate cell types: a method that maximizes the information about cell type labels (MutInfo) (Zhang et al., 2021; Li et al., 2017b), a method that identifies key gene predictors using feature importance scores (SMaSH) (Nelson et al., 2022), and the scGeneFit method (Dumitrascu et al., 2021) that uses linear programming. These methods span a range of selection criteria, but PERSIST is a flexible method that can be adapted to multiple experimental objectives relevant to practitioners, and that was designed specifically for transferability to spatial transcriptomics studies.

8.3.2 PERSIST enables more accurate scRNA-seq expression profile reconstruction

We first tested PERSIST on two scRNA-seq datasets: a SmartSeq v4 (Ramsköld et al., 2012) dataset consisting of 22,160 neurons from the mouse primary visual (V1) and anterior lateral motor (ALM) cortices (Tasic et al., 2018) (hereafter referred to as SSv4), and a 10X (Zheng et al., 2017) dataset consisting of 72,629 neurons from the human motor (M1) cortex (Bakken et al., 2021) (hereafter referred to as 10X). These datasets profile different brain

regions and species and were collected using two library preparation platforms that yield different levels of sparsity. Working with an initial set of 10,000 high-variance genes, we used PERSIST and the other gene selection methods to identify panels of $8-256$ marker genes, a range that spans the vast majority of FISH studies.

As a benchmark metric for our comparisons, we calculated the portion of variance explained in the genome-wide scRNA-seq expression profile by each selected gene panel. For gene panels of all sizes on both datasets, PERSIST explained more variance and outperformed the unsupervised methods Seurat, Cell Ranger and GeneBasis (Figure 8.2A, C). The GeneBasis approach is most competitive with PERSIST, but it explained considerably less variance for smaller gene panels, which are most commonly used in experiments. There is diminishing improvement as the panel size increases, and even large panels, such as those with 256 genes, fail to explain all the variance. This is due not only to the many factors of variation in the full expression profiles, but to high noise levels in the data. To verify this, we calculated the amount of variance explained by the cell types in each dataset. We found that cell type labels explained just 19% of the variance in the SSv4 data and 11% in the 10X data, suggesting high intra-type variability due to stochasticity in gene expression and detection. Perhaps surprisingly, the PERSIST panels can explain more variance than the cell type identities given enough genes.

Importantly, PERSIST binarizes gene expression levels during training whereas Seurat, Cell Ranger and GeneBasis all use either raw or logarithmized expression counts. This creates a degree of inconsistency among methods, so we asked whether this pre-processing step could account for differences in performance. A modified version of GeneBasis becomes more competitive with PERSIST, whereas Seurat and Cell Ranger perform worse with binarization (Figure F.2). For MutInfo, SMaSH and scGeneFit, which leverage cell type labels to select genes, we find that PERSIST outperforms these methods independent of binarization (Figure F.2). Overall, the results show that PERSIST's binarization step can be incorporated into several of the baselines to enable better transferability to FISH studies.

Recent work has demonstrated that the "dropout pattern" of a cell (i.e., the set of genes

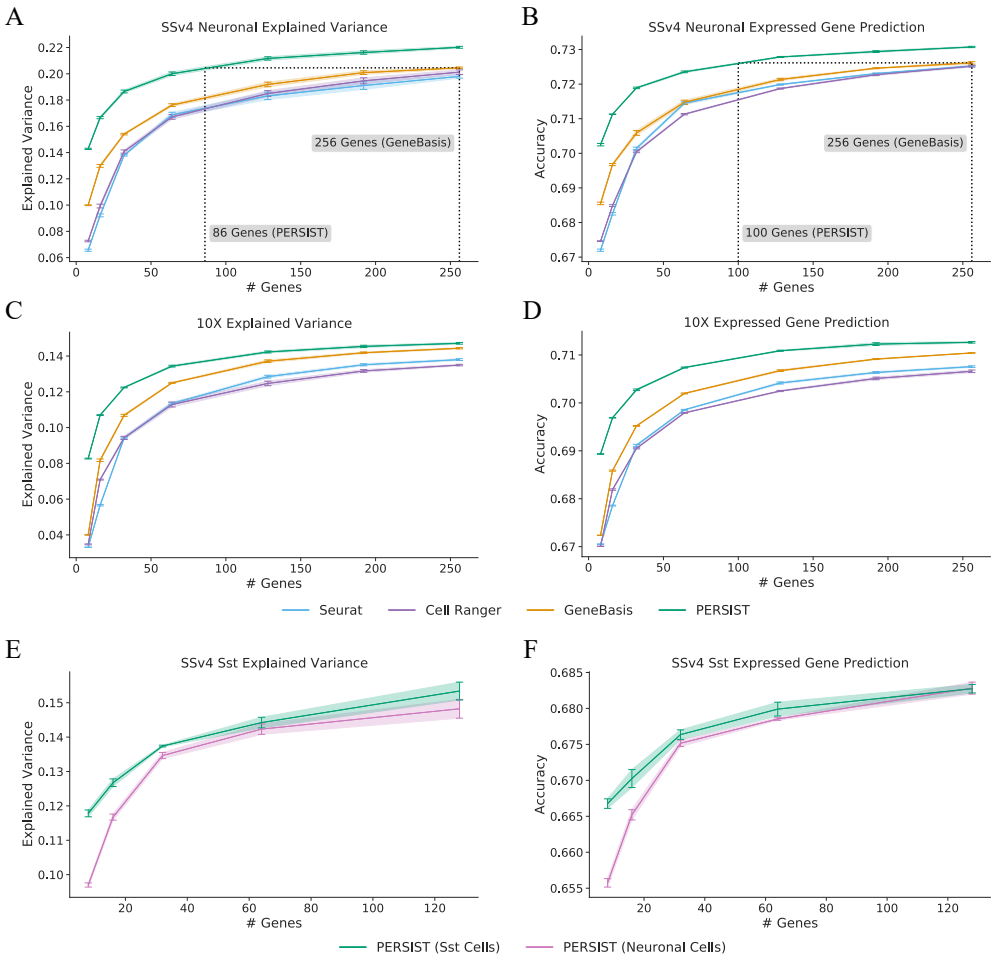


Figure 8.2: Evaluating gene panels based on their ability to reconstruct scRNA-seq expression profiles. The selected genes are used to predict either log-normalized expression counts for the remaining genes (**A**, **C**, **E**) or whether each gene is expressed (**B**, **D**, **F**). The panels selected by PERSIST perform better than those selected by previous methods. **A**, Explained variance for gene panels identified by PERSIST and other methods with the SSv4 V1/ALM neuronal cells (higher is better). **B**, Expressed gene prediction accuracy with the SSv4 V1/ALM neuronal cells (higher is better). **C**, Explained variance for gene panels with the 10X M1 cells. **D**, Expressed gene prediction accuracy for panels with the 10X M1 cells. **E**, Explained variance for gene panels with the SSv4 V1/ALM Sst cells. **F**, Expressed gene prediction accuracy for panels with the SSv4 V1/ALM Sst cells. All error bars represent 95% confidence intervals determined by training with five bootstrapped datasets.

not detected by scRNA-seq) is nearly as informative as quantitative expression levels for identifying cell types (Li and Quon, 2019; Qiu, 2020). We thus wondered whether the selected gene panels could predict the set of genes with non-zero transcript counts in the rest of the scRNA-seq dataset (Figure 8.2B, D). In this task, PERSIST again performs more accurately than other methods on both datasets for all panel sizes; this is due in part to our hurdle loss function, which involves predicting whether each gene is detected. For this analysis, we excluded genes that were rarely expressed and housekeeping genes that are ubiquitously expressed, focusing on those expressed in 20−80% of cells (n=4,972 genes), but similar results were found for varying cutoffs (Figure F.4). Figure F.3 shows the prediction accuracy for each gene, revealing that those with moderate mean expression are more difficult to predict than those with predominantly high or low expression.

Finally, from an experimental standpoint, it would be most practical to select a single general-purpose gene panel using the entire dataset rather than generating a gene panel for each particular subtype of interest. To assess the feasibility of this strategy, we focused on the class of somatostatin-expressing (Sst) interneurons (2,701 cells) in the SSv4 dataset and compared gene sets selected by PERSIST when trained on the entire scRNA-seq dataset versus just the Sst subpopulation. Performance improved when we only used data from the specific cell type of interest during gene selection (Figure 8.2E-F). However, the improvement diminished when 32 or more genes were selected, which suggests that general purpose gene panels may be appropriate for technologies that assay large numbers of genes (e.g., multiplexed methods like MERFISH Chen et al. 2015a).

8.3.3 PERSIST enables accurate cell type classification

As another evaluation metric, we tested how accurately the gene panels selected by each method can classify cell types, which is a common goal of spatial transcriptomics studies (Codeluppi et al., 2018; Eng et al., 2019; Wang et al., 2018; Zhang et al., 2021). We utilized transcriptomic cell types defined via the original SSv4 and 10X datasets for our evaluation, and the classification accuracy with binarized input data simulates the accuracy in a subse-

quent FISH experiment. In addition to the various gene selection methods, we also consider a panel of marker genes identified by Tasic et al. (2018) for cell types in the SSv4 dataset. For all gene selection approaches, larger panels enabled increasingly accurate cell type classification (Figure 8.3). As expected, supervised methods that use cell type annotations during their selection procedure (e.g., MutInfo) perform better than unsupervised methods that use only unlabeled scRNA-seq data (e.g., Seurat). To emphasize this distinction between methods, we present results separately for unsupervised (Figure 8.3A, C) and supervised methods (Figure 8.3B, D).

Among the unsupervised approaches, PERSIST outperforms Seurat, Cell Ranger and GeneBasis for panels of all sizes. For example, when using 64 genes, PERSIST reaches 74% accuracy with the SSv4 data and 78% with the 10X data considering a total of 113 and 117 cell types, respectively. GeneBasis is the most competitive unsupervised baseline with the 10X data, and either GeneBasis or Seurat are most competitive with the SSv4 data. The gap in performance is largest for small panels, and the various methods roughly converge in accuracy for panels of 128 genes. As additional results, Figures F.5 and F.6 show confusion matrices that represent how cells of each type are most often classified, and Figure F.7 shows that distinct expression patterns for the selected genes are visible within each cell type.

PERSIST is an unsupervised selection algorithm by default, but we can also adapt it to cell type classification by using cell type labels as the prediction target during training (Figure 8.1). This supervised version of our approach, termed PERSIST-Classification, matches or exceeds the performance of the other supervised approaches. For example, it reaches 81% accuracy in the SSv4 dataset and 82% accuracy in the 10X dataset using panels with 64 genes—a significant improvement over the unsupervised version. This illustrates the flexibility of our deep learning-based selection approach, and that PERSIST can be adapted to specific experimental objectives by simply adjusting its prediction target. The peak accuracy we observe with 128 genes is 85%, so our results also suggest that panels of these small sizes may be incapable of perfectly distinguishing fine-grained cell types, and therefore that FISH studies may benefit from analyzing more coarse clusterings.

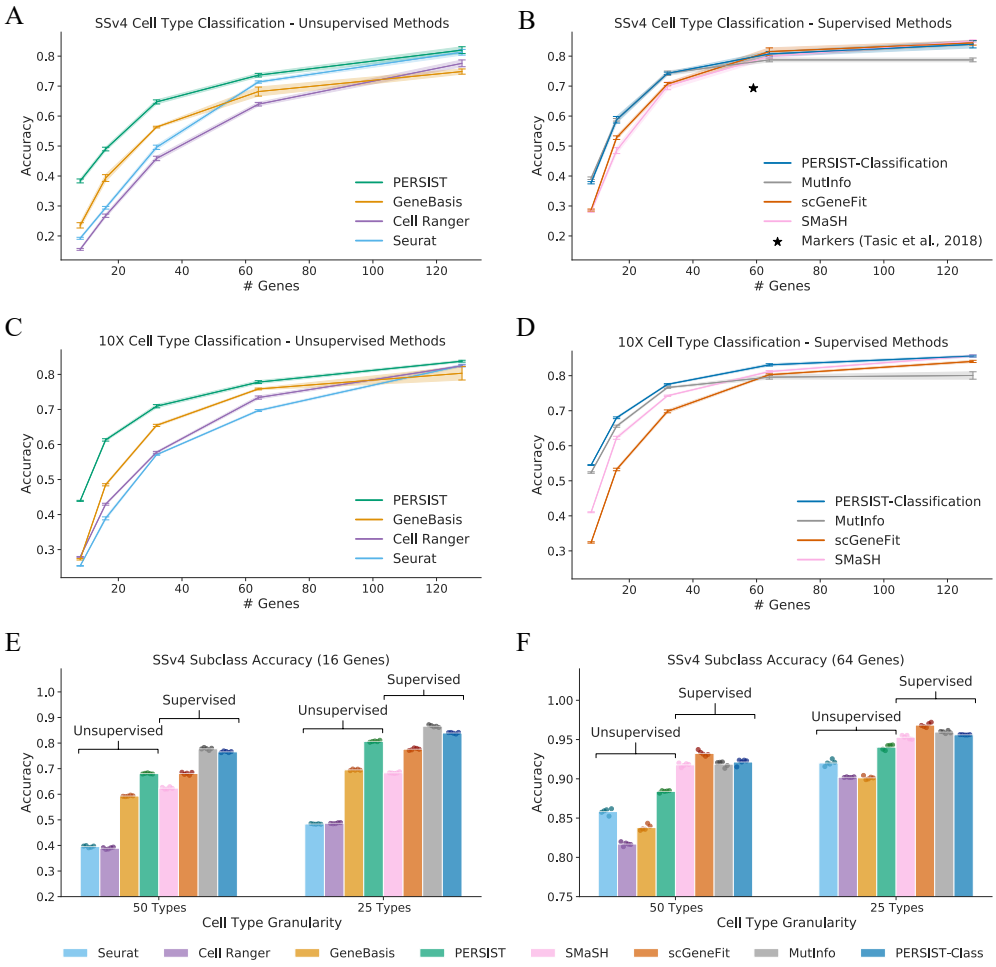


Figure 8.3: Evaluating gene panels based on their ability to classify cell types. The selected genes are used to classify scRNA-seq cell types, and PERSIST achieves strong performance despite not using cell type labels during the selection process. PERSIST-Classification, a version modified to distinguish cell types, matches the best existing supervised approaches. **A**, Cell type classification accuracy for gene panels selected by unsupervised methods with the SSv4 dataset (higher is better). **B**, Accuracy for gene panels selected by supervised methods with the SSv4 dataset. **C**, Accuracy for unsupervised methods with the 10X dataset. **D**, Accuracy for supervised methods with the 10X dataset. **E**, Accuracy for cell subtypes obtained by merging the SSv4 dataset's transcriptomic hierarchy, for panels of 16 genes. **F**, Accuracy with merged SSv4 cell subtypes for panels of 64 genes. All error bars represent 95% confidence intervals from training with five bootstrapped datasets, and **E**-**F** show results from the five runs.

Prior work suggests that the highly similar terminal nodes of the classification hierarchy may not all correspond to distinct cell types, but may instead reflect cell states or spatial gradients in gene expression (Stanley et al., 2020). We therefore divided cells into broader subclasses, which leads to greater classification accuracy because the groupings are more distinct and, trivially, there are fewer of them. For the SSv4 dataset, if we classify cells into 25 subclasses rather than the full 113 cell types, PERSIST-Classification reaches 84% accuracy using 16 genes (vs. 59% for 113 types) and 96% accuracy using 64 genes (vs. 81% for 113 types). In comparison, the unsupervised version of PERSIST provides accuracy just 3% and 1% worse, respectively (Figure 8.3E-F). The results are similar but slightly less accurate when we classify into 50 subclasses. With a reduced number of cell type subclasses, Seurat, Cell Ranger and GeneBasis are still not competitive with PERSIST, and PERSIST-Classification remains on par with the supervised procedures.

Although PERSIST does not match PERSIST-Classification in terms of classification accuracy, it is notable that PERSIST remains competitive despite not having access to cell type labels. This indicates that PERSIST successfully captures cell type information in an unsupervised manner. We attribute the strong cell type classification performance to our deep learning-based selection mechanism, which identifies non-redundant genes that help reconstruct the full expression profile, and to our use of gene expression binarization. In an ablation study, we also find that PERSIST's hurdle loss function is an important design choice, because it leads to better cell type classification accuracy than training with mean squared error loss (Figure 8.8). These results are promising because a consensus definition of cell types, and their continuous versus discrete nature, are far from settled (Trapnell, 2015; Tasic et al., 2018; Scala et al., 2021). Moreover, reference label information is currently available for only a handful of mouse and human tissues, and PERSIST can be used in an unsupervised manner in settings that lack established cell type hierarchies.

8.3.4 PERSIST can be adapted to predict electrophysiological properties

As a further demonstration of our method's flexibility, we developed a specialized variant of PERSIST to identify marker genes that predict each cell's electrophysiological properties. For this purpose, we used a multi-modal Patch-seq dataset (Gouwens et al., 2020; Gala et al., 2021) containing transcriptomic and electrophysiological information from 3,411 GABAergic neurons across 53 cell types in the mouse visual cortex. Specifically, the transcriptomic profile consists of the scRNA-seq counts for 1,252 curated genes, and the electrophysiological profile consists of a set of 44 sparse principal components (sPCs) summarizing different portions of the measurement protocol, as well as 24 biologically relevant features (Gala et al., 2021).

To select gene panels using the Patch-seq dataset, we used baseline methods that require only unlabeled expression data (Seurat, Cell Ranger and GeneBasis) because the dataset lacks cell type annotations. For PERSIST, we first ran it in an unsupervised manner by selecting genes that can optimally reconstruct the full expression profile. Next, we also ran PERSIST in a supervised manner by using the vector of electrophysiological features as the prediction target, yielding a variant of our approach that we refer to as PERSIST-Ephys. We then investigated how well each gene panel represents a cell's electrophysiological profile.

As an evaluation metric, we attempted to predict the electrophysiological features of each neuron using the expression levels of the genes in each panel. Similar to previous experiments, we binarized gene expression levels to simulate applicability in a subsequent FISH study. We find that PERSIST-Ephys achieves the highest predictive accuracy, reaching a higher portion of explained variance with panels of all sizes (Figure 8.4A). The unsupervised version of PERSIST is the second most accurate method, achieving comparable explained variance for larger panels, and Cell Ranger performs similarly in this case.

The strong performance of PERSIST-Ephys is primarily due to it being tailored to electrophysiological characterization. PERSIST-Ephys is designed to address this specific predictive task, but the other methods rely on selection criteria that are not explicitly related to electrophysiology. As a result, these methods picked distinct genes (Figure F.10) and

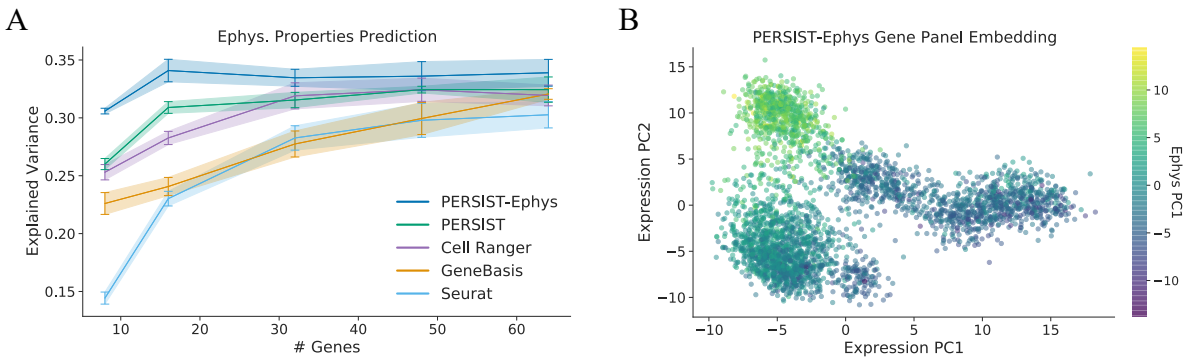


Figure 8.4: Using PERSIST to predict electrophysiological properties. PERSIST is adapted to this specific experimental goal by using electrophsyiological characteristics as its prediction target. The resulting version of our approach, PERSIST-Ephys, yields gene panels can predict cells' electrophysiological properties more accurately than other methods. **A**, Explained variance in the electrophysiological properties for gene panels selected by each method. Error bars represent 95% confidence intervals determined by training with five bootstrapped datasets **B**, Low-dimensional embedding for the PERSIST-Ephys panel containing 64 genes, colorized according to the cells' electrophysiological properties (using the first principal component); the cells are effectively clustered according to their properties, with similar cells appearing nearby in the embedding space.

PERSIST-Ephys is roughly as accurate with a panel of 8 genes as the other methods are with 64 genes. None of the gene panels we tested exceed 35% explained variance, but notably, we find that even the full set of 1,252 genes does not exceed this level of accuracy; this suggests that the unexplained variance represents noise in the experimental results, or factors that are not captured by gene expression.

As an exploratory analysis, we also examined a low-dimensional embedding of the PERSIST-Ephys gene panel to understand the relationship between cells that are nearby in expression space. Our plot displays individual cells using the first two principal components of the panel containing 64 genes (Figure 8.4B). To assess whether nearby cells have similar electrophysiological profiles, we colorized the cells by a scalar summary of the profile – the first principal component of the electrophysiological feature vector. The resulting plot reveals naturally occurring clusters of similar cells, which may be expected because PERSIST-Ephys selects

genes whose expression is maximally indicative of the neuron's electrophysiological profile.

8.3.5 Binarization enables gene expression prediction with MERFISH data

PERSIST can identify informative marker genes for a variety of experimental objectives, but our previous evaluations used only scRNA-seq data due to the challenge of providing an unbiased comparison via FISH studies conducted with multiple panels. Nevertheless, such cross-modal experiments represent an essential use case, which is applying predictive models trained using scRNA-seq to data collected from spatial transcriptomics studies. Here, it is important to verify that binarizing expression levels enables such models to transfer successfully between technologies, which is difficult to ascertain because accompanying annotations are seldom available for FISH datasets (e.g., ground truth cell type labels, or expression levels of genes that are not part of the FISH panel). To investigate this question, we therefore devised a multi-step *in silico* experiment using the SSv4 scRNA-seq dataset in combination with data from a recent, large-scale MERFISH study (Zhang et al., 2021).

In the MERFISH dataset, 258 genes were probed across 280,327 cells from the mouse primary motor cortex (MOp). Because ground truth cell type labels are not available, we instead chose to evaluate performance in the expressed gene prediction task (similar to Figure 8.2D-F), where our goal is to predict which individual genes are detected in each cell. To do so, we first used the V1 and ALM SSv4 scRNA-seq datasets to select panels of $8-32$ markers from within the Zhang et al. (2021) MERFISH gene set. Then, an imputation model was trained—using only the scRNA-seq data—to predict which of the remaining genes are detected. Finally, using the resulting model, we predicted the set of detected genes in the MERFISH dataset (Figure 8.5A). As in our previous experiments, we binarized both the scRNA-seq and MERFISH gene expression levels so the model trained with scRNA-seq data could transfer despite the measurement differences.

Encouragingly, we find that the scRNA-seq-trained models can predict expressed genes in the MERFISH data with high accuracy (Figure 8.5B-C). The prediction accuracy tends to improve with larger panels; for example, PERSIST reached 86.5% with a 32-gene panel

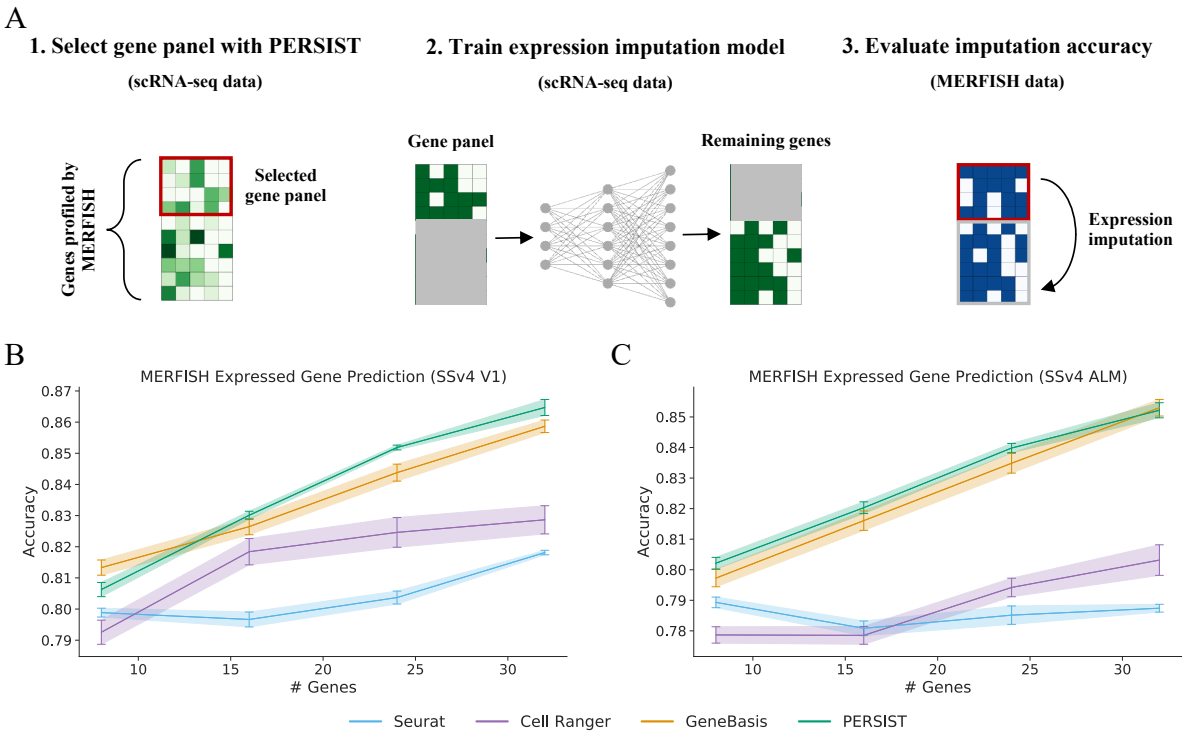


Figure 8.5: Predicting MERFISH gene expression after training with scRNA-seq data. To test whether models trained on scRNA-seq can transfer to FISH despite the domain shift between technologies, we design an experiment combining SSv4 and MERFISH. We find that binarizing gene expression levels enables accurate predictions in MERFISH cells, and that PERSIST outperforms prior gene selection approaches. **A**, In this multi-step experiment, a gene panel is first selected using scRNA-seq restricted to the genes profiled in the MERFISH data. Next, an imputation model is trained to predict whether the remaining genes are expressed, again using scRNA-seq data. Finally, the imputation model's accuracy is tested on MERFISH data, using the small gene panel to predict the remaining genes. The imputation model's high accuracy demonstrates that binarization enables successful transfer from scRNA-seq to FISH. **B**, Expressed gene prediction accuracy for panels selected by each method when using the V1 SSv4 data. **C**, Expressed gene prediction accuracy for panels selected by each method when using the ALM SSv4 data. All error bars represent 95% confidence intervals determined by training with five bootstrapped datasets

when trained on the V1 dataset. PERSIST in most cases achieves the highest accuracy, with GeneBasis being the most competitive baseline method. We also find that the prediction accuracy with the V1 cells is slightly higher than with the ALM cells. PERSIST not only

outperforms other unsupervised methods that can operate with only unlabeled scRNA-seq data, but also the approaches that leverage cell type labels when performing gene selection (Figure F.8). Crucially, the panels selected by all methods benefit from expression binarization to enable the imputation models to transfer across technologies.

In these experiments, we carefully determined the thresholds for binarizing the MERFISH expression counts by matching the quantile of the zero-threshold used with the scRNA-seq data. This procedure enables us to account for the zero-inflation present in scRNA-seq datasets. When we instead binarized the MERFISH data using a threshold value of zero, the prediction accuracy decreased for all gene panels (Figure F.8). Finally, because we have access to cells profiled using MERFISH with all genes in this case, we are able to determine that training the imputation model with MERFISH rather than scRNA-seq data results in an accuracy improvement of roughly $4-6\%$ (see Figure F.8). The gap is non-negligible, but it may be due in part to differences in the brain regions profiled in the reference dataset.

This evaluation approach does not represent a practical workflow for a real experiment, because the genes we predict are present in the panel; however, the fact that the model transferred successfully suggests that binarization can allow models for other predictive tasks to transfer from scRNA-seq to FISH data, including models for cell type classification or characterizing electrophysiological properties. This represents the first quantitative evidence, to our knowledge, that a predictive model trained exclusively with scRNA-seq data can be transferred successfully to a subsequent spatial transcriptomics study.

8.3.6 Variability in gene panel selections across algorithms

Because the gene selection algorithms tested here rely on diverse selection criteria, they produce different gene panels given the same reference scRNA-seq data. Here, we examine the overlap in gene panels selected by each algorithm, and we do so by calculating the proportion of overlapping genes within 32-gene panels chosen from among the 10,000 candidates in the SSv4 and 10X datasets. As expected, no two methods select the exact same gene set (Figure 8.6A-B), but there is overlap among many pairs of methods. The probability of two

random 32-gene panels sharing more than one gene is just 4.6×10^{-3}, so the overlap we observe suggests a shared reliance on a relatively small number of informative genes. The strongest similarity is between Seurat and Cell Ranger on the 10X dataset, at 53% overlap (17/32 genes); their overlap is lower with the SSv4 dataset, at just 22% (7/32 genes), and their similarity is perhaps due to the fact that both methods are based on per-gene variance levels. Another pair of similar methods is scGeneFit and GeneBasis, which have an overlap of 41% and 47% for the SSv4 and 10X datasets, respectively.

In comparison, we find that PERSIST has relatively low overlap with other methods. The highest overlap for PERSIST is with PERSIST-Classification and MutInfo: PERSIST shares 19% of genes with PERSIST-Classification on the 10X dataset and 16% on the SSv4 dataset, and with MutInfo it shares 16% on the 10X dataset and 19% on the SSv4 dataset. Meanwhile, PERSIST-Classification and MutInfo achieve higher levels of overlap, at 31% for the SSv4 dataset and 38% for the 10X dataset; similarly, PERSIST-Classification and SMaSH have 19% overlap on the SSv4 dataset and 28% overlap on the 10X dataset. For these three supervised methods, their similarity is likely due to their selection criteria that all aim to distinguish cell types. For panels containing either 16 or 128 genes, PERSIST's selections remain distinct from other methods, and they are still somewhat similar to those from PERSIST-Classification and MutInfo (Figure F.9). Overall, these results reflect that the various gene selection methods select distinct gene panels, and that PERSIST and PERSIST-Classification's improved performance across various metrics is enabled by the selection of substantially different panels.

Finally, we examined a somewhat unique characteristic of PERSIST: the stochasticity of its selections across runs. Because we implemented a deep learning model that is trained using stochastic gradient descent, the results from PERSIST and its supervised variants (PERSIST-Classification, PERSIST-Ephys) can differ across trials. This variability is somewhat unusual for a gene selection method, but this property is shared by other state-of-the-art feature selection techniques (Abid et al., 2019) and by the UMAP embedding method (McInnes et al., 2018). To examine the variability in the individual genes selected and the

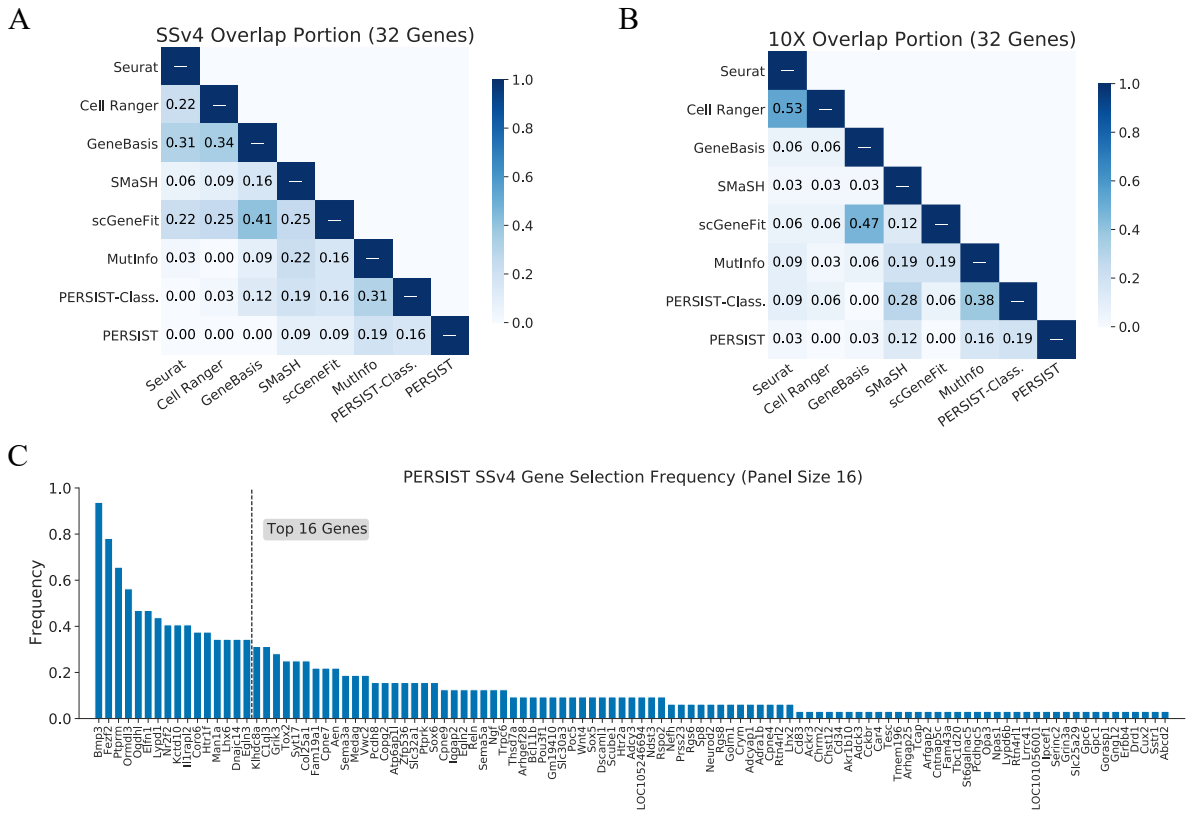


Figure 8.6: Diversity in gene panels selected by each method. Using the SSv4 and 10X datasets, we find that PERSIST and PERSIST-Classification select distinct gene panels from prior methods. Additionally, because PERSIST is non-deterministic across trials, we examine the frequency of its gene selections across multiple trials. **A**, Portion of overlapping genes between panels of 32 genes for the SSv4 cells. **B**, Portion of overlapping genes between panels of 32 genes for the 10X cells. **C**, Frequency of gene selections within PERSIST panels containing 16 genes, aggregated across 32 independent trials.

performance of the selected gene panels, we ran 32 independent trials of PERSIST with the SSv4 dataset.

For panels containing 16 genes, four genes were selected in at least half of the trials, with a single gene, *Bmp3*, being selected in 30 of 32 trials (Figure 8.6C). The remaining ones were selected less consistently; the sixteenth most frequently selected gene was chosen in just over a third of trials, and 38 genes were selected just once. Reassuringly, differences in the composition between the gene panels had little impact on cell type classification accuracy or

expression profile reconstruction (Figure F.11). The stability in performance despite changes in the gene panel composition is not surprising, because many genes have highly correlated expression patterns and thus can be readily substituted. In practice, we suggest running a small number of trials and then selecting the best trial using the validation loss achieved by the PERSIST deep learning model.

8.4 Discussion

Identifying an effective gene panel is a pre-requisite for successful spatial transcriptomics studies. This work introduces PERSIST, which uses deep learning to select genes that are highly predictive either for the genome-wide expression profile or for a specific experimental objective. Our experiments with several datasets show that PERSIST selects more informative targets than existing algorithms, generally providing better predictive accuracy and/or enabling the use of fewer genes. In addition to our deep learning-based selection mechanism, a key contribution of this work is PERSIST's robust inference ability, which is achieved by using binarized gene expression levels: this helps mitigate the complex relationship between scRNA-seq and FISH measurements, and we find that it allows models to transfer to FISH studies despite the challenging domain shift. We also note that while our explicit demonstration is based on a recent MERFISH dataset, the problems addressed here are common to a broader class of spatial transcriptomic methods (Qian et al., 2020; Sun et al., 2021). Therefore, our method is likely to improve target gene selection in a broader class of studies where genes are selected using surrogate data from a different technology.

From a computational perspective, PERSIST benefits from several aspects of deep learning that make it increasingly popular for data analysis in single-cell genomics (Lopez et al., 2018; Eraslan et al., 2019; Amodio et al., 2019). These include the simplicity of gradient-based optimization, the scalability to large datasets enabled by minibatched training, and the flexibility of the prediction target and loss function. We profiled PERSIST's computational cost and found that both the running time and memory usage remain manageable when using 10,000 candidate genes and selecting a relatively large panel of 256 genes (Fig-

ure 8.9). When using even larger datasets, PERSIST's computational cost can be managed by maintaining a smaller minibatch size, or by performing an initial filtering step to reduce the number of candidate genes. Finally, PERSIST does not require extensive tuning of parameters, and we used identical network architectures across all experiments (Table 8.1); however, future improvements may involve automatically setting all parameters to ensure maximum ease-of-use for both computational and biological users.

PERSIST is by default an unsupervised method that aims to reconstruct the genome-wide expression profile. Running it in an unsupervised fashion yields genes that are informative in general, but some information is necessarily sacrificed, because reconstructing the full expression profile using a small number of genes remains challenging. One interesting finding of our study is that while stochasticity in gene expression and detection limits the variance explained by the cell identity according to the reference clustering, relatively small PERSIST panels can explain more variance than the discrete cell type identity. This may suggest imperfections in the reference clustering, or biologically meaningful variability that lies in a continuum that is not captured by a discrete cell type label (Harris et al., 2018).

PERSIST can be tailored to arbitrary experimental goals by simply modifying the prediction target, thus enabling FISH studies to use gene expression as a bridge to other data modalities. Because alternative prediction tasks can be simpler than reconstructing many thousands of genes, using PERSIST in this fashion can enable the use of small gene panels that sacrifice minimal accuracy. Our experiments show examples with transcriptomic cell type classification and electrophysiological profile prediction, but other focused prediction tasks may include identifying disease properties or bridging with epigenetic information such as chromatin accessibility and methylation. Overall, PERSIST represents a powerful general-purpose solution for marker gene selection, and our design choices make it an effective tool for selecting small gene panels that can be used for FISH studies, and for spatial transcriptomics studies more generally.

8.5 Methods

8.5.1 Predictive and robust gene selection for spatial transcriptomics (PERSIST)

PERSIST aims to capture as much information as possible in a small gene panel, and it does so by selecting genes that can predict the genome-wide scRNA-seq expression profile. It relies on a deep learning model that reconstructs all the genes while using only a subset of the inputs, similar to an autoencoder but with a learned sparsity pattern. Deep learning-based reconstruction models have become popular for extracting low-dimensional embeddings in single-cell genomics (Lopez et al., 2018; Eraslan et al., 2019; Amodio et al., 2019), but PERSIST is designed to select a precise number of genes rather than fitting a general non-linear embedding. During the model's training, the input genes are selected by a custom network layer that enables training with stochastic gradient descent – a *binary mask layer*, which we describe below.

Deep learning is well suited to datasets with very large cell counts, and PERSIST's memory usage can be managed via the minibatch size used during training. In cases where memory usage is an issue, for example because the number of gene candidates and the intended panel size are both large, PERSIST can perform an initial filtering step using the *binary gates layer*, an alternative selection layer that we describe below. In addition to reducing memory usage, we find that such a two-stage approach provides minor performance improvements (Figure 8.8), and we use this approach for the majority of our experiments.

8.5.2 Hurdle loss function

Gene dropouts represent a significant source of noise in scRNA-seq data, and because PERSIST relies on scRNA-seq as a surrogate for FISH data, we must model gene dropouts appropriately when reconstructing the genome-wide expression profile. Recent computational tools for scRNA-seq data have shifted away from using mean squared error loss (Amodio et al., 2019; Arisdakessian et al., 2019) and towards zero-inflated models to account for this key source of noise (Pierson and Yau, 2015; Risso et al., 2018; Lopez et al., 2018; Eraslan

et al., 2019; Clivio et al., 2019). In line with these works, we propose a loss function that can be applied to zero-inflated and continuous-valued measurements, which arise from common normalization approaches like as counts per million (CPM) normalization (Luecken and Theis, 2019). A zero-inflated negative binomial (ZINB) loss can instead be used if CPM normalization is not applied, but variability in the total UMI counts per cell can make integer transcript counts difficult to predict.

When reconstructing the genome-wide scRNA-seq expression profile, PERSIST trains a model that outputs predictions for each target gene $i = 1, \ldots, d$. The predictions consist of a point prediction $\hat{y}_i$ as well as a probability $\hat{p}_i$ of the gene having non-zero expression. Given a fixed weighting parameter $\gamma > 0$, the hurdle loss $\ell_\gamma^{(i)}$ for gene i with observed expression level y_i is defined as

$$\ell_\gamma^{(i)}\big(\hat{p}_i, \hat{y}_i, y_i\big) = \begin{cases} \frac{1}{\gamma}(\hat{y}_i - y_i)^2 - \log \hat{p}_i & y_i > 0 \\ -\log(1 - \hat{p}_i) & y_i = 0. \end{cases} \tag{8.1}$$

The loss has a cross entropy component for predicting whether the gene is expressed or not, as well as a mean squared error component that is incorporated only when the gene is expressed ($y_i > 0$). Because we use log-normalized gene counts y_i, the loss $\ell_\gamma^{(i)}$ can be understood as the negative log-likelihood for a log-normal hurdle distribution (Finak et al., 2015; McDavid, 2016), where we implicitly simplify the log-normal component by assuming fixed standard deviations for each gene, and the weighting parameter $\gamma > 0$ controls the trade-off between predicting whether a gene is expressed and predicting its expression level. The total loss ℓ_γ for the full expression profile is the following, where $\hat{y}, \hat{p} \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$ represent vectors of predictions and expression levels for all genes:

$$\ell_\gamma\big(\hat{p}, \hat{y}, y\big) = \sum_{i=1}^{m} \ell_\gamma^{(i)}\big(\hat{p}_i, \hat{y}_i, y_i\big). \tag{8.2}$$

In practice, we fix $\gamma = 10$ so the mean squared error and cross entropy components of the loss have similar scale, but this parameter can also be tuned.

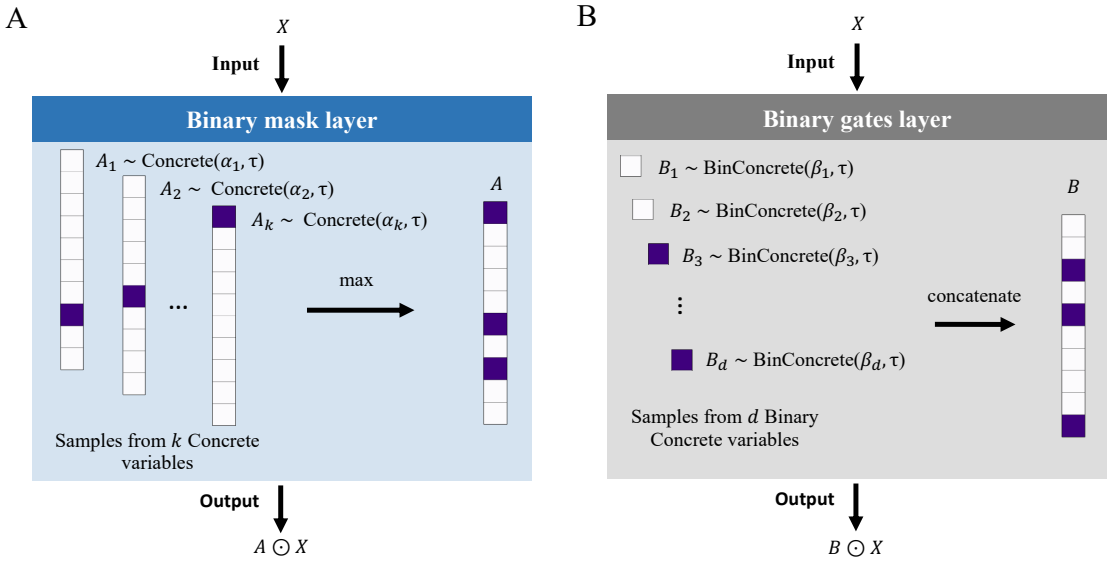


Figure 8.7: Feature selection layers. **A**, The binary mask layer multiplies its input with a learned, k-hot binary mask whose entries are the element-wise maximum of samples from k Concrete random variables. **B**, The binary gates layer multiplies the input with a learned binary mask whose entries are samples from d separate BinConcrete random variables.

8.5.3 Feature selection layers

We introduce the *binary mask layer* as a tool to select a user-specified number of inputs within a deep learning model. Our approach is based on the Concrete distribution (Maddison et al., 2016) (also known as the Gumbel-Softmax (Jang et al., 2016)), which lets us optimize discrete probability distributions using stochastic gradient descent. To select exactly k genes from d candidates, we multiply the model input by a binary mask generated using the element-wise maximum of k Concrete random variables, which are denoted as $\boldsymbol{a}_i \sim \text{Concrete}(\alpha_i, \tau)$ for $i = 1, \ldots, k$ (Figure 8.7). Each Concrete distribution is parameterized by unnormalized probabilities $\alpha_i \in \mathbb{R}_+^d$ and a temperature value $\tau > 0$, and each one converges to a multinomial distribution with probabilities given by $\frac{\alpha_i}{\mathbf{1}^\top \alpha_i}$ as $\tau \to 0$ (Maddison et al., 2016).

The binary mask layer's input is a vector of binarized gene expression levels $\mathbf{x} \in \mathbb{R}^d$, and its output is given by the element-wise product $\mathbf{a} \odot \mathbf{x}$, where $\mathbf{a} = \max_i \mathbf{a_i} \in \mathbb{R}^d$ is the

element-wise max of samples $\mathbf{a_i}$ from each Concrete random variable. The layer is followed by a neural network f_θ that predicts the label $\mathbf{y}$ given $\mathbf{a} \odot \mathbf{x}$, and we train the model by optimizing the following objective:

$$\min_{\theta,\alpha} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{a}}\Big[\ell_\gamma\big(f_\theta(\mathbf{a} \odot \mathbf{x}), \mathbf{y}\big)\Big]. \tag{8.3}$$

To select a specific number of genes, we need only ensure that the temperature parameter τ is sufficiently low at the end of training. In practice, we find that each Concrete random variable concentrates its probability mass on a single input, yielding a set of exactly k selected genes. The binary mask layer is similar to the CAE approach (Abid et al., 2019), but we find that our parameterization, which uses element-wise multiplication rather than a matrix multiplication, provides slightly better results (Figure 8.8).

We also introduce the *binary gates layer* as a memory-efficient alternative to the binary mask layer. Similar to previous work (Chang et al., 2017), we learn a Binary Concrete random variable for each input feature, denoted as $\boldsymbol{b}_i \sim \text{BinConcrete}(\beta_i, \tau)$, and we use these gate variables to perform element-wise multiplication with the corresponding genes (Figure 8.7). In the Binary Concrete distribution, each parameter $\beta_i > 0$ represents an unnormalized probability, $\tau > 0$ represents a temperature value, and each random variable $\boldsymbol{b}_i$ converges to a Bernoulli random variable with probability parameter $\frac{\beta_i}{1+\beta_i}$ as $\tau \to 0$ (Maddison et al., 2016).

The binary gates layer output is given by the element-wise product $\mathbf{b} \odot \mathbf{x}$, where $\mathbf{x}$ is the input and $\mathbf{b} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d] \in \mathbb{R}^d$ is a vector of samples from each random variable $\boldsymbol{b}_i \sim \text{BinConcrete}(\beta_i, \tau)$. The layer is followed by a neural network f_θ that predicts the label $\mathbf{y}$ given $\mathbf{b} \odot \mathbf{x}$. Genes are eliminated when we learn low β_i values, and we encourage this by augmenting the loss function with a penalty on the BinConcrete samples:

$$\min_{\theta,\beta} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{b}}\Big[\ell_\gamma\big(f_\theta(\mathbf{b} \odot \mathbf{x}), \mathbf{y}\big) + \lambda \mathbf{1}^\top \mathbf{b}\Big]. \tag{8.4}$$

The regularization term penalizes the number of selected genes, and the hyperparameter $\lambda > 0$ controls the trade-off between prediction accuracy and the number of genes used. Determining a specific number of genes requires choosing the correct λ value, and our implementation finds this value automatically by iteratively adjusting λ using the secant algorithm (Epperson, 2021). Briefly, given a desired number of candidate genes $d' < d$, our method iteratively updates the λ value based on the number of genes yielded by previous λ values. In our experiments with the SSv4 and 10X datasets, we initially narrow the set of candidates to roughly 500 genes. For the experiments involving Patch-seq and MERFISH data, which involve fewer candidate genes, we directly select gene panels using the binary mask layer.

8.5.4 Training

When training our deep learning models, we perform optimization using Adam with the standard learning rate (10^{-3}) (Kingma and Ba, 2014). Over the course of training, the temperature parameter τ is geometrically annealed from a high value to a low value to encourage discrete feature selection, similar to previous work (Abid et al., 2019). For both input layers, we use an initial temperature $\tau = 10.0$ and a final temperature of $\tau = 0.01$. After training, the parameters for the Concrete random variables are naturally learned such that the same genes are selected at every forward pass. That is, most Binary Concrete variables are deterministically equal to 0 or 1, and the Concrete variables are one-hot at the same entry in every sample. As a selection criterion for the binary gates layer, we retain all genes i such that $\frac{\beta_i}{1+\beta_i} > \frac{1}{2}$. For the binary mask layer, we select k genes using the maximum index of each vector of unnormalized probabilities α_i for $i = 1, \ldots, k$.

The binary mask layer is necessary for selecting a specific number of genes, but the binary gates layer is preferable for eliminating a large number of genes. The binary mask layer has $k \times d$ learnable parameters when selecting k genes, whereas the binary gates layer has only d. The binary mask layer can therefore be difficult to apply directly in scenarios where d and k are both large, and the binary gates layer is useful for our datasets with $d = 10{,}000$ total candidate genes (SSv4 and 10X). In practice, the outcome from running PERSIST may

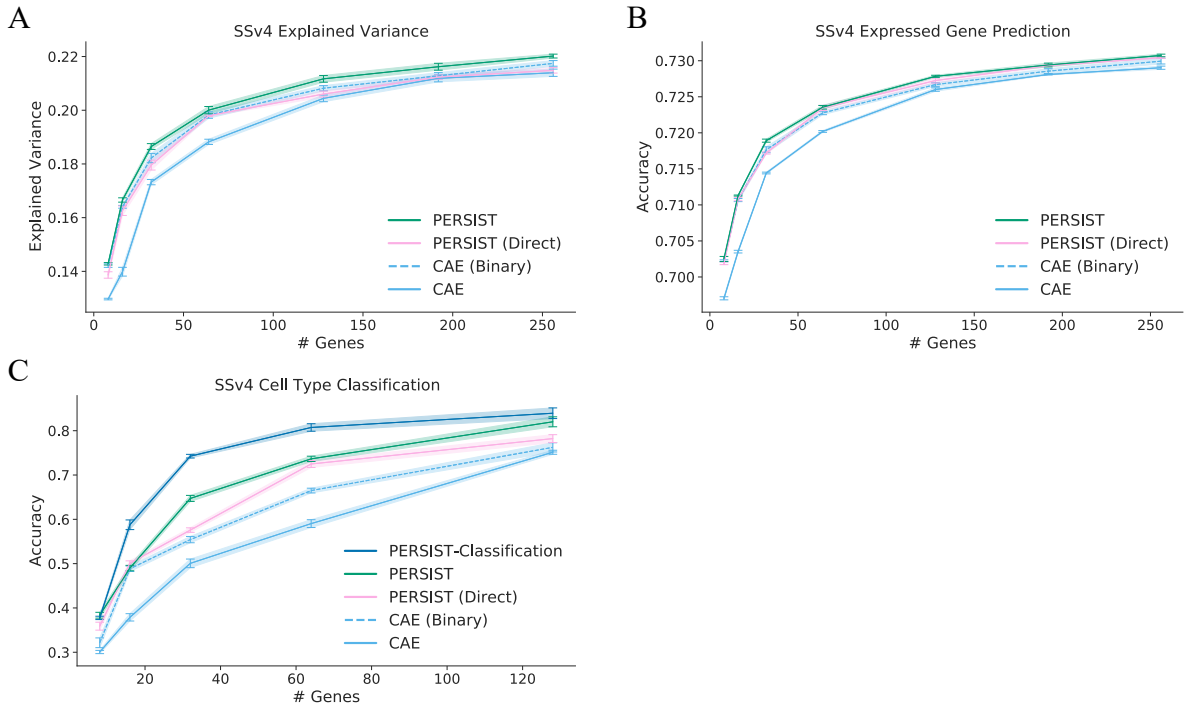


Figure 8.8: PERSIST ablations. PERSIST performs better than the CAE on expression profile reconstruction tasks and cell type classification. The performance gap narrows when the CAE uses binarized expression counts, but the difference in cell type classification accuracy remains large. PERSIST also performs slightly better when initially narrowing the set of candidate genes, rather than proceeding directly to training with the binary mask layer (PERSIST Direct). **A**, Explained variance for the SSv4 dataset. **B**, Expressed gene prediction accuracy for the SSv4 dataset. **C**, Cell type classification accuracy for the SSv4 dataset. All error bars represent 95% confidence intervals determined by training with five bootstrapped datasets.

differ across runs when the random seed is not fixed. For our experiments, we run five trials and use the gene panel that achieves the best validation loss.

8.5.5 Ablation experiments

We tested several variants of PERSIST to validate our design choices. First, we compared PERSIST to a version that skips the initial step of narrowing the set of candidate genes and proceeds directly to training with the binary mask layer. The results are slightly worse (Fig-

ure 8.8), which we attribute to the beneficial effect of iteratively reducing the number of genes. Next, we compared PERSIST to the CAE approach (Abid et al., 2019), which is closely related but differs in its choice of loss function (mean squared error) and feature selection layer. We find that the CAE underperforms PERSIST across our three evaluation metrics. Most noticeably, the CAE underperforms at cell type classification (Figure 8.8C), which is consistent with recent work that highlights the importance of modeling gene dropouts (Eraslan et al., 2019; Lopez et al., 2018). Finally, we observe that binarizing gene expression counts improves the CAE's performance on our metrics, but a small gap remains for expressed gene prediction, that the binarized CAE still does not match PERSIST's performance for cell type classification.

8.5.6 Memory and running time

To benchmark PERSIST's computational cost, we ran it across a variety of parameter settings and measured the running time and GPU memory usage. For simplicity, we used the SSv4 training set with 17,728 cells, we fixed the minibatch size to 128, and we selected panels by training directly with the binary mask layer for 500 epochs. Our models were all trained on a single NVIDIA GeForce RTX 2080 Ti. Figure 8.9 shows the results from varying two key parameters related to the dataset size. First, after fixing the panel size to 32, we varied the number of candidate genes from 2,000 to 10,000. The results show that the memory scales linearly, and the run-time scales sublinearly in the number of candidates. Next, after fixing the number of candidate genes to 10,000, we varied the panel size from 32 to 256. The results show that the memory scales linearly, and that the run-time scales roughly linearly. We did not test different numbers of cells because this does not affect memory usage, and the run-time would simply depend on how the number of epochs is tuned to each dataset size.

Across all the settings tested, PERSIST's run-time is not prohibitive. The GPU memory usage can become high when both the number of candidates and the panel size are large (e.g., 10,000 candidates and panel size of 256), but memory issues can be mitigated by either

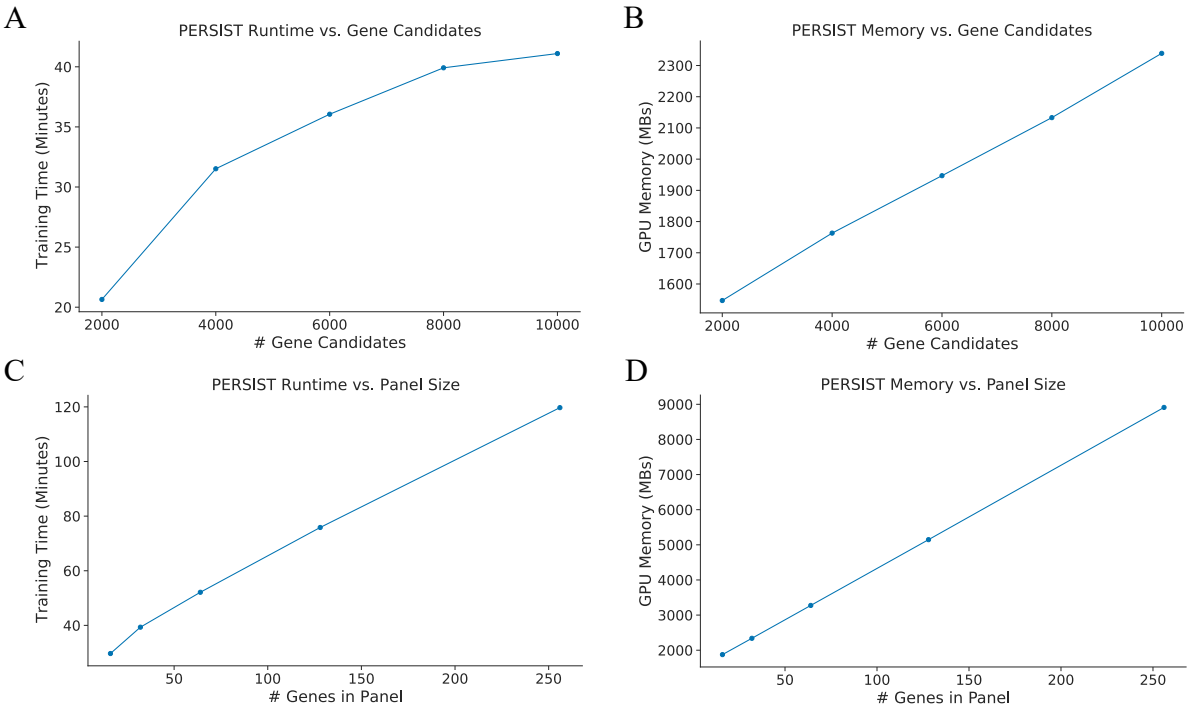


Figure 8.9: PERSIST run-time and memory usage. We varied two parameters that affect PERSIST's computational cost, the number of candidate genes (**A**-**B**) and the number of genes in the panel (**C**-**D**), and we then measured the run-time and GPU memory usage. **A**, Run-time versus the number of candidate genes, with the number of targets fixed to 32. **B**, Memory usage versus the number of candidate genes, with the number of targets fixed to 32. **C**, Run-time versus the panel size, with the number of candidates fixed to 10,000. **D**, Memory usage versus the panel size, with the number of candidates fixed to 10,000.

reducing the minibatch size or performing an initial filtering step using the binary gates layer. Compared to other gene selection methods, certain methods are faster than PERSIST because they do not require training a model (Seurat, Cell Ranger) or involve a simpler model that does not perform gene selection (SMaSH), but other methods can be slower, particularly with large datasets, because they involve expensive greedy heuristics (MutInfo, GeneBasis).

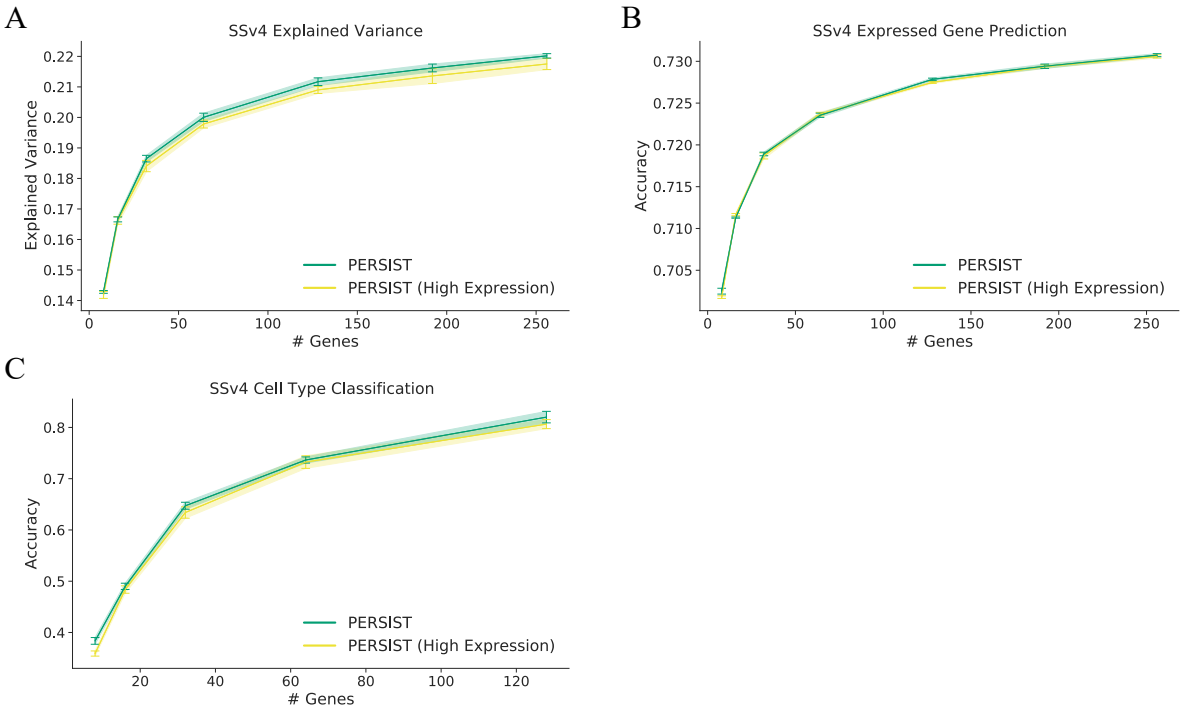


Figure 8.10: Using PERSIST with highly expressed genes only. PERSIST can incorporate expert knowledge, for example by considering only genes with high expression (whose maximum expression level is above the median); this helps ensures that transcripts are visible during the FISH experiment. The results show that even with this restriction on the gene candidates, PERSIST can identify informative gene panels and its performance is nearly identical. **A**, Explained variance for gene panels of different sizes with the SSv4 dataset. **B**, Expressed gene prediction accuracy with the SSv4 dataset. **C**, Cell type classification accuracy with the SSv4 dataset.

8.5.7 Expert knowledge and supervision

PERSIST can be used in a purely data-driven fashion, or it can be used while incorporating expert knowledge. For example, the candidate genes can be restricted to those with known biological function or other desirable properties, or the gene panel can be forced to include certain hand-selected genes. When such expert knowledge is applied to the set of candidate genes, PERSIST finds the best available panel among the current candidates and while accounting for any pre-selected genes.

Table 8.1: PERSIST hyperparameter choices.

Model	Description
PERSIST (SSv4, 10X, Patch-seq)	MLP (2×128 units, ReLU activations), mbsize 128, 500 epochs
PERSIST (SSv4 Sst)	MLP (2×128 units, ReLU activations), mbsize 1024, 4,000 epochs
PERSIST (SSv4 $\rightarrow$ MERFISH)	MLP (2×128 units, ReLU activations), mbsize 128, 250 epochs
PERSIST-Classification (SSv4, 10X)	MLP (2×128 units, ReLU activations), mbsize 1024, 500 epochs
PERSIST-Ephys (PatchSeq)	MLP (2×128 units, ReLU activations), mbsize 128, 250 epochs

For example, when working with specimens of lower RNA quality (e.g. post-mortem human samples), it can be useful to consider only highly expressed genes that can be readily detected. Figure 8.10 shows that when PERSIST is restricted to using only genes whose maximum expression level is higher than the median value, we achieve roughly the same performance as when choosing from among all 10,000 candidate genes in the SSv4 dataset. In general, filtering steps that preserve a large number of gene candidates should not prevent PERSIST from finding highly informative panels.

Rather than using PERSIST in an unsupervised manner, we can also incorporate supervision to align the gene panel with specific experimental aims. If the spatial transcriptomics study has a specific prediction objective, such as characterizing electrophysiological properties, and accompanying labels are available for the reference scRNA-seq data, PERSIST can incorporate this information into the selection procedure. Adapting PERSIST is straightforward, requiring only a change in the prediction target and loss function, as we demonstrate with PERSIST-Classification (multiclass cross entropy loss, see Figure 8.3) and PERSIST-Ephys (mean squared error loss, see Figure 8.4).

Table 8.2: Downstream task hyperparameter choices.

Task (Dataset)	Description
Cell type classification (SSv4, 10X)	LightGBM, learning rate 0.05, 10,000 boosters (with early stopping)
Explained variance (SSv4, SSv4 Sst, 10X)	MLP (2×128 units, ReLU activations), mbsize 512, 100 epochs (with early stopping)
Expressed gene prediction (SSv4, SSv4 Sst, 10X, SSv4 $\rightarrow$ MERFISH)	MLP (2×128 units, ReLU activations), mbsize 256, 100 epochs (with early stopping)
Electrophysiological properties (Patch-seq)	MLP (2×128 units, ReLU activations), mbsize 256, 100 epochs (with early stopping)

8.5.8 Models and hyperparameters

Each dataset is split into training, validation and test sets. The training and validation sets are used to select gene panels and train predictive models, and the test set is used only to evaluate the performance of trained models. Our hyperparameter choices were all made using the validation set, including for early stopping. For PERSIST and its supervised variants, we used identical neural network architectures across all datasets, and the only hyperparameters we adjusted are the number of training epochs and the minibatch size (Table 8.1). Our model choices for the various downstream tasks are shown in Table 8.2, where we use multi-layer perceptron (MLP) models for most tasks, and LightGBM models (Ke et al., 2017) for cell type classification. Confidence intervals for the downstream tasks were determined by training models with five bootstrapped training sets and measuring the test set performance across these models.

When using PERSIST, overfitting can be avoided by using network architectures that are not too large and performing early stopping during training. If overfitting is a significant concern, additional regularization techniques are straightforward to incorporate, including dropout, batch norm and ridge or lasso regularization. Finally, when fitting models for the downstream predictive tasks, such as cell type classification, one can further mitigate

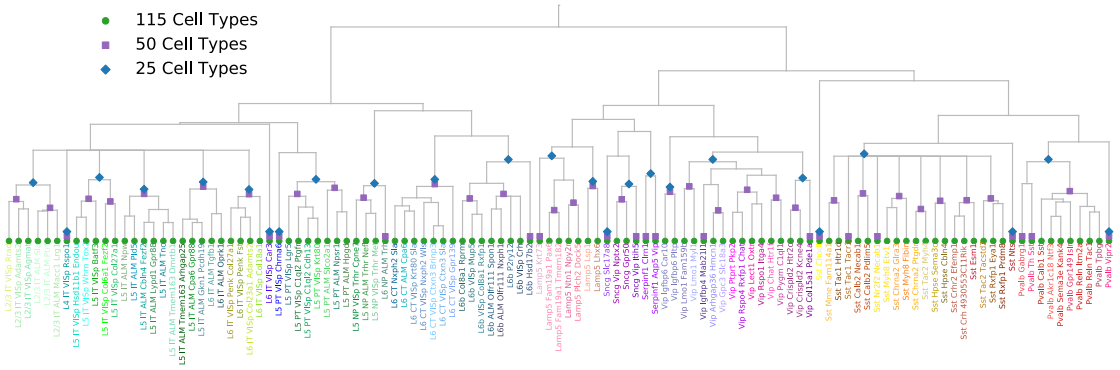


Figure 8.11: Transcriptomic hierarchy for the SSv4 V1/ALM neuronal cells. The cell types for this dataset are defined by a transcriptomic hierarchy containing 115 total cell types Tasic et al. (2018). To simplify the cell type classification task, we consider merging cell types according to the transcriptomic hierarchy, yielding 50 or 25 subclasses that are more clearly defined.

overfitting by using a non-neural network model with fewer learnable parameters (Grabski and Irizarry, 2022).

8.5.9 Evaluation metrics

The variance explained by a gene panel is measured by training a neural network to predict the full expression profile, measuring the mean squared error on the test set, and subtracting this quantity from the total variance. We provide results on an intuitive scale by calculating explained variance as a portion of total variance. For this metric, we used CPM normalized and logarithmized expression counts as the prediction target. The portion of explained variance does not approach 100% for any method, but this is in large part due to the stochasticity of gene expression and measurement; as described in the previous section, the ground truth cell type labels explain only 19% and 11% of the variance for the SSv4 and 10X datasets, respectively.

The cell type classification performance is the accuracy of a gradient-boosted decision tree (GBDT) model trained with a multi-class cross entropy loss. The cell type with the

highest classification probability is taken as the predicted class. To evaluate performance with cell subclasses for the SSv4 dataset, we trained models with cell types merged according to their order in the transcriptomic hierarchy (Figure 8.11).

Expressed gene prediction accuracy is measured using a neural network trained to separately predict whether each gene is expressed. Similar to a classifier model, the loss function is a per-gene cross entropy loss that is then added across genes. A gene is predicted to be expressed if the network's probability exceeds 0.5, and we calculate the accuracy by calculating how often the predictions agree with the true expression, and averaging the results across all target genes.

8.5.10 Data pre-processing

Due to technical noise in scRNA-seq data, we applied CPM normalization (Luecken and Theis, 2019) to the raw measurements and then applied the log1p operation. These values serve as prediction targets for PERSIST, whereas the inputs to our various models are binarized expression counts. For both the SSv4 and 10X datasets, we restricted our analysis to the 10,000 transcripts with the highest variance. We used exon counts for the high-resolution SSv4 dataset and the sum of intron and exon counts for the lower resolution 10X data. For the MERFISH dataset, the only pre-processing we applied was gene expression binarization. When working with the MERFISH data, we analyze 253 genes that appear in both the Zhang et al. (Zhang et al., 2021) and SSv4 datasets rather than the 258 genes described in the original work.

8.5.11 Expression quantization

We quantized gene expression levels in order to train models using scRNA-seq data that can transfer to FISH studies. There is a complex domain shift between the two technologies, but their quantized gene expression values should be similar if we assume that the transformation between measurements is monotonic. When applying models trained with scRNA-seq on FISH data in practice, we recommend using a threshold matching approach, i.e., finding the

quantile that the scRNA-seq threshold represents in the scRNA-seq measurements (we use a threshold value of zero), and then identifying the matching threshold in the FISH data. This approach is used for the results in Figure 8.5, and its utility is demonstrated in Figure F.8.

8.5.12 Baseline methods

For the Seurat and Cell Ranger gene selection protocols, we used the implementations available in the Scanpy (Wolf et al., 2018) package.[1] Seurat was run with raw gene expression counts and Cell Ranger with logarithmized counts. For the scGeneFit method, we used the authors' implementation with logarithmized expression counts, and we used the default hyperparameters.[2] For the GeneBasis method, we used the authors' R implementation.[3] For the SMaSH method, we used the authors' Python implementation with a neural network architecture identical to PERSIST, and with feature importance scores calculated using DeepSHAP (Lundberg and Lee, 2017).[4] Finally, for the MutInfo method, we implemented the greedy forward selection algorithm described in prior work (Battiti, 1994; Li et al., 2017b) using the hyperparameter $\beta = 1$ to account for gene correlations.

For the panel of marker genes used in Figure 8.3B, we used a set of genes identified in various tables of Tasic et al. (2018). The original work listed 77 such markers, and we used the 59 that were represented in our dataset after narrowing it to 10,000 high-variance genes.

8.5.13 Statistics & reproducibility

The samples from each data source were assigned at random to training, validation and test splits, and to obtain unbiased statistics we used the test data only when calculating evaluation metrics. No data was excluded from the datasets. Uncertainty estimates were provided throughout the experiments by fitting models with boostrapped training sets. Code

[1] `https://github.com/theislab/scanpy`

[2] `https://github.com/solevillar/scGeneFit-python`

[3] `https://github.com/MarioniLab/geneBasisR`

[4] `https://gitlab.com/cvejic-group/smash`

Table 8.3: Summary of datasets.

Species	Brain region	Technology	Num. Genes	Num Cells.	Annotations Used
Mouse	Primary visual (V1), anterior lateral motor (ALM)	SmartSeq v4	45,768 (10,000 in experiments)	22,160 neuronal, 2,701 Sst	Transcriptomic cell types (115 total)
Human	Motor cortex (M1)	10X	50,281 (10,000 in experiments)	72,629	Transcriptomic cell types (117 total)
Mouse	Primary visual (V1)	Patch-seq	1,252	3,411	Electrophysiological profiles (68 features)
Mouse	Primary motor (MOp)	MERFISH	258 (253 in experiments)	280,327	None

for running our method is available online (see "Code availability"), and the various datasets are available to download online (see "Data availability").

8.5.14 Data availability

The datasets used in this work are summarized in Table 8.3, including the species, brain regions, and annotations used for our experiments. The V1/ALM SmartSeq mouse neocortex data is available at `https://portal.brain-map.org/atlases-and-data/rnaseq/mouse-v1-and-alm-smart-seq`. The M1 10X data is available at `https://portal.brain-map.org/atlases-and-data/rnaseq/human-m1-10x`. The Patch-seq data is available at `https://github.com/AllenInstitute/coupledAE-patchseq`. The MOp MERFISH data is available at `https://download.brainimagelibrary.org/02/26/02265ddb0dae51de/`.

8.5.15 Code availability

Source code for PERSIST is available online,[5] along with tutorial notebooks and examples of data pre-processing code. The repository contains a list of software dependencies, and PERSIST is implemented in PyTorch (version 1.13.1).

[5] `https://github.com/iancovert/persist/`

Chapter 9

MAXIMIZING MUTUAL INFORMATION FOR DYNAMIC FEATURE SELECTION

9.1 Chapter Notes

This chapter presents joint work with Wei Qiu, Mingyu Lu, Nayoon Kim, Nathan White and Su-In Lee that was published in the International Conference on Machine Learning (ICML) in 2023. The content of this chapter is a minor variant of the published version of the paper. The main contribution of this work is to develop a method for dynamically selecting features at inference time based on their predictive utility, or their conditional mutual information with the response variable.

9.2 Introduction

A machine learning model's inputs can be costly to obtain, and feature selection is often used to reduce data acquisition costs. In applications where information is gathered sequentially, a natural option is to select features adaptively based on the currently available information, rather than using a fixed feature set. This setup is known as *dynamic feature selection* (DFS),[1] and the problem has been considered by several works in the last decade (Saar-Tsechansky et al., 2009; Dulac-Arnold et al., 2011; Chen et al., 2015c; Early et al., 2016a; He et al., 2016a; Kachuee et al., 2018).

Compared to *static* feature selection with a fixed feature set (Li et al., 2017a; Cai et al., 2018), DFS can offer better performance given a fixed budget. This is easy to see, because selecting the same features for all instances (e.g., all patients visiting a hospital's emergency

[1]The problem has also been referred to as *sequential feature selection*, *active feature acquisition*, and *information pursuit*.

room) is suboptimal when the most informative features vary across individuals. Although it should in theory offer better performance, DFS also presents a more challenging learning problem, because it requires learning both (i) a feature selection policy and (ii) how to make predictions given variable feature sets.

Prior work has approached DFS in several ways, though often using reinforcement learning (RL) (Dulac-Arnold et al., 2011; Shim et al., 2018; Kachuee et al., 2018; Janisch et al., 2019; Li and Oliva, 2021). RL is a natural approach for sequential decision-making problems, but current methods are difficult to train and do not reliably outperform static feature selection methods (Henderson et al., 2018; Erion et al., 2021). Our work therefore explores a simpler approach: greedily selecting features based on their conditional mutual information (CMI) with the response variable.

The greedy approach is known from prior work (Chen et al., 2015c; Ma et al., 2019) but is difficult to use in practice, because calculating CMI requires oracle access to the data distribution (Cover and Thomas, 2012). Our focus is therefore developing a practical approximation. Whereas previous work makes strong assumptions about the data (Geman and Jedynak, 1996) or approximates the data distribution with generative modeling (Ma et al., 2019), we develop a flexible approach that directly predicts the optimal selection at each step. Our method is based on a variational perspective on the greedy CMI policy, and it uses a technique known as *amortized optimization* (Amos, 2022) to enable training using only a standard labeled dataset. Notably, the model is trained with an objective that recovers the greedy policy when it is trained to optimality.

Our contributions in this work are the following:

1. We derive a variational, or optimization-based perspective on the greedy CMI policy, which shows it to be equivalent to minimizing the one-step-ahead prediction loss given an optimal classifier.

2. We develop a learning approach based on amortized optimization, where a policy network is trained to directly predict the greedy selection at each step. Rather than

requiring a dataset that indicates the correct selections, our training approach is based on a standard labeled dataset and an objective function whose global optimizer is the greedy CMI policy.

3. We propose a continuous relaxation for the inherently discrete learning objective, which enables efficient and architecture-agnostic gradient-based optimization.

Our experiments evaluate the proposed method on numerous datasets, and the results show that it outperforms many recent dynamic and static feature selection methods. Overall, our work shows that when learned properly, the greedy CMI policy is a simple and powerful method for DFS.

9.3 Problem Formulation

In this section, we describe the DFS problem and introduce notation used throughout this chapter.

9.3.1 Notation

Our goal is to design a policy that controls which features are selected given the currently available information. The selection policy can be viewed as a function $\pi(x_S) \in [d]$, meaning that it receives a subset of features as its input and outputs the next feature index to query. The policy is accompanied by a predictor $f(x_S)$ that can make predictions given the set of available features; for example, if $\boldsymbol{y}$ is discrete then predictions lie in the probability simplex, or $f(x_S) \in \Delta^{K-1}$ for K classes. As a shorthand, the notation $f(x_S \cup x_i)$ represents the prediction given the combined features. We initially consider policy and predictor functions that operate on feature subsets, and Section 9.5 proposes an implementation using a mask variable $m \in [0, 1]^d$ where the functions operate on a masked input $x \odot m$.

9.3.2 Dynamic Feature Selection

The goal of DFS is to select features with minimal budget that achieve maximum predictive accuracy. Having access to more features generally makes prediction easier, so the challenge is selecting a small number of informative features. There are multiple formulations for this problem, including non-uniform feature costs and different budgets for each sample (Kachuee et al., 2018), but we focus on the setting with a fixed budget and uniform costs. Our goal is to handle data samples at inference time by beginning with no features, sequentially selecting features x_S such that $|S| = k$ for a fixed budget $k < d$, and finally make an accurate prediction for the response variable y.

Given a loss function that measures the discrepancy between predictions and labels $\ell(\hat{y}, y)$, a natural scoring criterion is the expected loss after selecting k features. The scoring is applied to a policy-predictor pair (π, f), and we define the score for a fixed budget k as follows,

$$v_k(\pi, f) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \left[\ell\big(f\big(\{\boldsymbol{x}_{i_t}\}_{t=1}^{k}\big), \boldsymbol{y}\big)\right], \tag{9.1}$$

where feature indices are chosen sequentially for each $(\boldsymbol{x}, \boldsymbol{y})$ according to $i_n = \pi(\{\boldsymbol{x}_{i_t}\}_{t=1}^{n-1})$. The goal is to minimize $v_k(\pi, f)$, or equivalently, to maximize our final predictive accuracy.

One approach is to frame this as a Markov decision process (MDP) and solve it using standard RL techniques, so that π and f are trained to optimize a reward function based on eq. 9.1. Several recent works have designed such formulations (Shim et al., 2018; Kachuee et al., 2018; Janisch et al., 2019; Li and Oliva, 2021). However, these approaches are difficult to train effectively, so our work focuses on a greedy approach that is easier to learn and simpler to interpret.

9.4 Greedy Information Maximization

This section first defines the greedy CMI policy, and then describes an existing approximation strategy that relies on generative models.

9.4.1 The Greedy Selection Policy

As an idealized approach to DFS, we are interested in the greedy algorithm that selects the most informative feature at each step. This feature can be defined in multiple ways, but we focus on the information-theoretic perspective that the most useful feature has maximum CMI with the response variable (Cover and Thomas, 2012). The CMI, denoted as $I(\boldsymbol{x}_i; \boldsymbol{y} \mid x_S)$, quantifies how much information an unknown feature $\boldsymbol{x}_i$ provides about the response $\boldsymbol{y}$ when accounting for the current features x_S, and it is defined as the KL divergence between the joint and factorized distributions:

$$I(\boldsymbol{x}_i; \boldsymbol{y} \mid x_S) = D_{\mathrm{KL}}\big(p(\boldsymbol{x}_i, \boldsymbol{y} \mid x_S) \mid\mid p(\boldsymbol{x}_i \mid x_S)p(\boldsymbol{y} \mid x_S)\big).$$

Based on this, we define the greedy CMI policy as $\pi^*(x_S) \equiv \arg\max_i I(\boldsymbol{x}_i; \boldsymbol{y} \mid x_S)$, so that features are sequentially selected to maximize our information about the response variable. We can alternatively understand the policy as performing greedy uncertainty minimization, because this is equivalent to minimizing $\boldsymbol{y}$'s conditional entropy at each step, or $\pi^*(x_S) = \arg\min_i H(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)$ (Cover and Thomas, 2012). For a complete characterization of this idealized approach, we also consider that the policy is paired with the Bayes classifier as a predictor, or $f^*(x_S) = p(\boldsymbol{y} \mid x_S)$.

Maximizing the information about $\boldsymbol{y}$ at each step is intuitive and should be effective in many problems. Prior work has considered the same idea, but from two perspectives that differ from ours. First, Chen et al. (2015c) take a theoretical perspective and prove that the greedy algorithm achieves performance within a multiplicative factor of the optimal policy; the proof requires specific distributional assumptions, but we find that the greedy algorithm performs well with many real-world datasets. Second, from an implementation perspective, two works aim to provide practical approximations; however, these suffer from several limitations, so our work aims to develop a simple and flexible alternative. In these works, Ma et al. (2019) and Chattopadhyay et al. (2022) both require a conditional generative model of the data distribution, which we discuss next.

9.4.2 Estimating Conditional Mutual Information

The greedy policy is trivial to implement if we can directly calculate the CMI, but this is rarely the case in practice. Instead, one option is to estimate it. We now describe a procedure to do so iteratively for each feature, assuming for now that we have oracle access to the response distributions $p(\boldsymbol{y} \mid \boldsymbol{x}_S)$ for all $S \subseteq [d]$ and the feature distributions $p(\boldsymbol{x}_i \mid \boldsymbol{x}_S)$ for all $S \subseteq [d]$ and $i \in [d]$.

At any point in the selection procedure, given the current features x_S, we can estimate the CMI for a feature $\boldsymbol{x}_i$ where $i \notin S$ as follows. First, we can sample multiple values for $\boldsymbol{x}_i$ from its conditional distribution, or $x_i^j \sim p(\boldsymbol{x}_i \mid x_S)$ for $j \in [n]$. Next, we can generate Bayes optimal predictions for each sampled value, or $p(\boldsymbol{y} \mid x_S, x_i^j)$. Finally, we can calculate the mean prediction and the mean KL divergence relative to the mean prediction, which yields the following CMI estimator:

$$I_i^n = \frac{1}{n} \sum_{j=1}^{n} D_{\mathrm{KL}}\Big(p(\boldsymbol{y} \mid x_S, x_i^j) \;||\; \frac{1}{n} \sum_{l=1}^{n} p(\boldsymbol{y} \mid x_S, x_i^l)\Big). \tag{9.2}$$

This score measures the variability among predictions and captures whether different $\boldsymbol{x}_i$ values significantly affect $\boldsymbol{y}$'s conditional distribution. The estimator can be used to select features, or we can set $\pi(x_S) = \arg\max_i I_i^n$, due to the following limiting result (see Appendix G.1):

$$\lim_{n \to \infty} I_i^n = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S). \tag{9.3}$$

This procedure thus provides a way to identify the correct greedy selections by estimating the CMI. Prior work has explored similar ideas for scoring features based on sampled predictions (Saar-Tsechansky et al., 2009; Chen et al., 2015b; Early et al., 2016a,b), but the implementation choices in these works prevent them from performing greedy information maximization. In eq. 9.2, is it important that our estimator uses the Bayes classifier, that we sample features from the conditional distribution $p(\boldsymbol{x}_i \mid x_S)$, and that we use the KL divergence as a measure of prediction variability. However, this estimator is impractical because

we typically lack access to both $p(\boldsymbol{y} \mid \boldsymbol{x}_S)$ and $p(\boldsymbol{x}_i \mid \boldsymbol{x}_S)$.

In practice, we would instead require learned substitutes for each distribution. For example, we can use a a classifier that approximates $f(x_S) \approx p(\boldsymbol{y} \mid x_S)$ and a generative model that approximates samples from $p(\boldsymbol{x}_i \mid \boldsymbol{x}_S)$. Similarly, Ma et al. (2019) propose jointly modeling $(\boldsymbol{x}, \boldsymbol{y})$ with a conditional generative model, which is implemented via a modified VAE (Kingma et al., 2015). This approach is limited for several reasons, including (i) the difficulty of training an accurate conditional generative model, (ii) the challenge of modeling mixed continuous/categorical features (Ma et al., 2020; Nazabal et al., 2020), and (iii) the slow CMI estimation process. In our approach, which we discuss next, we bypass all three of these challenges by directly predicting the best selection at each step.

9.5 Proposed Method

We now introduce our approach, a practical approximation of the greedy policy trained using amortized optimization. Unlike prior work that estimates the CMI as an intermediate step, we develop a variational perspective on the greedy policy, which we then leverage to train a network that directly predicts the optimal selection given the current features.

9.5.1 A Variational Perspective on CMI

For our purpose, it is helpful to recognize that the greedy policy can be viewed as the solution to an optimization problem. Section 9.4 provides a conventional definition of CMI as a KL divergence, but this is difficult to integrate into an end-to-end learning approach. Instead, we now consider the one-step-ahead prediction achieved by a policy π and predictor f, and we determine the behavior that minimizes their loss. Given the current features x_S and a selection $i = \pi(x_S)$, the expected one-step-ahead loss is:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)} \Big[\ell\big(f(x_S \cup \boldsymbol{x}_i), \boldsymbol{y}\big) \Big]. \tag{9.4}$$

The variational perspective we develop here consists of two main results regarding this expected loss. The first result concerns the predictor, and we show that the loss-minimizing predictor can be defined independently of the policy π. We formalize this in the following proposition for classification tasks, and our results can also be generalized to regression tasks (see proofs in Appendix G.1).

Proposition 9.1. *When $\boldsymbol{y}$ is discrete and ℓ is cross entropy loss, eq.* 9.4 *is minimized for any policy π by the Bayes classifier, or $f^*(x_S) = p(\boldsymbol{y} \mid x_S)$.*

The above property requires that features are selected without knowledge of the remaining features or response variable, which is a valid assumption for DFS, but not in scenarios where selections are based on the full feature set (Chen et al., 2018a; Yoon et al., 2018; Jethani et al., 2021a). Now, assuming that we use the Bayes classifier f^* as a predictor, our second result concerns the selection policy. As we show next, the loss-minimizing policy is equivalent to making selections based on CMI.

Proposition 9.2. *When $\boldsymbol{y}$ is discrete, ℓ is cross entropy loss and the predictor is the Bayes classifier f^*, eq.* 9.4 *is minimized by the greedy CMI policy, or $\pi^*(x_S) = \arg\max_i \; I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$.*

With this, we can see that the greedy CMI policy defined in Section 9.4 is equivalent to minimizing the one-step-ahead prediction loss. Next, we exploit this variational perspective to develop a joint learning procedure for a policy and predictor network.

9.5.2 An Amortized Optimization Approach

Instead of estimating each feature's CMI to identify the next selection, we now develop an approach that directly predicts the best selection at each step. The greedy policy implicitly requires solving an optimization problem for each selection, or $\arg\max_i I(\boldsymbol{y}, \boldsymbol{x}_i; x_S)$, but since we lack access to this objective, we now formulate an approach that directly predicts the solution. Following a technique known as amortized optimization (Amos, 2022), we do so

by casting our variational perspective on CMI from above as an objective function to be optimized by a learnable network.

First, because it facilitates gradient-based optimization, we now consider that the policy outputs a distribution over feature indices. With slight abuse of notation, this section lets the policy be a function $\pi(x_S) \in \Delta^{d-1}$, which generalizes the previous definition $\pi(x_S) \in [d]$. Using this stochastic version of the policy, we can now formulate our objective function as follows.

Let the selection policy be parameterized by a neural network $\pi(\boldsymbol{x}_S; \phi)$ and the predictor by a neural network $f(\boldsymbol{x}_S; \theta)$. Let $p(\boldsymbol{S})$ represent a distribution with support over all subsets, or $p(S) > 0$ for all $|S| < d$. Then, our objective function $\mathcal{L}(\theta, \phi)$ is defined as

$$\mathcal{L}(\theta, \phi) = \mathop{\mathbb{E}}_{p(\boldsymbol{x}, \boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[\mathop{\mathbb{E}}_{i \sim \pi(\boldsymbol{x}_{\boldsymbol{S}}; \phi)} \big[\ell\big(f(\boldsymbol{x}_{\boldsymbol{S}} \cup \boldsymbol{x}_i; \theta), \boldsymbol{y}\big) \big] \Big]. \tag{9.5}$$

Intuitively, eq. 9.5 represents generating a random feature set $\boldsymbol{x}_S$, sampling a feature index according to $i \sim \pi(\boldsymbol{x}_S; \phi)$, and then measuring the loss of the prediction $f(\boldsymbol{x}_S \cup \boldsymbol{x}_i; \theta)$. Our objective thus optimizes for individual selections and predictions rather than the entire trajectory, which lets us build on Propositions 9.1 and 9.2. We describe this as an implementation of the greedy approach because it recovers the greedy CMI selections when it is trained to optimality. In the classification case, we show the following result under a mild assumption that there is a unique optimal selection.

Theorem 9.1. *When $\boldsymbol{y}$ is discrete and ℓ is cross entropy loss, the global optimum of eq.* 9.5 *is a predictor that satisfies $f(x_S; \theta^*) = p(\boldsymbol{y} \mid x_S)$ and a policy $\pi(x_S; \phi^*)$ that puts all probability mass on $i^* = \arg\max_i I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$.*

If we relax the assumption of a unique optimal selection, the optimal policy $\pi(\boldsymbol{x}_S; \phi^*)$ simply splits probability mass among the best indices. A similar result holds in the regression case, where we can interpret the greedy policy as performing conditional variance minimization.

Theorem 9.2. *When $\boldsymbol{y}$ is continuous and ℓ is squared error loss, the global optimum of eq. 9.5 is a predictor that satisfies $f(x_S; \theta^*) = \mathbb{E}[\boldsymbol{y} \mid x_S]$ and a policy $\pi(x_S; \phi^*)$ that puts all probability mass on $i^* = \arg\min_i \mathbb{E}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)]$.*

Proofs for these results are in Appendix G.1. Note that the function class for each model must be expressive enough to contain their respective optima, and that the result holds for any $p(\boldsymbol{S})$ with support over all subsets.

This approach has two key advantages over the CMI estimation procedure from Section 9.3. First, we avoid modeling the feature conditional distributions $p(\boldsymbol{x}_i \mid \boldsymbol{x}_S)$ for all (S, i). Modeling these distributions is a difficult intermediate step, and our approach instead aims to directly output the optimal index. Second, our approach is faster because each selection is made in a single forward pass: selecting k features using the procedure from Ma et al. (2019) requires $\mathcal{O}(dk)$ scoring steps, but our approach requires only k forward passes through the policy network $\pi(\boldsymbol{x}_S; \phi)$.

Furthermore, compared to a policy trained by RL, the greedy approach is easier to learn. Our training procedure can be viewed as a form of reward shaping (Sutton et al., 1998; Randløv and Alstrøm, 1998), where the reward accounts for the loss after each step and provides a strong signal about whether each selection is helpful. In comparison, observing the reward only after selecting k features provides a comparably weak signal to the policy network. RL methods generally face a challenging exploration-exploitation trade-off, but learning the greedy policy is simpler because it only requires finding the locally optimal choice at each step.

9.5.3 Training With a Continuous Relaxation

Our objective in eq. 9.5 yields the correct greedy policy when it is perfectly optimized, but $\mathcal{L}(\theta, \phi)$ is difficult to optimize by gradient descent. In particular, gradients are difficult to propagate through the policy network given a sampled index $i \sim \pi(\boldsymbol{x}_S; \phi)$. The REINFORCE trick (Williams, 1992) is one way to get stochastic gradients, but high gradient variance

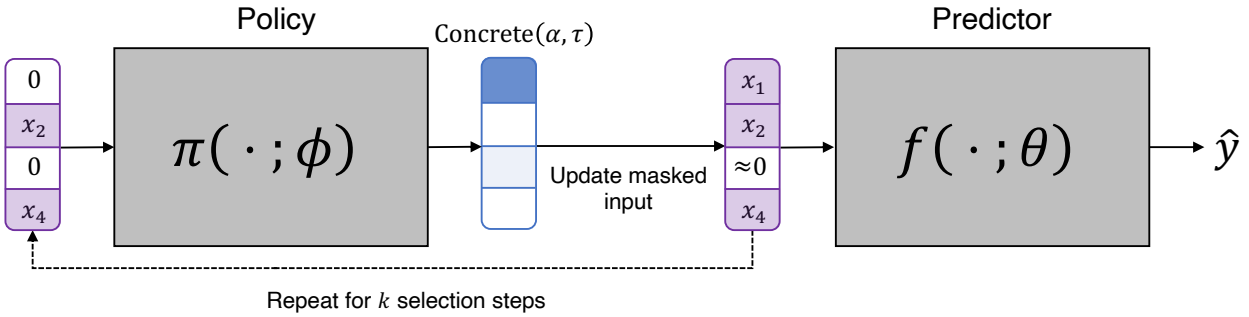


Figure 9.1: Diagram of our training approach. Left: features are selected by making repeated calls to the policy network using masked inputs. Right: predictions are made after each selection using the predictor network. Only solid lines are backpropagated through when performing gradient descent.

can make it ineffective in many problems. There is a robust literature on reducing gradient variance in this setting (Tucker et al., 2017; Grathwohl et al., 2018), but we propose using a simple alternative: the Concrete distribution (Maddison et al., 2016).

An index sampled according to $i \sim \pi(x_S; \phi)$ can be represented by a one-hot vector $m \in \{0, 1\}^d$ indicating the chosen index, and with the Concrete distribution we instead sample an *approximately* one-hot vector in the probability simplex, or $m \in \Delta^{d-1}$. This continuous relaxation lets us calculate gradients using the reparameterization trick (Maddison et al., 2016; Jang et al., 2016). Relaxing the subset $S \subseteq [d]$ to a continuous vector also requires relaxing the policy and predictor functions, so we let these operate on a masked input x, or the element-wise product $x \odot m$. To avoid ambiguity about whether features are zero or masked, we can also pass the mask as a model input.

Training with the Concrete distribution requires specifying a temperature parameter $\tau > 0$ to control how discrete the samples are. Previous works have typically trained with a fixed temperature or annealed it over a pre-determined number of epochs (Chang et al., 2017; Chen et al., 2018a; Balın et al., 2019), but we instead train with a sequence of τ values and perform early stopping at each step. This removes the temperature and number of epochs as important hyperparameters to tune. Our training procedure is summarized in Figure 9.1,

and in more detail by Algorithm 3.

There are also several optional steps that we found can improve optimization:

- Parameters can be shared between the predictor and policy networks $f(\boldsymbol{x};\theta), \pi(\boldsymbol{x},\phi)$. This does not complicate their joint optimization, and learning a shared representation in the early layers can in some cases help the networks optimize faster.

- Rather than training with a random subset distribution $p(\boldsymbol{S})$, we generate subsets using features selected by the policy $\pi(\boldsymbol{x};\phi)$. This allows the models to focus on subsets likely to be encountered at inference time, and it does not affect the globally optimal policy/predictor: gradients are not propagated between selections, so both eq. 9.5 and this sampling approach treat each feature set as an independent optimization problem, only with different weights (see Appendix G.4).

- We pre-train the predictor $f(\boldsymbol{x};\theta)$ using random subsets before jointly training the policy-predictor pair. This works better than optimizing $\mathcal{L}(\theta,\phi)$ from a random initialization, because a random predictor $f(\boldsymbol{x};\theta)$ provides no signal to $\pi(\boldsymbol{x};\phi)$ about which features are useful.

9.6 Related Work

Prior work has frequently addressed DFS using RL. For example, several works optimize a reward based on the final prediction accuracy (Dulac-Arnold et al., 2011; Shim et al., 2018; Janisch et al., 2019; Li and Oliva, 2021), and Kachuee et al. (2018) use a reward that accounts for prediction uncertainty. RL is a natural approach for sequential decision-making problems, but it can be difficult to optimize in practice: RL requires complex training routines, is slow to converge, and is highly sensitive to its initialization (Henderson et al., 2018). As a result, RL-based DFS does not reliably outperform static feature selection, as shown by Erion et al. (2021) and confirmed in our experiments.

Several other approaches include imitation learning (He et al., 2012, 2016a) and iterative feature scoring methods (Melville et al., 2004; Saar-Tsechansky et al., 2009; Chen et al., 2015b; Early et al., 2016b,a). Imitation learning casts DFS as supervised classification, whereas our training approach bypasses the need for an oracle policy. Most existing feature scoring techniques are greedy methods, like ours, but they use scoring heuristics that are unrelated to maximizing CMI (see Section 9.4). Two feature scoring methods are specifically designed to calculate the CMI, but they suffer from important practical limitations: both Ma et al. (2019) and Chattopadhyay et al. (2022) rely on difficult-to-train generative models, which can lead to inaccurate CMI estimation. Our approach is simpler, faster and more flexible, because the selection logic is contained within a policy network that avoids the need for generative modeling.

Static feature selection is a long-standing problem (Guyon and Elisseeff, 2003; Cai et al., 2018). There are no default approaches for neural networks, but one option is ranking features by local or global importance scores (Breiman, 2001; Shrikumar et al., 2017; Sundararajan et al., 2017; Covert et al., 2020). In addition, several prior works have leveraged continuous relaxations to learn feature selection strategies by gradient descent: for example, several methods perform static feature selection (Chang et al., 2017; Balın et al., 2019; Yamada et al., 2020; Lee et al., 2021; Covert et al., 2023a), and others perform instance-wise feature selection given access to all the features (Chen et al., 2018a; Jethani et al., 2021a). Our work uses a similar continuous relaxation for optimization, but in the DFS context, where our method learns a selection policy rather than a static selection layer.

Finally, several works have examined greedy feature selection algorithms from a theoretical perspective. For example, Das and Kempe (2011) and Elenberg et al. (2018) show that weak submodularity implies near-optimal performance for static feature selection. More relevant to our work, Chen et al. (2015c) find that the related notion of adaptive submodularity (Golovin and Krause, 2011) does not not apply in the DFS setting, but the authors manage to provide performance guarantees under specific distributional assumptions.

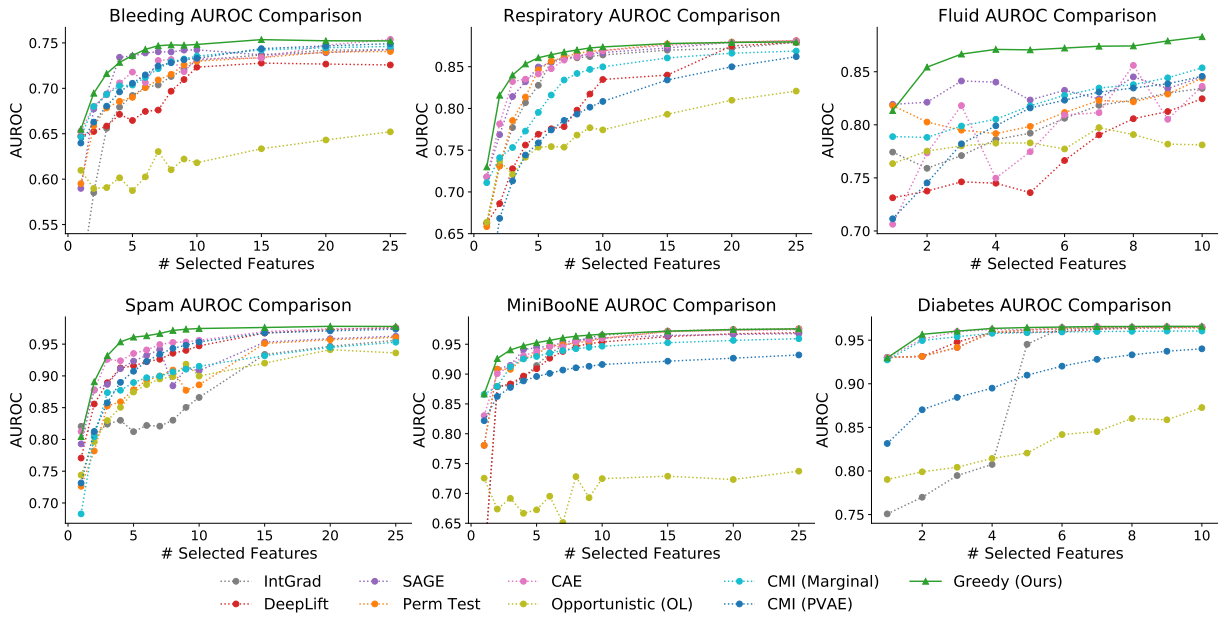


Figure 9.2: Evaluating the greedy approach on six tabular datasets. The results for each method are the average across five runs.

9.7 Experiments

We now demonstrate the use of our greedy approach on several datasets. We first explore tabular datasets of various sizes, including four medical diagnosis tasks, and we then consider two image classification datasets. Several of the tasks are natural candidates for DFS, and the remaining ones serve as useful tasks to test the effectiveness of our approach. Code for reproducing our experiments is available online.[2]

We evaluate our method by comparing to both dynamic and static feature selection methods. As static baselines, we use permutation tests (Breiman, 2001) and SAGE (Covert et al., 2020) to rank features by their importance to the model's accuracy, as well as per-prediction DeepLift (Shrikumar et al., 2017) and IntGrad (Sundararajan et al., 2017) scores aggregated across the dataset. We then use a supervised version of the Concrete Autoencoder

[2] `https://github.com/iancovert/dynamic-selection`

(CAE, Balın et al. 2019), a state-of-the-art static feature selection method. As dynamic baselines, we use two versions of the CMI estimation procedure described in Section 9.4. First, we use the PVAE generative model from Ma et al. (2019) to sample unknown features, and second, we instead sample unknown features from their marginal distribution; in both cases, we use a classifier trained with random feature subsets to make predictions. Finally, we also use the RL-based Opportunistic Learning (OL) approach (Kachuee et al., 2018). Appendix G.3 provides more information about each of the baselines.

9.7.1 Tabular Datasets

We first applied our method to three medical diagnosis tasks derived from an emergency medicine setting. The tasks involve predicting a patient's bleeding risk via a low fibrinogen concentration (Bleeding), whether the patient requires endotracheal intubation for respiratory support (Respiratory), and whether the patient will be responsive to fluid resuscitation (Fluid). See Appendix G.2 for more details about the datasets. In each scenario, gathering all possible inputs at inference time is challenging due to time and resource constraints, thus making DFS a natural solution.

We use fully connected networks for all methods, and we use dropout to reduce overfitting (Srivastava et al., 2014). Figure 9.2 (top) shows the results of applying each method with various feature budgets. The classification accuracy is measured via AUROC, and the greedy method achieves the best results for nearly all feature budgets on all three tasks. Among the baselines, several static methods are sometimes close, but the CMI estimation method is rarely competitive. Additionally, OL provides unstable and weak results. The greedy method's advantage is often largest when selecting a small number of features, and it usually becomes narrower once the accuracy saturates.

Next, we conducted experiments using three publicly available tabular datasets: spam classification (Dua and Graff, 2017), particle identification (MiniBooNE) (Roe et al., 2005) and diabetes diagnosis (Miller, 1973). The diabetes task is a natural application for DFS and was used in prior work (Kachuee et al., 2018). We again tested various numbers of features,

Table 9.1: AUROC averaged across budgets of 1-10 features (with 95% confidence intervals).

		Spam	MiniBooNE	Diabetes	Bleeding	Respiratory	Fluid
Static	IntGrad	82.84 ± 0.68	89.10 ± 0.33	88.91 ± 0.24	66.70 ± 0.27	81.10 ± 0.04	79.94 ± 0.94
	DeepLift	90.16 ± 1.24	88.62 ± 0.30	95.42 ± 0.13	67.75 ± 0.49	76.05 ± 0.35	76.96 ± 0.56
	SAGE	89.70 ± 1.10	92.64 ± 0.03	95.43 ± 0.01	71.34 ± 0.19	82.92 ± 0.26	83.27 ± 0.53
	Perm Test	85.64 ± 3.58	92.19 ± 0.15	95.46 ± 0.02	68.89 ± 1.06	81.56 ± 0.28	81.35 ± 1.04
	CAE	92.28 ± 0.27	92.76 ± 0.41	95.91 ± 0.07	70.69 ± 0.57	83.10 ± 0.45	79.40 ± 0.86
Dynamic	Opportunistic (OL)	85.94 ± 0.20	69.23 ± 0.64	83.07 ± 0.82	60.63 ± 0.55	74.44 ± 0.42	78.13 ± 0.31
	CMI (Marginal)	86.57 ± 1.54	92.21 ± 0.40	95.48 ± 0.05	70.57 ± 0.46	79.62 ± 0.62	81.97 ± 0.93
	CMI (PVAE)	89.01 ± 1.40	88.94 ± 1.25	90.50 ± 5.16	70.17 ± 0.74	74.12 ± 3.50	80.27 ± 1.02
	Greedy (Ours)	**93.91 ± 0.17**	**94.46 ± 0.12**	**96.03 ± 0.02**	**72.64 ± 0.31**	**84.48 ± 0.08**	**86.59 ± 0.25**

and Figure 9.2 (bottom) shows plots of the AUROC for each feature budget. On these tasks, the greedy method is once again most accurate for nearly all numbers of features. Table 9.1 summarizes the results via the mean AUROC across $k = 1, \ldots, 10$ features, further emphasizing the benefits of the greedy method across all six datasets. Appendix G.5 shows larger versions of the AUROC curves (Figure G.1 and Figure G.2), as well as plots demonstrating the variability of selections within each dataset.

The results with these datasets reveal that, perhaps surprisingly, dynamic methods can be outperformed by static methods. Interestingly, this point was not highlighted in prior works where strong static baselines were not tested (Kachuee et al., 2018; Janisch et al., 2019). For example, OL is not competitive on these datasets, and the two versions of the CMI estimation approach are not consistently among the top baselines. Dynamic methods are in principle capable of performing better, so the sub-par results from these methods underscore the difficulty of learning both a selection policy and a prediction function that works for multiple feature sets. In these experiments, our approach is the only dynamic method to do both successfully.

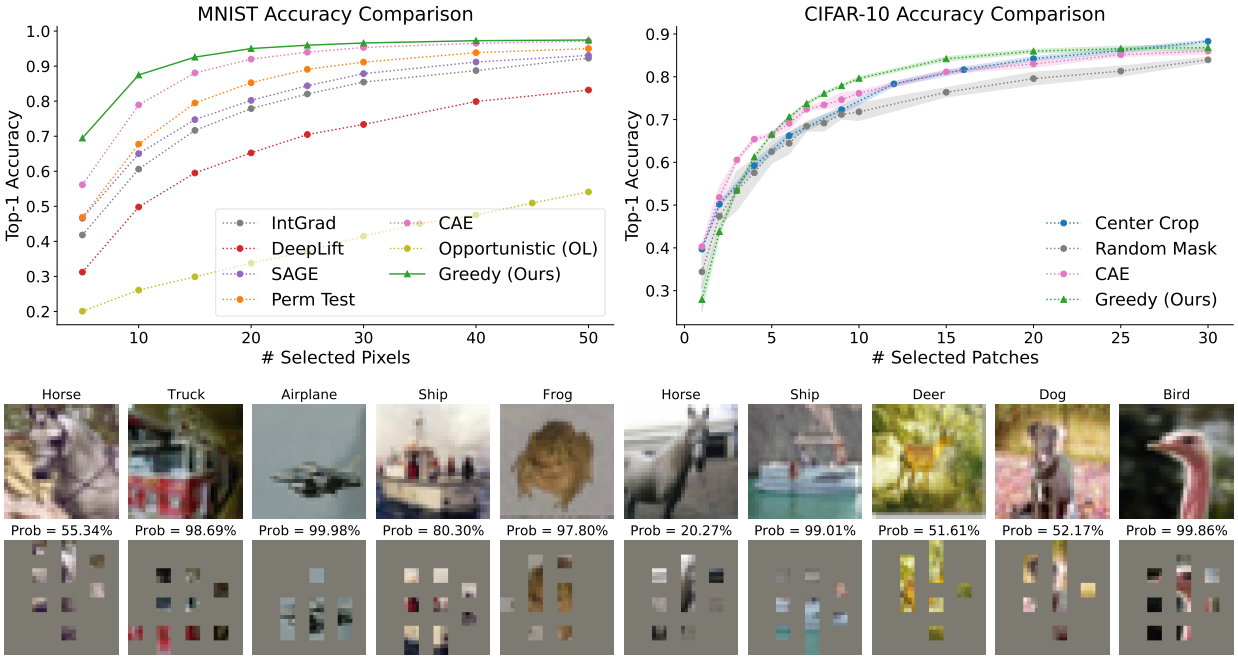


Figure 9.3: Greedy feature selection for image classification. Top left: accuracy comparison on MNIST with results averaged across five runs. Top right: accuracy comparison on CIFAR-10 with 95% confidence intervals. Bottom: example selections and predictions for the greedy method with 10 out of 64 patches for CIFAR-10 images.

9.7.2 Image Classification Datasets

Next, we considered two standard image classification datasets: MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009). Our goal is to begin with a blank image, sequentially reveal multiple pixels or patches, and ultimately make a classification using a small portion of the image. Although this is not an obvious use case for DFS, it represents a challenging problem for our method, and similar tasks were considered in several earlier works (Karayev et al., 2012; Mnih et al., 2014; Early et al., 2016a; Janisch et al., 2019).

For MNIST, we use fully connected architectures for both the policy and predictor, and we treat pixels as individual features; we therefore have $d = 784$. For CIFAR-10, we use a shared ResNet backbone (He et al., 2016b) for the policy and predictor networks, and each network uses its own output head. The 32×32 images are coarsened into $d = 64$ patches of

size 4×4, so the selector head generates logits corresponding to each patch, and the predictor head generates probabilities for each class.

Figure 9.3 shows our method's accuracy for different feature budgets. For MNIST, we use the previous baselines but exclude the CMI estimation method due to its computational cost: it becomes slow when evaluating many candidate features. We observe a large benefit for our method, particularly when making a small number of selections. Our greedy method reaches nearly 90% accuracy with just 10 pixels, which is roughly 10% higher than the best baseline and considerably higher than prior work (Balın et al., 2019; Yamada et al., 2020; Covert et al., 2020). OL yields the worst results, and it also trains slowly due to the large number of states.

For CIFAR-10, we omit several baseline comparisons due to their computational cost. We use the CAE, which is our most competitive static baseline, as well as two simple baselines: center crops and random masks of various sizes. For each method, we plot the mean and 95% confidence intervals determined from five trials. Our greedy approach is slightly less accurate with a very small number of patches, but it reaches significantly higher accuracy when using 6-20 patches. Figure 9.3 (bottom) also shows qualitative examples of our method's predictions after selecting 10 out of 64 patches, and Appendix G.5 shows similar plots with different numbers of patches.

9.8 Conclusion

In this work, we explored a greedy algorithm for dynamic feature selection (DFS) that selects features based on their CMI with the response variable. We proposed an approach to approximate this policy by directly predicting the optimal selection at each step, and we conducted experiments that show our method outperforms a variety of existing feature selection methods, including both dynamic and static baselines. Several problems we did not consider here include incorporating non-uniform features costs, and determining the ideal feature budget on a per-sample basis. We also did not explore architectures that are well-suited to processing partial inputs, particularly for structured data like images.

Chapter 10

ESTIMATING MUTUAL INFORMATION FOR DYNAMIC FEATURE SELECTION

10.1 Chapter Notes

This chapter presents joint work with Soham Gadgil (co-first author) and Su-In Lee that is currently a pre-print. This work addresses the same problem as Chapter 9, and its main contribution is to develop an approach that estimates each feature's conditional mutual information with the response variable (rather than the index with maximum information), which can be used to allow non-uniform feature costs and variable budgets across predictions.

10.2 Introduction

Many machine learning applications rely on high-dimensional datasets with significant data acquisition costs. For example, medical diagnosis can depend on a range of demographic features, lab tests and physical examinations, and each piece of information takes time and money to obtain (Kachuee et al., 2018; Erion et al., 2021; He et al., 2022). To improve interpretability and reduce data acquisition costs, a natural approach is to adaptively query features given the current information, so that each prediction relies on only a small number of features. This approach is referred to as *dynamic feature selection* (DFS),[1] and it is a promising paradigm that has been considered by several works in recent years (Kachuee et al., 2018; Janisch et al., 2019; Chattopadhyay et al., 2022, 2023; Covert et al., 2023b).

Among the existing methods that address this problem, two main approaches have emerged. One idea is to formulate DFS as a Markov decision process (MDP) and use reinforcement learning (RL) (Dulac-Arnold et al., 2011; Mnih et al., 2014; Kachuee et al., 2018;

[1]We use the same terminology as Chapter 9, but the problem has also been referred to by other names.

Janisch et al., 2019). This approach has the capacity to discover the optimal policy, but it faces training difficulties that are common in RL (Henderson et al., 2018). Alternatively, another line of work focuses on greedy approaches, where features are selected based on their conditional mutual information (CMI) with the response variable (Chen et al., 2015c; Ma et al., 2019). While less flexible, the greedy approach is near-optimal under certain assumptions about the data distribution (Chen et al., 2015c) and has been found to work better in practice (Chattopadhyay et al., 2023; Covert et al., 2023b).

Nevertheless, the greedy approach is non-trivial to implement because calculating the CMI requires detailed knowledge of the data distribution (see Section 10.3). Many recent works have explored approximating the CMI using generative models (Ma et al., 2019; Rangrej and Clark, 2021; Chattopadhyay et al., 2022; He et al., 2022), but these methods face a challenging modeling problem (Rangrej and Clark, 2021; Ma et al., 2020; Nazabal et al., 2020) and lead to a slow CMI estimation process (He et al., 2022). Instead, two recent works introduced a simpler approach, which is to directly estimate the feature index with maximum CMI (Chattopadhyay et al., 2023; Covert et al., 2023b). These methods rely on simpler learning objectives, are faster at inference time, and often provide better predictive accuracy. They can be thought of *discriminative* alternatives to earlier generative methods (Ng and Jordan, 2001; Chattopadhyay et al., 2023), and their main downside is that they bypass estimating the CMI, which can be useful for multiple purposes (e.g., determining when to stop selecting new features).

Here, our goal is to advance the greedy DFS approach by combining the best aspects of current methods: building on recent work (Chattopadhyay et al., 2023; Covert et al., 2023b), we aim to estimate the CMI itself in a discriminative fashion. We aim to do so without requiring additional labels, making strong assumptions about the data distribution, or fitting generative models. We find that this is possible by designing a suitable learning objective, which we prove recovers the CMI if our model is trained to optimality (Section 10.5). Based on this, we then explore a range of capabilities enabled by accurately estimating the CMI: these include accounting for non-uniform feature costs, trading off feature cost and

information in multiple ways, and leveraging modern architectures to improve our learning approach.

We find that our proposal offers a promising alternative to available methods: it enables the advantages of generative methods while retaining the simplicity of discriminative methods, and it shows improved performance in our experiments. The contributions presented in this chapter are the following:

1. We develop a learning approach to estimate the CMI in a discriminative fashion. Our method involves training a network to score candidate features based on their predictive utility, and we prove that training with our objective recovers the exact CMI at optimality.

2. We generalize our approach to incorporate prior information beyond the main features. Here, we again prove that our procedure recovers a modified version of the CMI at optimality.

3. Taking inspiration from adaptive submodular optimization, we show how to adapt our CMI-based approach to scenarios with non-uniform feature costs.

4. We analyze the role of variable feature budgets and how they enable an improved cost-accuracy trade-off. We show that a single instantiation of our method can be evaluated with multiple stopping criteria, and that a policy with no per-prediction budget constraints should perform best.

5. We investigate the role of modern architectures in improving performance in DFS. In particular, we find that for image data, our method benefits from using ViTs rather than standard CNNs.

Our experiments demonstrate the effectiveness of our approach across a range of applications, including several tabular and image datasets. We compare our approach to many recent

methods, and we find that our approach provides consistent gains across all the datasets we tested.

10.3 Problem Formulation

As in Chapter 9, we focus on a supervised learning task with input features $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ and response variable $\boldsymbol{y}$. Our goal is to select features given the currently available information, and do so on a per-instance basis to rapidly arrive at accurate predictions. For this, we require a predictor $f(\boldsymbol{x}_S)$ that makes predictions given any set of available features; for example, if $\boldsymbol{y}$ is discrete then its predictions lie in the simplex, or $f(\boldsymbol{x}_S) \in \Delta^{K-1}$ for K classes. We also require a selection policy $\pi(\boldsymbol{x}_S) \in [d]$ that takes a set of features as its input and outputs the next feature index to observe. We next discuss how to design these models, focusing on an approach motivated by information theory.

Previous work has explored several approaches to design a selection policy, including training the policy with RL (Kachuee et al., 2018; Janisch et al., 2019), imitation learning (He et al., 2012, 2016a), and following a greedy policy based on CMI (Ma et al., 2019; Rangrej and Clark, 2021; Chattopadhyay et al., 2023; Covert et al., 2023b). We focus here on the latter approach, where the idealized policy selects the feature with maximum CMI at each step, or where we define $\pi^*(x_S) = \arg\max_i I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$. Note that the CMI is equal to the following KL divergence (Cover and Thomas, 2012):

$$I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S) = D_{\mathrm{KL}} \left(p(\boldsymbol{x}_i, \boldsymbol{y} \mid x_S) \mid\mid p(\boldsymbol{x}_i \mid x_S) p(\boldsymbol{y} \mid x_S) \right). \tag{10.1}$$

This approach therefore identifies the most informative feature at each step; this is consistent with performing greedy uncertainty minimization (Covert et al., 2023b), and the same idea is also the basis of Bayesian experimental design (Lindley, 1956; Bernardo, 1979). The idealized selection policy is accompanied by an idealized predictor, which for classification problems is the Bayes classifier $f^*(x_S) = p(\boldsymbol{y} \mid x_S)$.

It is known that under certain assumptions about the data distribution, greedily selecting

features with $\pi^*(x_S)$ provides performance within a multiplicative factor of the optimal policy (Chen et al., 2015c). However, the CMI policy is difficult to implement because it requires oracle access to the data distribution: computing eq. 10.1 requires both the response and feature distributions $p(\boldsymbol{y} \mid x_S)$ and $p(\boldsymbol{x}_i \mid x_S)$ for all (S, i), which presents a challenging modeling problem. Some works have approximated $I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$ using generative models (Ma et al., 2019; Rangrej and Clark, 2021; Chattopadhyay et al., 2022; He et al., 2022), while others have directly modeled $\pi^*(x_S)$ (Chattopadhyay et al., 2023; Covert et al., 2023b).

When we follow the greedy CMI policy, another question that arises is how many features to select for each prediction. Previous work has focused mainly on the fixed-budget setting (Chen et al., 2015c; Ma et al., 2019; Rangrej and Clark, 2021; Chattopadhyay et al., 2023; Covert et al., 2023b), where we stop given x_S when $|S| = k$ for a specified budget $k < d$. We instead consider variable budgets in this work, where the goal is to achieve high accuracy given a low *average feature cost*. Unlike many works, we also consider non-uniform costs for each feature. As we discuss in Section 10.5, these goals are made easier by our method for estimating the CMI in a discriminative fashion.

10.4 Related Work

The related works here are similar to Chapter 9, but we include this discussion for completeness. One of the earliest works on DFS is Geman and Jedynak (1996), who used CMI as a selection criterion but made simplifying assumptions about the data distribution. Chen et al. (2015c) analyzed the greedy CMI approach from a theoretical perspective and showed conditions under which it achieves near-optimal performance. More recent works have focused on practical implementations: among them, several focused on generative modeling approaches to approximate the CMI (Ma et al., 2019; Rangrej and Clark, 2021; Chattopadhyay et al., 2022; He et al., 2022), and two concurrent works proposed discriminative approaches that directly predict the optimal feature index (Chattopadhyay et al., 2023; Covert et al., 2023b). Our work develops a similar discriminative approach, but in order to estimate the CMI itself rather than the argmax, which provides an alternative to current generative approaches.

Apart from these, many works have addressed DFS as an RL problem (Dulac-Arnold et al., 2011; Shim et al., 2018; Kachuee et al., 2018; Janisch et al., 2019; Li and Oliva, 2021). For example, Janisch et al. (2019) formulate DFS as a MDP where the reward is the 0-1 loss minus the feature cost. RL theoretically has the capacity to discover better policies than a greedy approach, but it has not been found to perform well in practice (Erion et al., 2021; Chattopadhyay et al., 2023; Covert et al., 2023b), seemingly due to training difficulties that are common in RL (Henderson et al., 2018). Other works have instead explored the use of imitation learning, where selections are made by mimicking an oracle policy (He et al., 2012, 2016a).

Finally, static feature selection has been an important subject in statistics and machine learning for decades; see Guyon and Elisseeff (2003); Li et al. (2017a); Cai et al. (2018) for reviews. CMI is also at the basis of some static methods, see Fleuret (2004) for example. Greedy methods have been proven to perform well under certain assumptions about the data distribution (Das and Kempe, 2011; Elenberg et al., 2018), and such methods are popular for models that are inexpensive to fit (e.g., linear regression). Feature selection with neural networks is more challenging, but methods now exist that leverage either group sparse penalties (Feng and Simon, 2017; Tank et al., 2021; Lemhadri et al., 2021) or differentiable gating mechanisms (Chang et al., 2017; Balın et al., 2019; Lindenbaum et al., 2021). Such gating approaches were the basis for recent DFS methods (Chattopadhyay et al., 2023; Covert et al., 2023b), but our work bypasses these techniques with a simpler regression objective.

10.5 Proposed Method

In this section, we introduce our method to dynamically select features by estimating the CMI in a discriminative fashion. We then discuss how to incorporate prior information into the selection process, handle non-uniform feature costs, and enable variable budgets across predictions.

10.5.1 Estimating the Conditional Mutual Information

We parameterize two networks to implement our selection policy. First, we have a predictor network $f(\boldsymbol{x}_S; \theta)$, e.g., a classifier with predictions in Δ^{K-1}. Next, we have a value network $v(\boldsymbol{x}_S; \phi) \in \mathbb{R}^d$ designed to estimate the CMI for each feature, or $v_i(x_S; \phi) \approx I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$. These are implemented with zero-masking to indicate missing features, and we can also pass the mask as a binary vector to indicate missing values. Once they are trained, we make selections according to $\arg\max_i v_i(\boldsymbol{x}_S; \phi)$, and we can make predictions at any time using $f(\boldsymbol{x}_S; \theta)$. The critical question is how to train the networks effectively, given that prior work required generative models to estimate the CMI (Ma et al., 2019; Chattopadhyay et al., 2022).

Our main insight is to train the models jointly but with their own objectives, and to design an objective for the value network that recovers the CMI at optimality. Specifically, we formulate a regression problem whose target is the incremental improvement in the loss when incorporating a single new feature. We train the predictor to make accurate predictions given any feature set, or

$$\min_{\theta} \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \left[\ell \left(f(\boldsymbol{x}_{\boldsymbol{S}}; \theta), y \right) \right], \tag{10.2}$$

and we simultaneously train the value network with the following regression objective,

$$\min_{\phi} \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \mathop{\mathbb{E}}_{p(i)} \left[\left(v_i(\boldsymbol{x}_{\boldsymbol{S}}; \phi) - \Delta(\boldsymbol{x}_{\boldsymbol{S}}, \boldsymbol{x}_i, \boldsymbol{y}) \right)^2 \right], \tag{10.3}$$

where we define the loss improvement as $\Delta(x_S, x_i, y) = \ell(f(x_S; \theta), y) - \ell(f(x_{S \cup i}; \theta), y)$. The regression objective in eq. 10.3 is motivated by the following property, which shows that if we assume an accurate predictor $f(\boldsymbol{x}_S; \theta)$ (i.e., the Bayes classifier), the value network's labels are unbiased estimates of the CMI (proofs are in Appendix H.1).

Lemma 10.1. *When we use the Bayes classifier $p(\boldsymbol{y} \mid \boldsymbol{x}_S)$ as a predictor and ℓ is cross entropy loss, the incremental loss improvement is an unbiased estimator of the CMI for each*

$(x_S, \boldsymbol{x}_i)$ *pair:*

$$\mathop{\mathbb{E}}_{p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)} [\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})] = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S). \tag{10.4}$$

Based on this result and the fact that the optimal predictor does not depend on the selection policy (Covert et al., 2023b), we can make the following claim about jointly training the two models. We assume that both models are infinitely expressive (e.g., very wide networks) so that they can achieve their respective optimizers.

Theorem 10.1. *When ℓ is cross entropy loss, the objectives eq.* 10.2 *and eq.* 10.3 *are jointly optimized by a predictor $f(x_S; \theta^*) = p(\boldsymbol{y} \mid x_S)$ and value network where $v_i(x_S; \phi^*) = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$ for $i \in [d]$.*

This allows us to train the models in an end-to-end fashion using stochastic gradient descent, see Figure 10.1. In Appendix H.1 we prove a similar result for regression problems: that the policy estimates the reduction in conditional variance associated with each candidate feature. Additional analysis in Appendix H.2 shows how suboptimality in the classifier can affect the learned CMI estimates; however, even if the learned estimates $v_i(\boldsymbol{x}_S; \phi)$ are imperfect in practice, we expect good performance because the policy replicates selections that yield large loss improvements during training.

Several other steps are important during training, and these are detailed in Appendix H.3. First, like several prior methods, we pre-train the predictor with random feature sets before beginning joint training (Rangrej and Clark, 2021; Chattopadhyay et al., 2023; Covert et al., 2023b). Next, we generate training samples (x_S, x_i, y) by executing the current policy with a random exploration probability $\epsilon \in (0, 1)$, which can be decayed throughout training. Finally, we sometimes share parameters between the models, particularly when they are large; this helps in our experiments with image data, which use either CNNs or ViTs (Dosovitskiy et al., 2020).

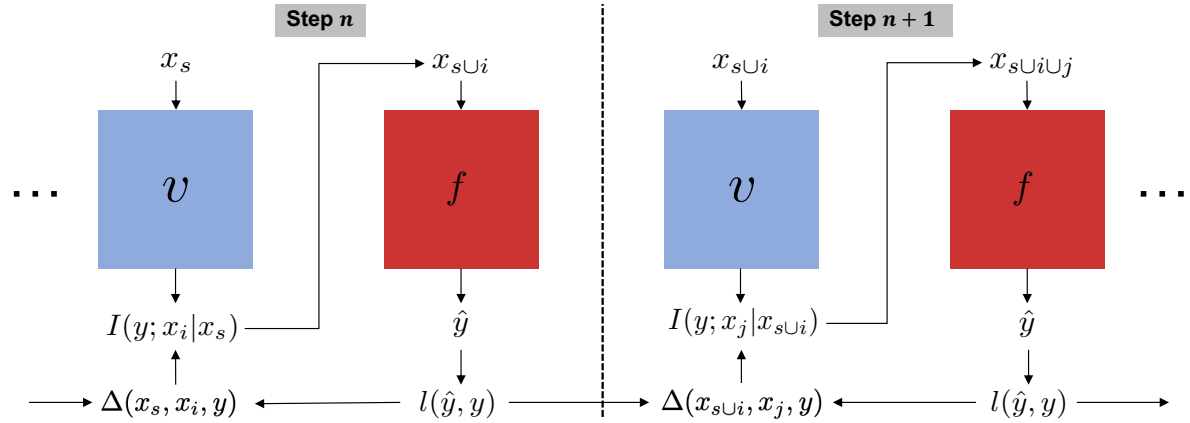


Figure 10.1: Diagram of our training approach. At each selection step n, the value network $v(x_S; \phi)$ predicts the CMI for all features, and a single feature x_i is chosen for the next prediction $f(x_{S\cup i}; \theta)$. The prediction loss is used to update the predictor (see eq. 10.2), and the loss improvement is used to update the value network (see eq. 10.3). The networks are trained jointly with SGD.

10.5.2 Incorporating Prior Information

A further direction is utilizing *prior information* obtained before beginning the selection process. We view such prior information as a separate random variable $\boldsymbol{z}$, and it can be either an exogenous input, a subset of features that are available with no associated cost, or even a noisy or low-resolution version of the real input $\boldsymbol{x}$ (Ranzato, 2014; Ba et al., 2014, 2015). Situations of this form arise in multiple applications, e.g., a patient's demographic features in a medical diagnosis setting.

Given such prior information, our idealized DFS policy should be modified as follows. First, the selections must be based on $I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S, z)$, which captures how informative $\boldsymbol{x}_i$ is given knowledge of both $\boldsymbol{x}_S$ and $\boldsymbol{z}$. Next, the predictions made at any time are given by $p(\boldsymbol{y} \mid x_S, z)$, because $\boldsymbol{z}$ provides information that can improve our predictions. As for our proposed CMI estimation procedure in Section 10.5.1, it is straightforward to modify. The two models must take the prior information $\boldsymbol{z}$ as an additional input, and we can train them

with modified versions of eqs. 10.2 and 10.3,

$$\min_{\theta} \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \left[\ell\left(f(\boldsymbol{x}_{\boldsymbol{S}}, \boldsymbol{z}; \theta), y\right)\right]$$
$$\min_{\phi} \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \mathop{\mathbb{E}}_{p(i)} \left[\left(v_i(\boldsymbol{x}_{\boldsymbol{S}}, \boldsymbol{z}; \phi) - \Delta(\boldsymbol{x}_{\boldsymbol{S}}, \boldsymbol{x}_i, \boldsymbol{z}, \boldsymbol{y})\right)^2\right], \tag{10.5}$$

where $\Delta(x_S, x_i, z, y) = \ell(f(x_S, z, ; \theta), y) - \ell(f(x_{S\cup i}, z; \theta), y)$ is the incremental loss improvement. We then have the following result for jointly training the two models.

Theorem 10.2. *When ℓ is cross entropy loss, the objectives in eq.* 10.5 *are jointly optimized by a predictor $f(x_S, z; \theta^*) = p(\boldsymbol{y} \mid x_S, z)$ and value network where $v_i(x_S, z; \phi^*) = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S, z)$ for all $i \in [d]$.*

The same implementation details discussed in Section 10.5.1 apply here, and as before, we expect strong performance even if the CMI estimates $v_i(\boldsymbol{x}_S, \boldsymbol{z}; \phi)$ do not reach their optimal values in practice.

10.5.3 Allowing a Variable Feature Budget

Given our approach for estimating each feature's CMI, a natural question is how to trade off information with feature acquisition costs. We now consider two challenges related to features costs: (i) how to handle non-uniform costs between features, and (ii) when to stop collecting new features.

Non-uniform costs For the first challenge, consider medical diagnosis as a motivating example. Diagnoses can be informed by heterogeneous data, including demographic variables, questionnaires, physical examinations, and lab tests; each measurement can require a different amount of time or money (Kachuee et al., 2018; Erion et al., 2021), and feature costs must be balanced with the information they provide. For simplicity, we consider that each feature has a cost $c_i > 0$ and that costs are additive across features.

There are multiple ways to trade off cost with information, but we take inspiration

from adaptive submodular optimization, where item costs are accounted for via the ratio between the expected improvement and cost. Here, this suggests that our selections should be $\arg\max_i I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)/c_i$. For adaptive submodular objectives, this criterion guarantees near-optimal performance (Golovin and Krause, 2011); the DFS problem is known *not* to be adaptive submodular (Chen et al., 2015c), which means that we cannot offer performance guarantees, but we find that this approach works well in practice (see Section 10.6).

Variable budgets Next, we consider when to stop acquiring new features. Many previous works focused on the budget-constrained setting, where we adopt a budget k for all predictions (Chen et al., 2015c; Ma et al., 2019; Rangrej and Clark, 2021; Covert et al., 2023b). This can be viewed as a stopping criterion, and it generalizes to non-uniform costs: we can keep collecting new information as long as $\sum_{i \in s} c_i \leq k$. Alternatively, we can adopt a confidence-constrained setup (Chattopadhyay et al., 2023), where selection terminates once the predictions have low uncertainty. For classification problems, a natural approach is to stop collecting features when $H(\boldsymbol{y} \mid x_S) \leq m$.

In general, it is unclear whether we should adopt a budget- or confidence-constrained approach, or whether there is another option that offers a better cost-accuracy trade-off. We resolve this question by considering the optimal performance achievable by *non-greedy policies*, and we present the following insight: that policies with per-prediction constraints are Pareto-dominated by those that achieve their constraints *on average*. The following proposition states this in a simplified form, and we defer the formal version to Appendix H.1 due to the extra setup and notation it requires.

Proposition 10.1. *(Informal) For any feature budget k, the best policy to achieve this budget on average achieves lower loss than the best policy with a per-prediction budget constraint. Similarly, for any confidence level m, the best policy to achieve this confidence on average achieves lower cost than the best policy with a per-prediction confidence constraint.*

Intuitively, when designing a policy to achieve a given average cost or confidence level, it can help to let the policy violate that level for certain predictions. If we only care about the

average cost or accuracy, it does not help to constrain the policy on a per-prediction level. For example, for a patient whose medical condition is inherently uncertain and will not be resolved by any number of tests, it is preferable from a cost-accuracy perspective to stop early rather than run many expensive tests.

Proposition 10.1 suggests that we should avoid adopting budget or confidence constraints and instead seek the optimal unconstrained policy. We do not have access to this optimal policy, because we assume that we only have CMI estimates (Section 10.5.1), so we opt for a simple alternative that avoids per-prediction constraints: we adopt a penalty parameter $\lambda > 0$, we make selections at each step according to $I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)/c_i$, and we terminate the algorithm when $\arg\max_i I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)/c_i < \lambda$.

Following this approach, we see that a single instantiation of our model can be run with three different stopping criteria: we can use the budget k, the confidence m, or the penalty λ. In contrast, prior methods that penalize feature costs required training separately with each λ value (Janisch et al., 2019). Our claim in Proposition 10.1 does not apply to our penalized policy with the λ criterion, which is not guaranteed to be optimal, but we find that it leads to improved performance across most datasets.

10.6 Experiments

We now present results from applying our method to several datasets. We refer to our approach as *DIME*[2] (**di**scriminative **m**utual information **e**stimation) and we explore two data modalities, image and tabular, to evaluate its performance. Our tabular datasets include two medical diagnosis tasks, which represent natural and valuable use cases for DFS; we also use MNIST, which was considered in many prior works (Ma et al., 2019; Chattopadhyay et al., 2022, 2023; Covert et al., 2023b). As for our image datasets, we include these because they are studied in several earlier works (Karayev et al., 2012; Mnih and Gregor, 2014; Janisch et al., 2019; Rangrej and Clark, 2021) and represent challenging problems for our

[2] `https://github.com/suinleelab/DIME`

method. We show results for our method when evaluated with either the budget-constrained or penalized stopping criteria, and we defer comparison with the confidence-constrained approach to Appendix H.7. We note again that DIME only needs be trained once to be tested with all three stopping criteria.

In terms of baselines, we compare DIME to both static and dynamic feature selection methods. As a static baseline, we compare to a supervised version of the Concrete Autoencoder (CAE) (Balın et al., 2019), a state-of-the-art static method that outperformed several dynamic methods in recent work (Covert et al., 2023b). As dynamic baselines, we consider multiple approaches: first, we compare to the recent discriminative methods that directly predict the CMI's argmax (Chattopadhyay et al., 2023; Covert et al., 2023b), which we refer to as Argmax Direct. Next, as a generative approach, we consider EDDI (Ma et al., 2019), which uses a partial variational autoencoder (PVAE) to sample unknown features. Finally, because EDDI does not scale as well beyond tabular datasets, we compare to probabilistic hard attention (Hard Attention) (Rangrej and Clark, 2021), a method that adapts EDDI to work with image data. We omit comparisons to several RL approaches (Mnih and Gregor, 2014; Kachuee et al., 2018; Janisch et al., 2019) because these are harder to train have not proved to be competitive with our other baselines (Rangrej and Clark, 2021; Chattopadhyay et al., 2023; Covert et al., 2023b). Appendix H.6 provides more information about the baseline methods.

10.6.1 Tabular Datasets

We first apply our method to three tabular datasets, two of which are medical diagnosis applications. The first task involves predicting whether a patient requires endotracheal intubation for respiratory support (Intubation) in an emergency medicine setting ($d = 112$) (Covert et al., 2023b). The second one uses cognitive, demographic, and medical history data from the Religious Orders Study and Rush Memory and Aging Project (ROSMAP) (Bennett et al., 2012a,b), two longitudinal aging cohort studies, to predict imminent dementia onset (i.e., a positive diagnosis within the next three years) ($d = 46$). In both scenarios, it is

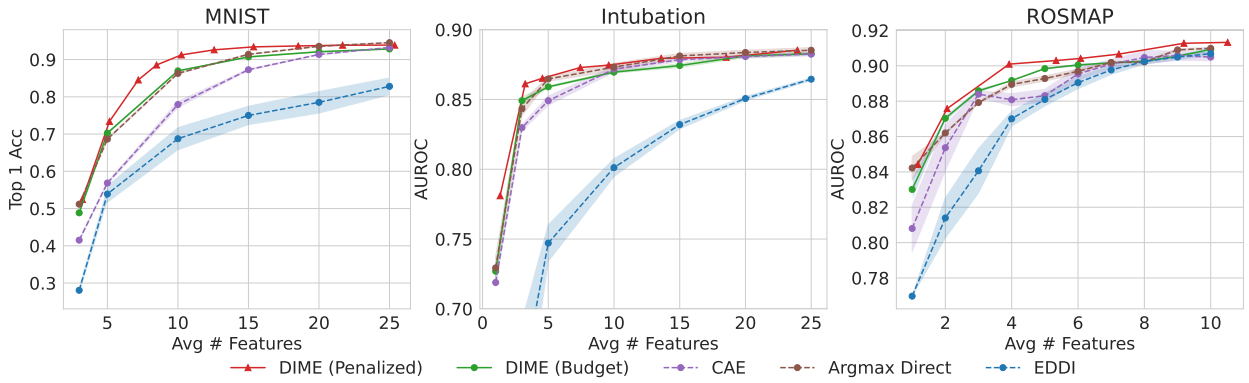


Figure 10.2: Evaluation with tabular datasets for varying feature acquisition budgets. Results are averaged across 5 trials, and shaded regions indicate the standard error for each method.

difficult to acquire the complete feature set due to time constraints, thereby showcasing the utility of dynamic feature selection. The third is the standard MNIST digit classification dataset (LeCun et al., 1998), which we formulate as a tabular problem by treating each pixel as an individual feature ($d = 784$). Across all methods, we use fully connected networks with dropout to reduce overfitting (Srivastava et al., 2014). Classification performance is measured using AUROC for the medical tasks and top-1 accuracy for MNIST. Appendix H.4 provides more details about the datasets, and Appendix H.5 provides more information about the models.

Uniform feature costs We first consider the scenario with equal costs for all features. Figure 10.2 shows the results of applying each method for varying numbers of features. For DIME, we show results when it uses the exact number of features (the budget-constrained approach) and when it uses each number of features *on average* (the penalized approach). DIME with the penalized stopping criterion achieves the best results for nearly all budgets across all three tasks. It performs the best on MNIST, where we are able to achieve above 90% accuracy with only ∼10/784 features (1.27%). Among the baselines, Argmax Direct is the strongest dynamic method: its performance is nearly identical to DIME with a budget

constraint, reflecting the strong similarity between the methods. CAE is a competitive static method, and EDDI usually does not perform well, except for ROSMAP where it approaches the other methods as more features are selected. DIME generally shows the greatest advantage for moderate numbers of features, and the gap reduces as the performance saturates.

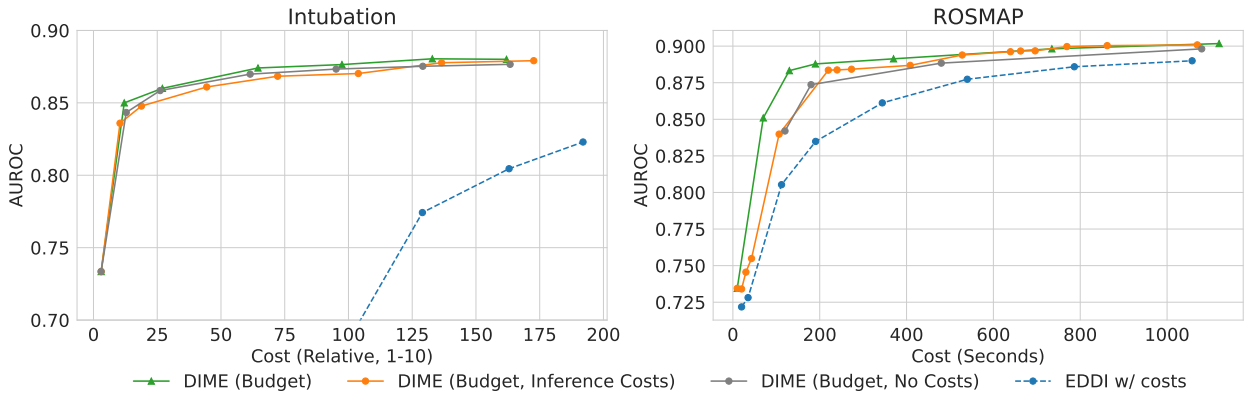


Figure 10.3: Evaluation with non-uniform feature costs for medical diagnosis tasks. Costs are relative for Intubation and expressed in seconds for ROSMAP. The results show the classification performance for varying levels of average feature acquisition cost.

Non-uniform costs Our CMI estimation approach lets us incorporate non-uniform feature costs into DIME, and we demonstrate this using the Intubation and ROSMAP datasets. For ROSMAP, we use costs expressed as the time required to acquire each feature, and for Intubation we use relative costs estimated by a board-certified physician (Appendix H.4). For comparisons, EDDI is the only baseline that can be adapted to use non-uniform feature costs as described in Section 10.5. We also compare to two ablations of our approach: (i) using uniform costs during training but the true costs during inference (Budget, Inference Costs), which tests DIME's robustness to changing feature costs after training; and (ii) using uniform costs during both training and inference (Budget, No Costs), which simply demonstrates the importance of using correct costs. All methods are compared here with the budget-constrained stopping criterion.

Figure 10.3 shows the results when using non-uniform feature costs. DIME outperforms EDDI by a substantial margin, echoing the earlier results and reflecting the improved CMI estimation with our discriminative approach. Comparing to the variations of DIME, using the true non-uniform costs during both training and inference outperforms both variations, showing that considering costs when making selections is important. The version that uses costs only during inference slightly outperforms ignoring costs on ROSMAP, indicating a degree of robustness to changing feature costs between training and inference, but both versions give similar results on Intubation.

10.6.2 Image Datasets

Next, we applied our method to three image classification datasets. The first two are subsets of the standard ImageNet dataset (Deng et al., 2009), one with 10 classes (ImageNette Howard and Gugger 2020) and the other with 100 classes (ImageNet-100 Ambityga). The third is a histopathology classification dataset (MHIST Wei et al. 2021), comprising hematoxylin and eosin (H&E)-stained fixed-size images of colorectal polyps, obtained by extracting diagnostically-relevant image tiles from multiple whole-slide images (WSIs). The task is to predict the histological pattern of each image as either a benign (hyperplastic polyp) or precancerous (sessile serrated adenoma) lesion. WSIs have extremely high resolution and are infeasible to be used directly in any classification task, making them a potential use case for DIME to identify patches most informative to the prediction. The images in all three datasets are 224×224, and we view them as $d = 196$ patches of size 16×16. The value network predicts the CMI for each patch and the predictor generates class predictions. We explore different architectures for the value and predictor networks, namely ResNets (He et al., 2016b) and Vision Transformers (ViTs) (Dosovitskiy et al., 2020). In both cases, we use a shared backbone, with each component having its own output head. Classification performance is measured using top-1 accuracy for both ImageNet subsets, and AUROC for the histopathology task.

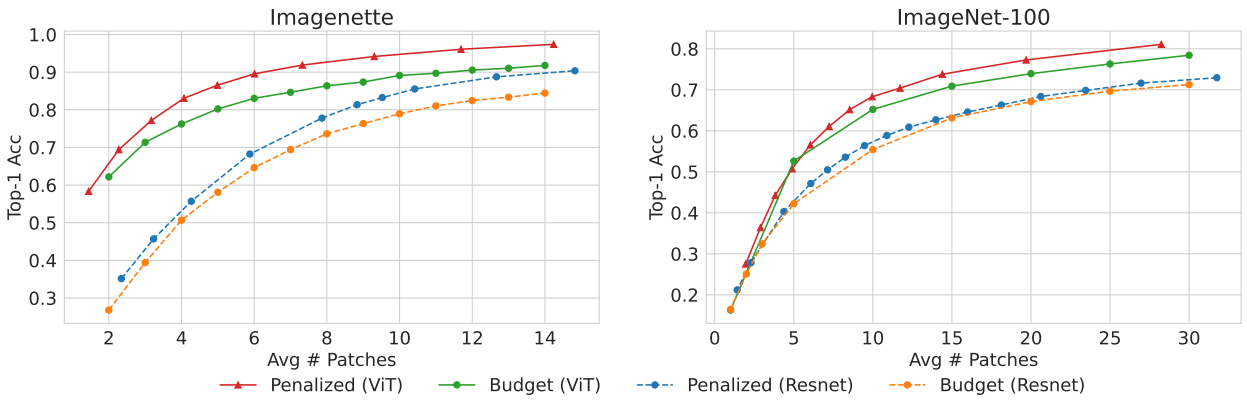


Figure 10.4: Evaluation of DIME on images with different vision architectures.

Exploring modern architectures Figure 10.4 compares the performance of ResNet and ViT architectures for the ImageNette and ImageNet-100 datasets. We use DIME to conduct this analysis, because its discriminative approach lets us seamlessly plug in any network architecture. Across both datasets, DIME's penalized version outperforms the budget-constrained version, and we find that ViTs significantly outperform ResNets. This can be attributed to the ViT's use of self-attention: this architecture is better suited to handle partial information because it aggregates information from patches anywhere in the image, a property that has made ViTs useful in other applications with partial inputs (Naseer et al., 2021; Jain et al., 2021; Salman et al., 2022). Given their superior performance, we use ViTs as backbones in subsequent experiments where possible. This includes DIME, CAE and Argmax Direct, but Hard Attention lacks this flexibility because it relies on a recurrent module to process subsets of image regions.

Figure 10.5 compares DIME to the baselines across multiple feature budgets. The penalized version outperforms the baselines across all datasets for almost all feature budgets, with the largest gains observed for ImageNette. Notably, we are able to achieve nearly 97% accuracy on ImageNette with only ∼15/196 patches (7.7%). The Argmax Direct baseline is competitive with DIME's budget-constrained version, as expected, but the Hard Attention baseline shows a large drop in performance.

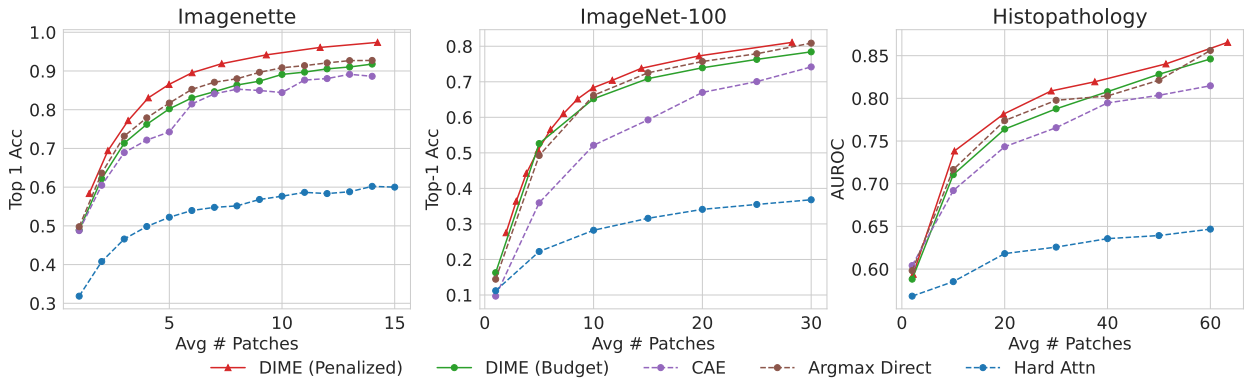


Figure 10.5: Evaluation with image datasets for varying numbers of average patches selected.

Incorporating prior information Next, we explored the possibility of incorporating prior information into DIME's selection process for the histopathology dataset. To simulate informing our selections with a less exact but easily acquirable version of the tissue, we use the Canny edge image (Canny, 1986) as a sketch, which can help generate more valuable selections than a blank image. We use separate ViT backbones for the original and edge images, and we concatenate the resulting embeddings before estimating the CMI or making class predictions (see Appendix H.5 for details). Figure 10.6 shows example images, along with the results obtained with DIME for various feature budgets. The results show that the prior information is incorporated successfully, leading to improved performance for both the penalized and budget-constrained versions. To verify that the accuracy increase is not due solely to predictive signal provided by the edge image, we conduct an ablation where the sketch is integrated into the predictor only for a pre-trained and otherwise frozen version of DIME. This middle ground improves upon no prior information, but it generally performs well below the version that uses prior information both when making selections and predictions.

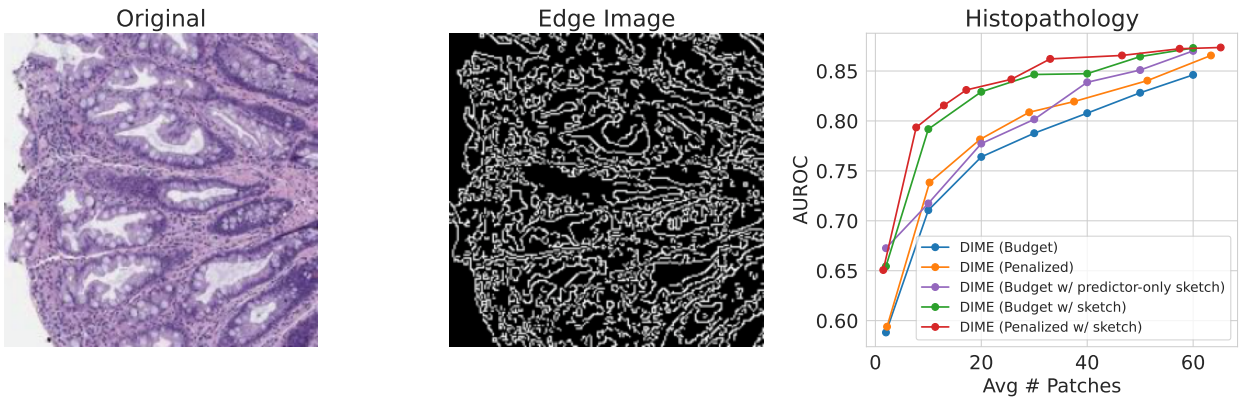


Figure 10.6: Evaluation of DIME with prior information for histopathology classification. Left: Original MHIST image. Center: Canny edge image. Right: Results for varying number of average patches.

10.7 Conclusion

This chapter presents DIME, a new approach for dynamic feature selection enabled by estimating the CMI in a discriminative fashion. Our approach involves learned value and predictor networks, trained jointly in an end-to-end fashion with a straightforward regression objective. From a theoretical perspective, we prove that our training approach recovers the exact CMI at optimality. Empirically, DIME is able to accurately estimate the CMI and enables an improved cost-accuracy trade-off compared to existing methods: it offers a significant improvement over existing generative methods, thereby reducing the need for such models in this setting, and it exceeds the discriminative methods it builds upon (Chattopadhyay et al., 2023; Covert et al., 2023b) by taking advantage of variable feature budgets. Our evaluation considers a range of tabular and image datasets and demonstrates the potential to implement several additions to the classic greedy CMI selection policy: these include allowing non-uniform features costs, variable budgets, and incorporating prior information. The results also show that DIME is robust to higher image resolutions, scales to more classes, and benefits from modern architectures. Future work may focus on promising applications like MRIs and region-of-interest selection within WSIs, using DIME to initialize RL methods,

and otherwise accelerating or improving DIME's training.

Chapter 11

DISCUSSION

In this thesis, we explored a range of approaches to provide transparency for modern machine learning models, and particularly different types of deep neural networks. Chapters 3 and 4 considered a broad class of model explanation methods, which we refer to as *removal-based explanations*; we analyzed the implementation choices that distinguish these methods, as well as their shared theoretical foundations in information theory and game theory. Next, we focused on Shapley values, a theoretically compelling attribution method that is computationally costly in practice: Chapter 5 developed an improved statistical estimator based on linear regression, and Chapters 6 and 7 presented the first methods for amortized Shapley value estimation via a learned explainer model. Finally, we shifted our focus to feature selection via deep learning, which offers the benefits of simplified input dependencies and reduced data acquisition costs: Chapter 8 developed a method to select genes for spatial transcriptomics studies, and Chapters 9 and 10 presented new methods for dynamic feature selection based on the conditional mutual information of unobserved features with the response variable.

Within the problems addressed by these projects, there is still room for future work. On the subject of Shapley value estimation, it remains to be seen whether there are more efficient or simpler ways to train explainer models. Under our proposed approach, training the explainer model is roughly as computationally costly as training the original model, even when we do so via fine-tuning; there may also be a performance gap relative to using ground truth labels (Chuang et al., 2023). Furthermore, it is possible that improved non-model-based approaches will be developed, as the Shapley value can be viewed from many distinct mathematical perspectives (Bian et al., 2021; Chen et al., 2022). For dynamic

feature selection, one obvious future direction is to explore promising medical applications, such as efficient magnetic resonance imaging (MRI) via subsampled measurements. On the methodological side, it is perhaps surprising how poorly reinforcement learning methods performed in our experiments (Chapter 9), but we speculate that greedy approaches could play a role in improving these by providing a strong initialization.

We also briefly address related topics that were not considered at length in this work. The question of how to evaluate model explanations lacks a universally accepted solution, and we largely relied on existing techniques in the literature. Following several works that emphasize the importance of testing fidelity to the model rather than fidelity to a human decision-making process, we focused on metrics like insertion and deletion (Petsiuk et al., 2018), ROAR (Hooker et al., 2019), and sensitivity-n (Ancona et al., 2018). However, Chapter 3 briefly discusses how these metrics are implicitly tied to removal-based methods, and Chapter 7 discusses an information leakage issue with ROAR, which was also alluded to by Jethani et al. (2021a). We leave to future work a more complete resolution to the topic of metrics for model explanations.

Another related topic is the proper downstream usage of model explanations. This thesis was mainly focused on methodology, so we do not provide new perspectives on this topic. However, we point readers to other works on model debugging (DeGrave et al., 2021), biomedical knowledge discovery (Janizek et al., 2021; Iizuka et al., 2019; Novakovsky et al., 2022), the potential for improving human-AI teamwork (Lundberg et al., 2018; Bansal et al., 2021; Fok and Weld, 2023), and the interpretability benefits of feature selection (Guyon and Elisseeff, 2003; Chattopadhyay et al., 2023). We expect that questions around how to best leverage model explanations may be better addressed by human factors research rather than research into new methodologies.

It is worth emphasizing that explainable machine learning (XML) has grown into a proper subfield, which now considers many questions beyond feature attribution and feature selection. Important problems that we did not touch on here include understanding important concepts within a model, and identifying valuable training data examples. The former is

important because a model's raw input features may not provide the best perspective to understand a model's dependencies (Koh et al., 2020; DeGrave et al., 2023). The latter is becoming increasingly important, because we now rely on noisy web-scale datasets for tasks like contrastive learning (Radford et al., 2019) and language modeling (Brown et al., 2020).

Finally, we briefly consider the role of XML in the era of large language models (LLMs). When it comes to explaining how a decision or prediction was made, LLMs like ChatGPT and GPT-4 (OpenAI, 2023) offer a more compelling interface than current XML methods: users can interact with the system, ask any question in natural language, and ask follow-up questions when necessary. Furthermore, such transparency is effectively built into the models, because LLMs exhibit their best performance when asked to explain their reasoning via a series of intermediate steps (known as chain-of-thought prompting) Wei et al. (2022). While conversational interfaces may represent the path forward, a concerning issue is the lack of grounding in such explanations: just as LLMs can hallucinate plausible-sounding facts, they can provide linguistically coherent explanations that are divorced from how a decision was actually made (Turpin et al., 2023). The grounding of current XML methods in the model's prediction mechanism is in this respect an important benefit, and it will be interesting to see whether any lessons from these methods influence the next generation of LLM-based explanation tools.

BIBLIOGRAPHY

National health and nutrition examination survey, 2018. URL `https://www.cdc.gov/nchs/nhanes`.

Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *arXiv preprint arXiv:1903.10464*, 2019.

Kjersti Aas, Thomas Nagler, Martin Jullum, and Anders Løland. Explaining predictive models using Shapley values and non-parametric vine copulas. *arXiv preprint arXiv:2102.06416*, 2021.

Khaled Abdeljawad, Krishna C Vemulapalli, Charles J Kahi, Oscar W Cummings, Dale C Snover, and Douglas K Rex. Sessile serrated polyp prevalence determined by a colonoscopist with a high lesion detection rate and an experienced pathologist. *Gastrointestinal Endoscopy*, 81(3):517–524, 2015.

Abubakar Abid, Muhammad Fatih Balin, and James Zou. Concrete autoencoders for differentiable feature selection and reconstruction. *arXiv preprint arXiv:1901.09346*, 2019.

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

Brian Aevermann, Yun Zhang, Mark Novotny, Trygve Bakken, Jeremy Miller, Rebecca Hodge, Boudewijn Lelieveldt, Ed Lein, and Richard H Scheuermann. NS-Forest: A machine learning method for the objective identification of minimum marker gene combinations for cell type determination from single cell RNA sequencing. *bioRxiv*, 2020.

Chirag Agarwal and Anh Nguyen. Explaining image classifiers by removing input features using generative models. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

Chirag Agarwal, Marinka Zitnik, and Himabindu Lakkaraju. Probing GNN explainers: A rigorous theoretical and empirical analysis of gnn explanation methods. In *International Conference on Artificial Intelligence and Statistics*, pages 8969–8996. PMLR, 2022.

Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. In *International Conference on Machine Learning*, pages 110–119. PMLR, 2021.

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

Ambityga. Imagenet100. `https://www.kaggle.com/datasets/ambityga/imagenet100`.

Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature Methods*, pages 1–7, 2019.

Brandon Amos. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.

Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for Shapley value approximation. In *International Conference on Machine Learning*, pages 272–281. PMLR, 2019.

Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2020.

Alexandre Araujo, Aaron Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu. A unified algebraic perspective on Lipschitz neural networks. *arXiv preprint arXiv:2303.03169*, 2023.

Cédric Arisdakessian, Olivier Poirion, Breck Yunits, Xun Zhu, and Lana X Garmire. DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data. *Genome Biology*, 20(1):1–14, 2019.

Robert JJ Aumann. Economic applications of the Shapley value. In *Game-theoretic Methods in General Equilibrium Analysis*, pages 121–133. Springer, 1994.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

Jimmy Ba, Russ R Salakhutdinov, Roger B Grosse, and Brendan J Frey. Learning wake-sleep recurrent attention models. *Advances in Neural Information Processing Systems*, 28, 2015.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One*, 10(7):e0130140, 2015.

Trygve E Bakken, Nikolas L Jorstad, Qiwen Hu, Blue B Lake, Wei Tian, Brian E Kalmbach, Megan Crow, Rebecca D Hodge, Fenna M Krienen, Staci A Sorensen, et al. Comparative cellular analysis of motor cortex in human, marmoset and mouse. *Nature*, 598(7879): 111–119, 2021.

Muhammed Fatih Balın, Abubakar Abid, and James Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International Conference on Machine Learning*, pages 444–453. PMLR, 2019.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the whole exceed its parts? the effect of AI explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2021.

Naman Bansal, Chirag Agarwal, and Anh Nguyen. SAM: The sensitivity of attribution methods to hyperparameters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8673–8683, 2020.

John F Banzhaf. Weighted voting doesn't work: a mathematical analysis. *Rutgers Law Review*, 19:317, 1964.

Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

Nicasia Beebe-Wang, Alex Okeson, Tim Althoff, and Su-In Lee. Efficient and explainable risk assessments for imminent dementia in an aging cohort study. *IEEE Journal of Biomedical and Health Informatics*, 25(7):2409–2420, 2021.

Mohamed Belghazi, Maxime Oquab, and David Lopez-Paz. Learning about an exponential amount of conditional distributions. In *Advances in Neural Information Processing Systems*, pages 13703–13714, 2019.

David Bennett, Julie Schneider, Zoe Arvanitakis, and Robert Wilson. Overview and findings from the religious orders study. *Current Alzheimer Research*, 9(6):628–645, 2012a.

David Bennett, Julie Schneider, Aron Buchman, Lisa Barnes, Patricia Boyle, and Robert Wilson. Overview and findings from the rush memory and aging project. *Current Alzheimer Research*, 9(6):646–663, 2012b.

Ashton C Berger, Anil Korkut, Rupa S Kanchi, Apurva M Hegde, Walter Lenoir, Wenbin Liu, Yuexin Liu, Huihui Fan, Hui Shen, Visweswaran Ravikumar, et al. A comprehensive pan-cancer molecular study of gynecologic and breast cancers. *Cancer Cell*, 33(4):690–705, 2018.

José M Bernardo. Expected information as expected utility. *The Annals of Statistics*, pages 686–690, 1979.

Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Better plain ViT baselines for ImageNet-1k. *arXiv preprint arXiv:2205.01580*, 2022.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020.

Umang Bhatt, Adrian Weller, and José MF Moura. Evaluating and aggregating feature-based model explanations. In *International Conference on International Joint Conferences on Artificial Intelligence*, 2021.

Yatao Bian, Yu Rong, Tingyang Xu, Jiaxiang Wu, Andreas Krause, and Junzhou Huang. Energy-based learning for cooperative games, with applications to valuation problems in machine learning. *arXiv preprint arXiv:2106.02938*, 2021.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001.

Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on ImageNet. *arXiv preprint arXiv:1904.00760*, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020.

Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2956–2964, 2015.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

Chun-Hao Chang, Ladislav Rampasek, and Anna Goldenberg. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017.

Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. *arXiv preprint arXiv:1807.08024*, 2018.

A Charnes and Daniel Granot. Coalitional and chance-constrained solutions to n-person games. i: The prior satisficing nucleolus. *SIAM Journal on Applied Mathematics*, 31(2): 358–367, 1976.

A Charnes, B Golany, M Keane, and J Rousseau. Extremal principle solutions of games in characteristic function form: core, Chebychev and Shapley value generalizations. In *Econometrics of Planning and Efficiency*, pages 123–133. Springer, 1988.

Abraham Charnes and Daniel Granot. Prior solutions: Extensions of convex nucleus solutions to chance-constrained games. Technical report, Texas University at Austin Center for Cybernetic Studies, 1973.

Aditya Chattopadhyay, Stewart Slocum, Benjamin D Haeffele, Rene Vidal, and Donald Geman. Interpretable by design: Learning predictors by composing interpretable queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Aditya Chattopadhyay, Kwan Ho Ryan Chan, Benjamin D Haeffele, Donald Geman, and René Vidal. Variational information pursuit for interpretable predictions. *arXiv preprint arXiv:2302.02876*, 2023.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 782–791, 2021.

Hugh Chen, Joseph D Janizek, Scott Lundberg, and Su-In Lee. True to the model or true to the data? *arXiv preprint arXiv:2006.16234*, 2020.

Hugh Chen, Ian C Covert, Scott M Lundberg, and Su-In Lee. Algorithms to estimate Shapley value feature attributions. *arXiv preprint arXiv:2207.07605*, 2022.

Jianbo Chen and Michael Jordan. LS-Tree: Model interpretation when the data are linguistic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3454–3461, 2020.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *arXiv preprint arXiv:1802.07814*, 2018a.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. L-Shapley and C-Shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610*, 2018b.

Kok Hao Chen, Alistair N Boettiger, Jeffrey R Moffitt, Siyuan Wang, and Xiaowei Zhuang. Spatially resolved, highly multiplexed RNA profiling in single cells. *Science*, 348(6233), 2015a.

Suming Chen, Arthur Choi, and Adnan Darwiche. Value of information based on decision robustness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015b.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

Yuxin Chen, S Hamed Hassani, Amin Karbasi, and Andreas Krause. Sequential information maximization: When is greedy near-optimal? In *Conference on Learning Theory*, pages 338–363. PMLR, 2015c.

Yu-Neng Chuang, Guanchu Wang, Fan Yang, Quan Zhou, Pushkar Tripathi, Xuanting Cai, and Xia Hu. CoRTX: Contrastive framework for real-time explanation. *arXiv preprint arXiv:2303.02794*, 2023.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does BERT look at? an analysis of BERT's attention. *arXiv preprint arXiv:1906.04341*, 2019.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Oscar Clivio, Romain Lopez, Jeffrey Regier, Adam Gayoso, Michael I Jordan, and Nir Yosef. Detecting zero-inflated genes in single-cell transcriptomics data. *bioRxiv*, page 794875, 2019.

Jennie L Close, Brian R Long, and Hongkui Zeng. Spatially resolved transcriptomics in neuroscience. *Nature Methods*, 18(1):23–25, 2021.

Simone Codeluppi, Lars E Borm, Amit Zeisel, Gioele La Manno, Josina A van Lunteren, Camilla I Svensson, and Sten Linnarsson. Spatial organization of the somatosensory cortex revealed by osmFISH. *Nature Methods*, 15(11):932–935, 2018.

Cameron Condylis, Abed Ghanbari, Nikita Manjrekar, Karina Bistrong, Shenqin Yao, Zizhen Yao, Thuc Nghi Nguyen, Hongkui Zeng, Bosiljka Tasic, and Jerry L. Chen. Dense functional and molecular readout of a circuit hub in sensory cortex. *Science*, 375(6576): eabl5981, 2022. doi: 10.1126/science.abl5981. URL `https://www.science.org/doi/abs/10.1126/science.abl5981`.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 3457–3465. PMLR, 2021.

Ian Covert, Scott M Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33:17212–17223, 2020.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Ian Covert, Chanwoo Kim, and Su-In Lee. Learning to estimate Shapley values with vision transformers. *arXiv preprint arXiv:2206.05282*, 2022.

Ian Covert, Rohan Gala, Tim Wang, Karel Svoboda, Uygar Sümbül, and Su-In Lee. Predictive and robust gene selection for spatial transcriptomics. *Nature Communications*, 14(1): 2091, 2023a.

Ian Covert, Wei Qiu, Mingyu Lu, Nayoon Kim, Nathan White, and Su-In Lee. Learning to maximize mutual information for dynamic feature selection. *arXiv preprint arXiv:2301.00557*, 2023b.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6967–6976, 2017.

Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 598–617. IEEE, 2016.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887):70–74, 2021.

Alex J DeGrave, Joseph D Janizek, and Su-In Lee. AI for radiographic COVID-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 3(7):610–619, 2021.

Alex J DeGrave, Zhuo Ran Cai, Joseph D Janizek, Roxana Daneshjou, and Su-In Lee. Dissection of medical AI reasoning processes via physician and generative-AI collaboration. *medRxiv*, pages 2023–05, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. 2020.

Guoli Ding, Robert F Lax, Jianhua Chen, and Peter P Chen. Formulas for approximating pseudo-boolean random variables. *Discrete Applied Mathematics*, 156(10):1581–1597, 2008.

Guoli Ding, Robert F Lax, Jianhua Chen, Peter P Chen, and Brian D Marx. Transforms of pseudo-boolean random variables. *Discrete Applied Mathematics*, 158(1):13–24, 2010.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 32, 2019.

Ann-Kathrin Dombrowski, Christopher J Anders, Klaus-Robert Müller, and Pan Kessel. Towards robust explanations for deep neural networks. *Pattern Recognition*, 121:108194, 2022.

Jiayun Dong and Cynthia Rudin. Exploring the cloud of variable importance for the set of all good models. *Nature Machine Intelligence*, 2(12):810–824, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Laura Douglas, Iliyan Zarov, Konstantinos Gourgoulias, Chris Lucas, Chris Hart, Adam Baker, Maneesh Sahani, Yura Perov, and Saurabh Johri. A universal marginalizer for amortized inference in generative models. *arXiv preprint arXiv:1711.00695*, 2017.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Pradeep Dubey and Lloyd S Shapley. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari. Datum-wise classification: a sequential approach to sparsity. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 375–390. Springer, 2011.

Bianca Dumitrascu, Soledad Villar, Dustin G Mixon, and Barbara E Engelhardt. Optimal marker gene selection for cell type discrimination in single cell analyses. *Nature Communications*, 12(1):1–8, 2021.

Kirstin Early, Stephen E Fienberg, and Jennifer Mankoff. Test time feature ordering with FOCUS: Interactive predictions with minimal user burden. In *Proceedings of the 2016*

ACM International Joint Conference on Pervasive and Ubiquitous Computing, pages 992–1003, 2016a.

Kirstin Early, Jennifer Mankoff, and Stephen E Fienberg. Dynamic question ordering in online surveys. *arXiv preprint arXiv:1607.04209*, 2016b.

Ethan R Elenberg, Rajiv Khanna, Alexandros G Dimakis, and Sahand Negahban. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568, 2018.

Chee-Huat Linus Eng, Michael Lawson, Qian Zhu, Ruben Dries, Noushin Koulena, Yodai Takei, Jina Yun, Christopher Cronin, Christoph Karp, Guo-Cheng Yuan, et al. Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+. *Nature*, 568 (7751):235–239, 2019.

James F Epperson. *An introduction to numerical methods and analysis*. John Wiley & Sons, 2021.

Kai Epstude and Neal J Roese. The functional theory of counterfactual thinking. *Personality and social psychology review*, 12(2):168–192, 2008.

Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell RNA-seq denoising using a deep count autoencoder. *Nature Communications*, 10(1):1–14, 2019.

Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

Gabriel Erion, Joseph D Janizek, Carly Hudelson, Richard B Utarnachitt, Andrew M McCoy, Michael R Sayre, Nathan J White, and Su-In Lee. CoAI: Cost-aware artificial intelligence for health care. *medRxiv*, 2021.

Kawin Ethayarajh and Dan Jurafsky. Attention flows are Shapley value explanations. *arXiv preprint arXiv:2105.14652*, 2021.

Lijie Fan, Shengjia Zhao, and Stefano Ermon. Adversarial localization network. In *Learning with limited labeled data: weak supervision and beyond, NIPS Workshop*, 2017.

Rongxin Fang, Chenglong Xia, Jennie L Close, Meng Zhang, Jiang He, Zhengkai Huang, Aaron R Halpern, Brian Long, Jeremy A Miller, Ed S Lein, et al. Conservation and divergence of cortical cell organization in human and mouse revealed by MERFISH. *Science*, 377(6601):56–62, 2022.

Robert M Fano. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, 1961.

Alton B Farris, Joseph Misdraji, Amitabh Srivastava, Alona Muzikansky, Vikram Deshpande, Gregory Y Lauwers, and Mari Mino-Kenudson. Sessile serrated adenoma: challenging discrimination from other serrated colonic polyps. *The American Journal of Surgical Pathology*, 32(1):30–35, 2008.

Andrea M Femino, Fredric S Fay, Kevin Fogarty, and Robert H Singer. Visualization of single RNA transcripts in situ. *Science*, 280(5363):585–590, 1998.

Jean Feng and Noah Simon. Sparse-input neural networks for high-dimensional nonparametric regression and classification. *arXiv preprint arXiv:1711.07592*, 2017.

Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer, 2015.

Greg Finak, Andrew McDavid, Masanao Yajima, Jingyuan Deng, Vivian Gersuk, Alex K Shalek, Chloe K Slichter, Hannah W Miller, M Juliana McElrath, Martin Prlic, et al.

MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. *Genome Biology*, 16(1):278, 2015.

François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5(9), 2004.

Raymond Fok and Daniel S Weld. In search of verifiability: Explanations rarely enable complementary performance in AI-advised decision making. *arXiv preprint arXiv:2305.07722*, 2023.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2950–2958, 2019.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

Christopher Frye, Colin Rowat, and Ilya Feige. Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability. *Advances in Neural Information Processing Systems*, 33:1229–1239, 2020.

Christopher Frye, Damien de Mijolla, Tom Begley, Laurence Cowton, Megan Stanley, and Ilya Feige. Shapley explainability on the data manifold. In *International Conference on Learning Representations*, 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

Rohan Gala, Agata Budzillo, Fahimeh Baftizadeh, Jeremy Miller, Nathan Gouwens, Anton Arkhipov, Gabe Murphy, Bosiljka Tasic, Hongkui Zeng, Michael Hawrylycz, et al. Consistent cross-modal identification of cortical neurons with coupled autoencoders. *Nature Computational Science*, 1(2):120–127, 2021.

Damien Garreau and Ulrike von Luxburg. Looking deeper into LIME. *arXiv preprint arXiv:2008.11092*, 2020.

Donald Geman and Bruno Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

Amirata Ghorbani and James Zou. Data Shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

Amirata Ghorbani and James Y Zou. Neuron Shapley: Discovering the responsible neurons. *Advances in Neural Information Processing Systems*, 33:5922–5932, 2020.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

Jacob Gildenblat and contributors. PyTorch library for CAM methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

Katharina Glatz, Bobbi Pritt, Dieter Glatz, Arndt Hartmann, Michael J O'Brien, and Hagen Blaszyk. A multinational, internet-based assessment of observer variability in the diagnosis of serrated colorectal polyps. *American Journal of Clinical Pathology*, 127(6):938–945, 2007.

Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis,

et al. Structure-based protein function prediction using graph convolutional networks. *Nature Communications*, 12(1):1–14, 2021.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42: 427–486, 2011.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

Nathan W Gouwens, Staci A Sorensen, Fahimeh Baftizadeh, Agata Budzillo, Brian R Lee, Tim Jarsky, Lauren Alfiler, Katherine Baker, Eliza Barkan, Kyla Berry, et al. Integrated morphoelectric and transcriptomic classification of cortical gabaergic cells. *Cell*, 183(4): 935–953, 2020.

Michel Grabisch, Jean-Luc Marichal, and Marc Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000.

Isabella N Grabski and Rafael A Irizarry. A probabilistic gene expression barcode for annotation of cell types from single-cell RNA-seq data. *Biostatistics*, 23(4):1150–1164, 2022.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and

Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):1–42, 2018.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

Prashnna K Gyawali, Xiaoxia Liu, James Zou, and Zihuai He. Ensembling improves stability and power of feature selection for deep learning models. In *Machine Learning in Computational Biology*, pages 33–45. PMLR, 2022.

Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.

Kenneth D Harris, Hannah Hochgerner, Nathan G Skene, Lorenza Magno, Linda Katona, Carolina Bengtsson Gonzales, Peter Somogyi, Nicoletta Kessaris, Sten Linnarsson, and Jens Hjerling-Leffler. Classes and continua of hippocampal CA1 inhibitory neurons revealed by single-cell transcriptomics. *PLoS Biology*, 16(6):e2006387, 2018.

He He, Hal Daumé III, and Jason Eisner. Cost-sensitive dynamic feature selection. In *ICML Inferning Workshop*, 2012.

He He, Paul Mineiro, and Nikos Karampatziakis. Active information acquisition. *arXiv preprint arXiv:1602.02181*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016b.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

Weijie He, Xiaohao Mao, Chao Ma, Yu Huang, José Miguel Hernàndez-Lobato, and Ting

Chen. BSODA: a bipartite scalable framework for online disease diagnosis. In *Proceedings of the ACM Web Conference 2022*, pages 2511–2521, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, 32, 2019.

Tom Heskes, Evi Sijben, Ioan Gabriel Bucur, and Tom Claassen. Causal Shapley values: Exploiting causal knowledge to explain individual predictions of complex models. *arXiv preprint arXiv:2011.01625*, 2020.

A Ali Heydari, Oscar A Davalos, Katrina K Hoyer, and Suzanne S Sindi. N-ACT: An interpretable deep learning model for automatic cell type and salient gene identification. *bioRxiv*, 2022.

Denis J Hilton. Conversational processes and causal explanation. *Psychological Bulletin*, 107 (1):65, 1990.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

Giles Hooker and Lucas Mentch. Please stop permuting features: An explanation and alternatives. *arXiv preprint arXiv:1905.03151*, 2019.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Jeremy Howard and Sylvain Gugger. FastAI: A layered API for deep learning. *Information*, 11(2):108, 2020.

Tomomichi Iizuka, Makoto Fukasawa, and Masashi Kameyama. Deep-learning-based imaging-classification identified cingulate island sign in dementia with Lewy bodies. *Scientific Reports*, 9(1):8944, 2019.

Ferenc Illés and Péter Kerényi. Estimation of the Shapley value by ergodic sampling. *arXiv preprint arXiv:1906.05224*, 2019.

Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382*, 2018.

Saachi Jain, Hadi Salman, Eric Wong, Pengchuan Zhang, Vibhav Vineet, Sai Vemprala, and Aleksander Madry. Missingness bias in model debugging. In *International Conference on Learning Representations*, 2021.

Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3959–3966, 2019.

Joseph D Janizek, Ayse B Dincer, Safiye Celik, Hugh Chen, William Chen, Kamila Naxerova, and Su-In Lee. Uncovering expression signatures of synergistic drug response using an ensemble of explainable AI models. *bioRxiv*, pages 2021–10, 2021.

Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable AI: A causal problem. In *International Conference on Artificial Intelligence and Statistics*, pages 2907–2916. PMLR, 2020.

Jos Jaspars, Miles Hewstone, and Frank D Fincham. Attribution theory and research: The state of the art. *Attribution theory and research: Conceptual, developmental and social dimensions*, pages 3–36, 1983.

Neil Jethani, Mukund Sudarshan, Yindalon Aphinyanaphongs, and Rajesh Ranganath. Have we learned to explain?: How interpretability methods can learn to encode predictions in their interpretations. In *International Conference on Artificial Intelligence and Statistics*, pages 1459–1467. PMLR, 2021a.

Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time Shapley value estimation. In *International Conference on Learning Representations*, 2021b.

Ian T Jolliffe. Principal components in regression analysis. In *Principal Component Analysis*, pages 129–155. Springer, 1986.

Matt Jordan, Jonathan Hayase, Alex Dimakis, and Sewoong Oh. Zonotope domains for lagrangian neural network verification. *Advances in Neural Information Processing Systems*, 35:8400–8413, 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Mohammad Kachuee, Orpaz Goldstein, Kimmo Kärkkäinen, Sajad Darabi, and Majid Sarrafzadeh. Opportunistic learning: Budgeted cost-sensitive learning from data streams. In *International Conference on Learning Representations*, 2018.

Mohammad Kachuee, Kimmo Karkkainen, Orpaz Goldstein, Davina Zamanzadeh, and Majid Sarrafzadeh. Cost-sensitive diagnosis and learning leveraging public health data. *arXiv preprint arXiv:1902.07102*, 2019.

Daniel Kahneman and Dale T Miller. Norm theory: Comparing reality to its alternatives. *Psychological Review*, 93(2):136, 1986.

Daniel Kahneman and Amos Tversky. The simulation heuristic. *Judgment Under Uncertainty: Heuristics and Biases*, pages 201–208, 1982.

Sergey Karayev, Tobias Baumgartner, Mario Fritz, and Trevor Darrell. Timely object recognition. *Advances in Neural Information Processing Systems*, 25, 2012.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

Omer Khalid, Sofyan Radaideh, Oscar W Cummings, Michael J O'Brien, John R Goldblum, and Douglas K Rex. Reinterpretation of histology of proximal colon polyps called hyperplastic in 2001. *World Journal of Gastroenterology*, 15(30):3767, 2009.

Zulqarnain Khan, Aria Masoomi, Davin Hill, and Jennifer Dy. Analyzing the effects of classifier lipschitzness on explainers. *arXiv preprint arXiv:2206.12481*, 2022.

Areum Daseul Kim, Rui Zhang, Xia Han, Kyoung Ah Kang, Mei Jing Piao, Young Hee Maeng, Weon Young Chang, and Jin Won Hyun. Involvement of glutathione and glutathione metabolizing enzymes in human colorectal cancer cell lines and tissues. *Molecular Medicine Reports*, 12(3):4314–4319, 2015.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, pages 2668–2677. PMLR, 2018.

Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. *arXiv preprint arXiv:1705.05598*, 2017.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 28, 2015.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Knowledge Discovery and Data Mining*, volume 96, pages 202–207, 1996.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for PyTorch. *arXiv preprint arXiv:2009.07896*, 2020.

Andrey N Kolmogorov. Foundations of the theory of probability, 1950.

Georgios Koutalellis, Konstantinos Stravodimos, Margaritis Avgeris, Konstantinos Mavridis, Andreas Scorilas, Andreas Lazaris, and Constantinos Constantinides. L-dopa decarboxylase (DDC) gene expression is related to outcome in patients with prostate cancer. *BJU International*, 110(6b):E267–E273, 2012.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

I Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. Problems with Shapley-value-based explanations as feature importance measures. *arXiv preprint arXiv:2002.11097*, 2020.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

Michael H Kutner, Christopher J Nachtsheim, John Neter, William Li, et al. *Applied linear statistical models*, volume 5. McGraw-Hill Irwin New York, 2005.

Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Samuel J Gershman, and Finale Doshi-Velez. Human evaluation of models built for interpretability. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 59–67, 2019.

Pierre-Simon Laplace. Mémoire sur les probabilités. *Mémoires de l'Académie Royale des sciences de Paris*, 1778:227–332, 1781.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. AT&T labs, 2010.

Changhee Lee, Fergus Imrie, and Mihaela van der Schaar. Self-supervision enhanced feature selection with correlated gates. In *International Conference on Learning Representations*, 2021.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

Ismael Lemhadri, Feng Ruan, and Rob Tibshirani. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pages 10–18. PMLR, 2021.

Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017a.

Ruoxin Li and Gerald Quon. scBFA: modeling detection patterns to mitigate technical noise in large-scale single-cell genomics data. *Genome Biology*, 20(1):1–20, 2019.

Xiang Li, Wei Li, Xiaodong Xu, and Wei Hu. Cell classification using convolutional neural networks in medical hyperspectral imagery. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pages 501–504. IEEE, 2017b.

Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In *International Conference on Machine Learning*, pages 6450–6459. PMLR, 2021.

Deron Liang, Chia-Chi Lu, Chih-Fong Tsai, and Guan-An Shih. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. *European Journal of Operational Research*, 252(2):561–572, 2016.

Moshe Lichman et al. UCI machine learning repository, 2013.

Ofir Lindenbaum, Uri Shaham, Erez Peterfreund, Jonathan Svirsky, Nicolas Casey, and Yuval Kluger. Differentiable unsupervised feature selection based on a gated Laplacian. *Advances in Neural Information Processing Systems*, 34:1530–1542, 2021.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

Hanqing Liu, Jingtian Zhou, Wei Tian, Chongyuan Luo, Anna Bartlett, Andrew Aldridge, Jacinta Lucero, Julia K Osteen, Joseph R Nery, Huaming Chen, et al. DNA methylation atlas of the mouse brain at single-cell resolution. *Nature*, 598(7879):120–128, 2021a.

Jonathan Liu, Vanessa Tran, Venkata Naga Pranathi Vemuri, Ashley Byrne, Michael Borja, Snigdha Agarwal, Ruofan Wang, Kyle Awayan, Abhishek Murti, Aris Taychameekiatchai, et al. Comparative analysis of MERFISH spatial transcriptomics with bulk and single-cell RNA sequencing. *bioRxiv*, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021b.

Tania Lombrozo. Simplicity and probability in causal explanation. *Cognitive Psychology*, 55 (3):232–257, 2007.

Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, 2018.

Romain Lopez, Achille Nazaret, Maxime Langevin, Jules Samaran, Jeffrey Regier, Michael I Jordan, and Nir Yosef. A joint model of unpaired data from scRNA-seq and spatial transcriptomics for imputing missing gene expression measurements. *arXiv preprint arXiv:1905.02269*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Malte D Luecken and Fabian J Theis. Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular Systems Biology*, 15(6):e8746, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749–760, 2018.

Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1): 2522–5839, 2020.

Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. EDDI: Efficient dynamic discovery of high-value

information with partial VAE. In *International Conference on Machine Learning*, pages 4234–4243. PMLR, 2019.

Chao Ma, Sebastian Tschiatschek, Richard Turner, José Miguel Hernández-Lobato, and Cheng Zhang. VAEM: a deep generative model for heterogeneous mixed type data. *Advances in Neural Information Processing Systems*, 33:11237–11247, 2020.

John Leslie Mackie. *The cement of the universe: A study of causation*. Oxford: Clarendon Press, 1974.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based Shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

Dina Mardaoui and Damien Garreau. An analysis of LIME for text data. *arXiv preprint arXiv:2010.12487*, 2020.

Jean-Luc Marichal and Pierre Mathonet. Weighted Banzhaf power and interaction indexes through weighted approximations of games. *European Journal of Operational Research*, 211(2):352–358, 2011.

Vivien Marx. Method of the year: spatially resolved transcriptomics. *Nature methods*, 18 (1):9–14, 2021.

Masayoshi Mase, Art B Owen, and Benjamin Seiler. Explaining black box decisions by Shapley cohort refinement. *arXiv preprint arXiv:1911.00467*, 2019.

Andrew McDavid. *Statistical Hurdle Models for Single Cell Gene Expression: Differential Expression and Graphical Modeling*. PhD thesis, University of Washington, 2016.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29), 2018.

Prem Melville, Maytal Saar-Tsechansky, Foster Provost, and Raymond Mooney. Active feature-value acquisition for classifier induction. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 483–486. IEEE, 2004.

Luke Merrick and Ankur Taly. The explanation game: Explaining machine learning models with cooperative game theory. *arXiv preprint arXiv:1909.08128*, 2019.

John Stuart Mill. *A system of logic, ratiocinative and inductive: Being a connected view of the principles of evidence and the methods of scientific investigation*. Harper, 1884.

Henry W Miller. Plan and operation of the health and nutrition examination survey, United States, 1971-1973. *DHEW publication no. (PHS)-Dept. of Health, Education, and Welfare (USA)*, 1973.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

Tim Miller, Piers Howe, and Liz Sonenberg. Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547*, 2017.

Alsu Missarova, Jaison Jain, Andrew Butler, Shila Ghazanfar, Tim Stuart, Maigan Brusko, Clive Wasserfall, Harry Nick, Todd Brusko, Mark Atkinson, et al. genebasis: an iterative approach for unsupervised selection of targeted gene panels from scrna-seq. *Genome Biology*, 22(1):1–22, 2021.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for Shapley value estimation. *arXiv preprint arXiv:2104.12199*, 2021.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799. PMLR, 2014.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in Neural Information Processing Systems*, 27, 2014.

Dov Monderer, Dov Samet, et al. Variations on the Shapley value. *Handbook of Game Theory*, 3:2055–2076, 2002.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 193–209, 2019.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Michael W Mosesson. Fibrinogen and fibrin structure and functions. *Journal of Thrombosis and Haemostasis*, 3(8):1894–1904, 2005.

Yadati Narahari. *Game Theory and Mechanism Design*, volume 4. World Scientific, 2014.

Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using VAEs. *Pattern Recognition*, 107:107501, 2020.

Michael E Nelson, Simone G Riva, and Ana Cvejic. SMaSH: a scalable, general marker gene identification framework for single-cell RNA-sequencing. *BMC Bioinformatics*, 23(1):1–16, 2022.

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems*, 14, 2001.

Yilin Ning, Mingxuan Liu, and Nan Liu. Shapley variable importance cloud for machine learning models. *arXiv preprint arXiv:2212.08370*, 2022.

Silje H Nordgard, Fredrik E Johansen, Grethe IG Alnæs, Elmar Bucher, Ann-Christine Syvänen, Bjørn Naume, Anne-Lise Børresen-Dale, and Vessela N Kristensen. Genome-wide analysis identifies 16q deletion associated with survival, molecular subtypes, mRNA expression, and germline haplotypes in breast cancer patients. *Genes, Chromosomes and Cancer*, 47(8):680–696, 2008.

Donald A Norman. Some observations on mental models. *Mental Models*, 7(112):7–14, 1983.

Silvia Noschese, Lionello Pasquini, and Lothar Reichel. Tridiagonal toeplitz matrices: properties and novel applications. *Numerical Linear Algebra With Applications*, 20(2):302–326, 2013.

Gherman Novakovsky, Nick Dexter, Maxwell W Libbrecht, Wyeth W Wasserman, and Sara Mostafavi. Obtaining genetics insights from deep learning via explainable artificial intelligence. *Nature Reviews Genetics*, pages 1–13, 2022.

Andrzej S Nowak. On an axiomatization of the Banzhaf value without the additivity axiom. *International Journal of Game Theory*, 26(1):137–141, 1997.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2 (11):e7, 2017.

OpenAI. Gpt-4 technical report, 2023.

Art B Owen. Sobol' indices and Shapley value. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):245–251, 2014.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505. IEEE, 2012.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

Judea Pearl. *Causality*. Cambridge University Press, 2009.

Leon Petrosjan and Georges Zaccour. Time-consistent Shapley value allocation of pollution cost reduction. *Journal of Economic Dynamics and Control*, 27(3):381–398, 2003.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

Emma Pierson and Christopher Yau. ZIFA: dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biology*, 16(1):241, 2015.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, pages 6638–6648, 2018.

Xiaoyan Qian, Kenneth D Harris, Thomas Hauling, Dimitris Nicoloutsopoulos, Ana B Muñoz-Manchado, Nathan Skene, Jens Hjerling-Leffler, and Mats Nilsson. Probabilistic cell typing enables fine mapping of closely related cell types in situ. *Nature Methods*, 17(1):101–106, 2020.

Peng Qiu. Embracing the dropouts in single-cell RNA-seq analysis. *Nature Communications*, 11(1):1–9, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in neural information processing systems*, 31, 2018.

Pranav Rajpurkar, Jeremy Irvin, Aarti Bagul, Daisy Ding, Tony Duan, Hershel Mehta, Brandon Yang, Kaylie Zhu, Dillon Laird, Robyn L Ball, et al. MURA: Large dataset for abnormality detection in musculoskeletal radiographs. *arXiv preprint arXiv:1712.06957*, 2017.

Daniel Ramsköld, Shujun Luo, Yu-Chieh Wang, Robin Li, Qiaolin Deng, Omid R Faridani, Gregory A Daniels, Irina Khrebtukova, Jeanne F Loring, Louise C Laurent, et al. Full-length mRNA-Seq from single-cell levels of RNA and individual circulating tumor cells. *Nature Biotechnology*, 30(8):777, 2012.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pages 463–471. Citeseer, 1998.

Samrudhdhi B Rangrej and James J Clark. A probabilistic hard attention model for sequentially observed scenes. *arXiv preprint arXiv:2111.07534*, 2021.

Marc'Aurelio Ranzato. On learning where to look. *arXiv preprint arXiv:1405.5488*, 2014.

Narges Razavian, Vincent J Major, Mukund Sudarshan, Jesse Burk-Rafel, Peter Stella, Hardev Randhawa, Seda Bilaloglu, Ji Chen, Vuthy Nguy, Walter Wang, et al. A validated, real-time prediction model for favorable outcomes in hospitalized COVID-19 patients. *NPJ Digital Medicine*, 3(1):1–13, 2020.

Stephen J Read and Amy Marcus-Newhall. Explanatory coherence in social explanations: A parallel distributed processing account. *Journal of Personality and Social Psychology*, 65(3):429, 1993.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

Davide Risso, Fanny Perraudeau, Svetlana Gribkova, Sandrine Dudoit, and Jean-Philippe Vert. A general and flexible method for signal extraction from single-cell RNA-seq data. *Nature Communications*, 9(1):1–17, 2018.

Dan R Robinson, Yi-Mi Wu, Pankaj Vats, Fengyun Su, Robert J Lonigro, Xuhong Cao, Shanker Kalyana-Sundaram, Rui Wang, Yu Ning, Lynda Hodges, et al. Activating ESR1 mutations in hormone-resistant metastatic breast cancer. *Nature Genetics*, 45(12):1446, 2013.

BP Roe, HJ Yand, J Zhu, Y Lui, I Stancu, et al. Boosted decision trees, an alternative to artificial neural networks. *Nucl. Instrm. Meth. A*, 543:577–584, 2005.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Luis M Ruiz, Federico Valenciano, and Jose M Zarzuelo. The family of least square values for transferable utility games. *Games and Economic Behavior*, 24(1-2):109–130, 1998.

Maytal Saar-Tsechansky, Prem Melville, and Foster Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, 2009.

Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Madry. Certified patch robustness via smoothed vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15137–15147, 2022.

Adriel Saporta, Xiaotong Gui, Ashwin Agrawal, Anuj Pareek, Steven QH Truong, Chanh DT Nguyen, Van-Doan Ngo, Jayne Seekins, Francis G Blankenberg, Andrew Y Ng, et al. Benchmarking saliency methods for chest x-ray interpretation. *medRxiv*, pages 2021–02, 2021.

Abhishek Sarkar and Matthew Stephens. Separating measurement and expression models clarifies confusion in single-cell RNA sequencing analysis. *Nature Genetics*, 53(6):770–777, 2021.

Federico Scala, Dmitry Kobak, Matteo Bernabucci, Yves Bernaerts, Cathryn René Cadwell, Jesus Ramon Castro, Leonard Hartmanis, Xiaolong Jiang, Sophie Laturnus, Elanine Miranda, et al. Phenotypic variation of transcriptomic cell types in mouse motor cortex. *Nature*, 598(7879):144–150, 2021.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. *arXiv preprint arXiv:2001.00396*, 2020.

Patrick Schwab and Walter Karlen. CXPlain: Causal explanations for model interpretation under uncertainty. In *Advances in Neural Information Processing Systems*, pages 10220–10230, 2019.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

Sofia Serrano and Noah A Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, 2019.

Sheel Shah, Eric Lubeck, Wen Zhou, and Long Cai. In situ transcription profiling of single cells reveals spatial organization of cells in the mouse hippocampus. *Neuron*, 92(2):342–357, 2016.

Sheel Shah, Yodai Takei, Wen Zhou, Eric Lubeck, Jina Yun, Chee-Huat Linus Eng, Noushin Koulena, Christopher Cronin, Christoph Karp, Eric J Liaw, et al. Dynamics and spatial genomics of the nascent transcriptome by intron seqFISH. *Cell*, 174(2):363–376, 2018.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317, 1953.

Hajin Shim, Sung Ju Hwang, and Eunho Yang. Joint active feature acquisition and classification with variable-size set encoding. *Advances in Neural Information Processing Systems*, 31, 2018.

Anthony F Shorrocks. Decomposition procedures for distributional analysis: a unified framework based on the Shapley value. Technical report, Mimeo, University of Essex, 1999.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

Emman Shubbar, Anikó Kovács, Shahin Hajizadeh, Toshima Z Parris, Szilárd Nemes, Katrin Gunnarsdóttir, Zakaria Einbeigi, Per Karlsson, and Khalil Helou. Elevated cyclin b2 expression in invasive breast carcinoma is associated with unfavorable clinical outcome. *BMC Cancer*, 13(1):1, 2013.

Grah Simon and Thouvenot Vincent. A projected stochastic gradient algorithm for estimating Shapley value applied in attribute importance. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 97–115. Springer, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3 (POPL):1–30, 2019.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Brett Snider, Bhumi Patel, and Edward McBean. Insights into co-morbidity and other risk factors related to COVID-19 within Ontario, Canada. *Frontiers in Artificial Intelligence*, page 85, 2021.

Dongyuan Song, Kexin Li, Zachary Hemminger, Roy Wollman, and Jingyi Jessica Li. scP-NMF: sparse gene encoding of single cells to facilitate gene selection for targeted gene profiling. *Bioinformatics*, 37:i358–i366, 2021.

Eunhye Song, Barry L Nelson, and Jeremy Staum. Shapley effects for global sensitivity analysis: Theory and computation. *SIAM/ASA Journal on Uncertainty Quantification*, 4 (1):1060–1083, 2016.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Geoffrey Stanley, Ozgun Gokce, Robert C Malenka, Thomas C Südhof, and Stephen R Quake. Continuous and discrete neuron types of the adult murine striatum. *Neuron*, 105 (4):688–699, 2020.

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):307, 2008.

Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 2010.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, 2014.

Erik Štrumbelj, Igor Kononenko, and M Robnik Šikonja. Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, 68(10): 886–904, 2009.

Tim Stuart, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M Mauck III, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902, 2019.

ATLS Subcommittee, International ATLS Working Group, et al. Advanced trauma life support (ATLS®): the ninth edition. *The Journal of Trauma and Acute Care Surgery*, 74(5):1363–1366, 2013.

Aravind Subramanian, Rajiv Narayan, Steven M Corsello, David D Peck, Ted E Natoli, Xiaodong Lu, Joshua Gould, John F Davis, Andrew A Tubelli, Jacob K Asiedu, et al. A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. *Cell*, 171(6):1437–1452, 2017.

Yu-Chi Sun, Xiaoyin Chen, Stephan Fischer, Shaina Lu, Huiqing Zhan, Jesse Gillis, and Anthony M Zador. Integrating barcoded neuroanatomy with spatial transcriptional profiling enables identification of gene correlates of projections. *Nature Neuroscience*, 24(6): 873–885, 2021.

Mukund Sundararajan and Amir Najmi. The many Shapley values for model explanation. *arXiv preprint arXiv:1908.08474*, 2019.

Mukund Sundararajan and Amir Najmi. The many Shapley values for model explanation. In *International Conference on Machine Learning*, pages 9269–9278. PMLR, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Saeid Asgari Taghanaki, Mohammad Havaei, Tess Berthier, Francis Dutil, Lisa Di Jorio, Ghassan Hamarneh, and Yoshua Bengio. Infomask: Masked variational latent representation to localize chest disease. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 739–747. Springer, 2019.

Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Nikola Tarashev, Kostas Tsatsaronis, and Claudio Borio. Risk attribution using the Shapley value: Methodology and policy applications. *Review of Finance*, 20(3):1189–1213, 2016.

Bosiljka Tasic, Zizhen Yao, Lucas T Graybuck, Kimberly A Smith, Thuc Nghi Nguyen, Darren Bertagnolli, Jeff Goldy, Emma Garren, Michael N Economo, Sarada Viswanathan, et al. Shared and distinct transcriptomic cell types across neocortical areas. *Nature*, 563 (7729):72–78, 2018.

Paul Thagard. Explanatory coherence. *Behavioral and Brain Sciences*, 12(3):435–502, 1989.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Daniel Shu Wei Ting, Carol Yim-Lui Cheung, Gilbert Lim, Gavin Siew Wei Tan, Nguyen D Quang, Alfred Gan, Haslina Hamzah, Renata Garcia-Franco, Ian Yew San Yeo, Shu Yen Lee, et al. Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. *Jama*, 318(22):2211–2223, 2017.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

Cole Trapnell. Defining cell types and states with single-cell genomics. *Genome Research*, 25(10):1491–1498, 2015.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *arXiv preprint arXiv:2305.04388*, 2023.

Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of SHAP explanations. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*, 2021.

Aad W Van der Vaart. *Asymptotic Statistics*, volume 3. Cambridge University Press, 2000.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Nazneen Rajani, et al. BERTology meets biology: Interpreting attention in protein language models. In *International Conference on Learning Representations*, 2020.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31:841, 2017.

Fay Wang, John Flanagan, Nan Su, Li-Chong Wang, Son Bui, Allissa Nielson, Xingyong Wu, Hong-Thuy Vo, Xiao-Jun Ma, and Yuling Luo. RNAscope: a novel in situ RNA

analysis platform for formalin-fixed, paraffin-embedded tissues. *The Journal of Molecular Diagnostics*, 14(1):22–29, 2012.

Jiaxuan Wang, Jenna Wiens, and Scott Lundberg. Shapley flow: A graph-based approach to interpreting model predictions. In *International Conference on Artificial Intelligence and Statistics*, pages 721–729. PMLR, 2021.

Tianhao Wang and Ruoxi Jia. Data Banzhaf: A data valuation framework with maximal robustness to learning stochasticity. *arXiv preprint arXiv:2205.15466*, 2022.

Xiao Wang, William E Allen, Matthew A Wright, Emily L Sylwestrak, Nikolay Samusik, Sam Vesuna, Kathryn Evans, Cindy Liu, Charu Ramakrishnan, Jia Liu, et al. Three-dimensional intact-tissue sequencing of single-cell transcriptional states. *Science*, 361 (6400), 2018.

Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. Transformer-based acoustic modeling for hybrid speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6874–6878. IEEE, 2020a.

Zifan Wang, Haofan Wang, Shakul Ramkumar, Piotr Mardziel, Matt Fredrikson, and Anupam Datta. Smoothed geometry for robust attribution. *Advances in Neural Information Processing Systems*, 33:13623–13634, 2020b.

Robert J Weber. Probabilistic values for games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, pages 101–119, 1988.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Jerry Wei, Arief Suriawinata, Bing Ren, Xiaoying Liu, Mikhail Lisovsky, Louis Vaickus, Charles Brown, Michael Baker, Naofumi Tomita, Lorenzo Torresani, et al. A petri dish for histopathology image analysis. In *Artificial Intelligence in Medicine*, pages 11–24. Springer, 2021.

BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, 2019.

Ross Wightman. PyTorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Brian Williamson and Jean Feng. Efficient nonparametric statistical inference on population feature importance using Shapley values. In *International Conference on Machine Learning*, pages 10282–10291. PMLR, 2020.

F Alexander Wolf, Philipp Angerer, and Fabian J Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1):1–5, 2018.

Newton ACS Wong, Linda P Hunt, Marco R Novelli, Neil A Shepherd, and Bryan F Warren. Observer agreement in the diagnosis of serrated polyps of the large bowel. *Histopathology*, 55(1):63–66, 2009.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon

Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.

Shawn Xu, Subhashini Venugopalan, and Mukund Sundararajan. Attribution in scale and space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9680–9689, 2020.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *International Conference on Machine Learning*. PMLR, 2020.

Zizhen Yao, Hanqing Liu, Fangming Xie, Stephan Fischer, Ricky S Adkins, Andrew I Aldridge, Seth A Ament, Anna Bartlett, M Margarita Behrens, Koen Van den Berge, et al. A transcriptomic and epigenomic cell atlas of the mouse primary motor cortex. *Nature*, 598(7879):103–110, 2021.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

H Peyton Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72, 1985.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

Hongkui Zeng. What is a cell type and how to define it? *Cell*, 185(15):2739–2755, 2022.

Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

Meng Zhang, Stephen W Eichhorn, Brian Zingg, Zizhen Yao, Kaelan Cotter, Hongkui Zeng, Hongwei Dong, and Xiaowei Zhuang. Spatially resolved cell atlas of the mouse primary motor cortex by MERFISH. *Nature*, 598(7879):137–143, 2021.

Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8(1):1–12, 2017.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. *arXiv preprint arXiv:1412.6856*, 2014.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do feature attribution methods correctly attribute features? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9623–9633, 2022.

Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *International Conference on Learning Representations*, 2017.

Appendix A

EXPLAINING BY REMOVING: A UNIFIED FRAMEWORK

A.1 Method Details

To preserve space, we refer readers to Appendix A of the original work (Covert et al., 2021).

A.2 Additive Model Proofs

LIME calculates feature attribution values by fitting a weighted regularized linear model to an *interpretable representation* of the input (Ribeiro et al., 2016). If we consider that the interpretable representation is binary (the default behavior in LIME's implementation), then the model is represented by a set function $u : \mathcal{P}(d) \mapsto \mathbb{R}$ when we take an expectation over the distribution of possible feature imputations. LIME is therefore equivalent to fitting an additive model to a set function, which means solving the optimization problem

$$\min_{\beta_0,\ldots,\beta_d} L(\beta_0, \ldots, \beta_d) + \Omega(\beta_1, \ldots, \beta_d),$$

where we define the weighted least squares component L as

$$L(\beta_0, \ldots, \beta_d) = \sum_{S \subseteq [d]} \pi(S) \Big(\beta_0 + \sum_{i \in S} \beta_i - u(S) \Big)^2. \tag{A.1}$$

For convenience, we refer to this as the weighted least squares (WLS) approach to summarizing set functions, and we show that several familiar attribution values can be derived by choosing different weighting kernels π and omitting the regularization term Ω.

A.2.1 Include Individual

Consider the weighting kernel $\pi_{\text{Inc}}(S) = \mathbb{1}(|S| \leq 1)$, which puts weight only on coalitions that have no more than one player. With this kernel, the WLS problem reduces to:

$$a_1, \dots a_d = \underset{\beta_0,\dots,\beta_d}{\arg\min} \ \Big(\beta_0 - u(\varnothing)\Big)^2 + \sum_{i=1}^{d} \Big(\beta_0 + \beta_i - u(\{i\})\Big)^2.$$

It is clear that the unique global minimizer of this problem is given by the following solution:

$$\begin{aligned} a_0 &= u(\varnothing) \\ a_i &= u(\{i\}) - u(\varnothing). \end{aligned}$$

The WLS approach will therefore calculate the attribution values $a_i = u(\{i\}) - u(\varnothing)$, which is equivalent to how Occlusion, PredDiff and CXPlain calculate local feature importance and how permutation tests and feature ablation (LOCO) summarize global feature importance (Breiman, 2001; Strobl et al., 2008; Zeiler and Fergus, 2014; Zintgraf et al., 2017; Lei et al., 2018; Schwab and Karlen, 2019).

A.2.2 Remove Individual

Consider the weighting kernel $\pi_{\text{Ex}}(S) = \mathbb{1}(|S| \geq d-1)$, which puts weight only on coalitions that are missing no more than one player. With this kernel, the WLS problem reduces to:

$$a_1, \dots, a_d = \underset{\beta_0,\dots,\beta_d}{\arg\min} \ \Big(\beta_0 + \sum_{j\in[d]} \beta_j - u([d])\Big)^2 + \sum_{i=1}^{d} \Big(\beta_0 + \sum_{j\in[d]\setminus i} \beta_j - u([d] \setminus i)\Big)^2.$$

It is clear that the unique global minimizer of this problem is given by the following solution:

$$\begin{aligned} a_0 &= u([d]) - \sum_{i\in[d]} a_i \\ a_i &= u([d]) - u([d] \setminus i). \end{aligned}$$

The WLS approach will therefore calculate the attribution values $a_i = u([d]) - u([d] \setminus i)$, which is equivalent to how the univariate predictors approach summarizes global feature importance (Guyon and Elisseeff, 2003).

A.2.3 Banzhaf Value

Consider the weighting kernel $\pi_{\mathrm{B}}(S) = 1$, which yields an unweighted least squares problem. This version of the WLS problem has been analyzed in prior work, which showed that the optimal coefficients are the Banzhaf values (Hammer and Holzman, 1992).

As an alternative proof, we demonstrate that a solution that uses the Banzhaf values is optimal by proving that its partial derivatives are zero. To begin, consider the following candidate solution, which uses the Banzhaf values $a_i = \psi_i(u)$ for $i \in [d]$ and a carefully chosen intercept term a_0:

$$
\begin{aligned}
a_0 &= \frac{1}{2^d} \sum_{S \subseteq [d]} u(S) - \frac{1}{2} \sum_{j=1}^{d} a_j \\
a_i &= \frac{1}{2^{d-1}} \sum_{S \subseteq [d] \setminus i} \Big(u(S \cup i) - u(S) \Big) = \psi_i(u).
\end{aligned}
$$

We can verify whether this is a solution to the unweighted least squares problem by checking if the partial derivatives are zero. We begin with the derivative for the intercept:

$$
\begin{aligned}
\frac{\partial}{\partial \beta_0} L(a_0, \ldots, a_d) &= 2 \sum_{S \subseteq [d]} \Big(a_0 + \sum_{j \in S} a_j - u(S) \Big) \\
&= 2 \Big(2^d a_0 + 2^{d-1} \sum_{j=1}^{d} a_j - \sum_{S \subseteq [d]} u(S) \Big) \\
&= 0.
\end{aligned}
$$

Next, we verify the derivatives for the other parameters a_i for $i \in [d]$:

$$\begin{aligned}
\frac{\partial}{\partial \beta_i} L(a_0, \ldots, a_d) &= 2 \sum_{T \supseteq \{i\}} \Big(a_0 + \sum_{j \in T} a_j - u(T) \Big) \\
&= 2 \Big(2^{d-1} a_0 + 2^{d-2} \sum_{j=1}^{d} a_j + 2^{d-2} a_i - \sum_{T \supseteq \{i\}} u(T) \Big) \\
&= 2^{d-1} a_i + \sum_{S \subseteq [d] \setminus i} \Big(u(S) - u(S \cup i) \Big) \\
&= 0.
\end{aligned}$$

Because the gradient is zero and the problem is jointly convex in $(\beta_0, \ldots, \beta_d)$, we conclude that the solution given above is optimal and unique. The optimal coefficients $(a_1, \ldots, a_d)$ are precisely the Banzhaf values $\psi_1(u), \ldots, \psi_d(u)$ of the cooperative game u.

A.2.4 Shapley Value

The weighted least squares problem is optimized by the Shapley value when we use the following weighting kernel:

$$\pi_{\text{Sh}}(S) = \frac{d-1}{\binom{d}{|S|} |S| (d - |S|)}.$$

Since this connection has been noted in other model explanation works, we direct readers to existing proofs (Charnes et al., 1988; Lundberg and Lee, 2017).

A.3 Axioms for Other Approaches

Here, we describe which Shapley axioms or Shapley-like axioms apply to other summarization approaches.

A.3.1 Additive Model Axioms

By fitting a regularized weighted least squares model to a cooperative game, as in LIME (Ribeiro et al., 2016), we effectively create an explanation mapping of the form $E : \mathcal{U} \mapsto \mathbb{R}^d$. We can show that this mapping satisfies a subset of the Shapley value axioms (Section 3.8). To do so, we make the following assumptions about the weighting kernel π and regularization function Ω (introduced in eq. 3.26):

1. The weighting kernel π is non-negative and finite for all $S \subseteq [d]$ except for possibly the sets $\varnothing$ and $[d]$.

2. The weighting kernel π satisfies the inequality

$$\begin{pmatrix} \mathbf{1} & X \end{pmatrix}^\top W \begin{pmatrix} \mathbf{1} & X \end{pmatrix} \succeq 0,$$

   where $X \in \mathbb{R}^{2^d \times d}$ contains an enumeration of binary representations for all subsets $S \subseteq [d]$, and $W \in \mathbb{R}^{2^d \times 2^d}$ is a diagonal matrix containing an enumeration of $\pi(S)$ for $S \subseteq [d]$. This ensures that the weighted least squares component of the objective is strictly convex.

3. The regularizer Ω is convex (e.g., the Lasso or ridge penalty).

We now address each property in turn for the weighted least squares (WLS) approach:

- (Efficiency) The WLS approach satisfies the efficiency property only when the weighting kernel is chosen such that $\pi(\varnothing) = \pi([d]) = \infty$. These weights are equivalent to the constraints $\beta_0 = u(\varnothing)$ and $\sum_{i \in [d]} \beta_i = u([d]) - u(\varnothing)$. In cooperative game theory, additive models with these constraints are referred to as *faithful linear approximations* (Hammer and Holzman, 1992).

- (Symmetry) The WLS approach satisfies the symmetry axiom as long as the weighting kernel π and the regularizer Ω are permutation-invariant (i.e., π is a function of the

subset size, and Ω is invariant to the ordering of parameters). To see this, consider an optimal solution with parameters $(\beta_0^*, \dots \beta_d^*)$. Swapping the coefficients β_i^* and β_j^* for features with identical marginal contributions gives the same objective value, so this is still optimal. The strict convexity of the objective function implies that there is a unique global optimum, so we conclude that $\beta_i^* = \beta_j^*$.

- (Null player) The null player property holds for the Shapley and Banzhaf weighting kernels π_{B} and π_{Sh}, but it does not hold for arbitrary π, Ω.

- (Linearity) The linearity property holds when the regularizer is set to $\Omega = 0$. This can be seen by viewing the solution to the WLS problem as a linear function of the response variables (Kutner et al., 2005).

- (Marginalism) Given two games u, u' where players have identical marginal contributions, we can see that $u' = u + c$ for some $c \in \mathbb{R}$. The WLS approach satisfies the marginalism property because it learns identical coefficients $\beta_1^*, \dots, \beta_d^*$ but different intercepts that vary by exactly c.

A.3.2 Feature Selection Axioms

The feature selection summarization techniques (MP, MIR, L2X, INVASE, REAL-X, EP, MM, FIDO-CA) satisfy properties that are similar to the Shapley value axioms. Each method outputs an optimal coalition $S^* \subseteq [d]$ rather than an allocation $a \in \mathbb{R}^d$, so the Shapley value axioms do not apply directly. However, we identify the following analogous properties:

- (Symmetry) If there are two players i, j with identical marginal contributions and there exists an optimal coalition S^* that satisfies $i \in S$ and $j \notin S$, then the coalition $\left(S^* \cup j\right) \setminus i$ is also optimal.

- (Null player) For a player i that makes zero marginal contribution, there must be an optimal solution S^* such that $i \notin S^*$.

- (Marginalism) For two games u, u' where all players have identical marginal contributions, the coalition S^* is optimal for u if and only if it is optimal for u'.

The feature selection explanations do not seem to satisfy properties that are analogous to the efficiency or linearity axioms. And unlike the attribution case, these properties are insufficient to derive a unique, axiomatic approach.

A.4 Consistency Proofs

Here, we restate and prove the results from Section 3.9, which relate to subset extensions of a model $f \in \mathcal{F}$ that are consistent with a probability distribution $q(\boldsymbol{x})$ (Definition 3.5). Both results can be shown using simple applications of basic probability laws.

Proposition A.1. *For a classification model $f \in \mathcal{F}$ that estimates a discrete $\boldsymbol{y}$'s conditional probability, there is a unique subset extension $F \in \mathfrak{F}$ that is consistent with $q(\boldsymbol{x})$,*

$$F(x_S) = \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} | x_S)}[f(x_S, \boldsymbol{x}_{\bar{S}})],$$

where $q(\boldsymbol{x}_{\bar{S}} \mid x_S)$ is the conditional distribution induced by $q(\boldsymbol{x})$, i.e., the distribution

$$q(x_{\bar{S}} \mid x_S) = \frac{q(x_{\bar{S}}, x_S)}{\int_{\boldsymbol{x}_{\bar{S}}} q(\boldsymbol{x}_{\bar{S}}, x_S)}.$$

Proof. We begin by assuming the existence of a subset function $F \in \mathfrak{F}$ that satisfies $q(\boldsymbol{y} \mid x) = F(x) = f(x)$ for all $x \in \mathcal{X}$ and consider how the probability axioms can be used to compute the conditional probability $q(\boldsymbol{y} \mid x_S)$ given only $q(\boldsymbol{x})$ and $q(\boldsymbol{y} \mid x)$.

For this proof we consider the case of discrete $\boldsymbol{x}$, but a similar argument can be used to prove the same result with continuous $\boldsymbol{x}$. Below, we provide a step-by-step derivation of $q(\boldsymbol{y} \mid x_S)$ that indicates which axioms or definitions are used in each step. (Axiom 1 refers

to the countable additivity property and Axiom 2 refers to Bayes rule.)

$$
\begin{aligned}
q(y \mid x_S) &= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} q(y, x_{\bar{S}} \mid x_S) && \text{(Axiom 1)} \\
&= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} \frac{q(y, x_S, x_{\bar{S}})}{q(x_S)} && \text{(Axiom 2)} \\
&= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} q(y \mid x_S, x_{\bar{S}}) \frac{q(x_S, x_{\bar{S}})}{q(x_S)} && \text{(Axiom 2)} \\
&= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} f(x_S, x_{\bar{S}}) \cdot q(x_{\bar{S}} \mid x_S) && \text{(Definition of } f\text{, Axiom 2)} \\
&= \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} | x_S)} \left[f(x_S, \boldsymbol{x}_{\bar{S}}) \right] && \text{(Definition of expectation)}
\end{aligned}
$$

This derivation shows that in order to be consistent with $q(\boldsymbol{x})$ according to the probability laws, F must be defined as follows:

$$
F(x_S) \equiv \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} | x_S)} \left[f(x_S, \boldsymbol{x}_{\bar{S}}) \right]. \tag{A.2}
$$

To complete the proof, we consider whether there are other ways of deriving F's behavior that may demonstrate inconsistency. The only other case to consider is whether we can derive a unique definition for $F(x_{S \setminus T})$ when beginning from $F(x)$ and when beginning from $F(x_S)$ for $T \subset S \subset [d]$. The first result is given by eq. A.2, and we derive the second result

as follows:

$$
\begin{aligned}
F(x_{S\setminus T}) &= q(y \mid x_{S\setminus T}) \\
&= \sum_{x_T \in \mathcal{X}_T} q(y, x_T \mid x_{S\setminus T}) && \text{(Axiom 1)} \\
&= \sum_{x_T \in \mathcal{X}_T} \frac{q(y, x_T, x_{S\setminus T})}{q(x_{S\setminus T})} && \text{(Axiom 2)} \\
&= \sum_{x_T \in \mathcal{X}_T} q(y \mid x_T, x_{S\setminus T}) \frac{q(x_T, x_{S\setminus T})}{q(x_{S\setminus T})} && \text{(Axiom 2)} \\
&= \sum_{x_T \in \mathcal{X}_T} q(y \mid x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Axiom 2)} \\
&= \sum_{x_T \in \mathcal{X}_T} F(x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Definition of } F\text{)} \\
&= \sum_{x_T \in \mathcal{X}_T} \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} \mid x_S)} \left[f(x_T, x_{S\setminus T}, \boldsymbol{x}_{\bar{S}}) \right] \cdot q(x_T \mid x_{S\setminus T}) && \text{(Definition of } F\text{)} \\
&= \sum_{x_T \in \mathcal{X}_T} \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} f(x_T, x_{S\setminus T}, x_{\bar{S}}) \cdot q(x_{\bar{S}} \mid x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Expectation)} \\
&= \sum_{x_T \in \mathcal{X}_T} \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} f(x_T, x_{S\setminus T}, x_{\bar{S}}) \cdot q(x_T, x_{\bar{S}} \mid x_{S\setminus T}) && \text{(Axiom 2)} \\
&= \mathop{\mathbb{E}}_{q(\boldsymbol{x}_T, \boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_{S\setminus T} = x_{S\setminus T})} \left[f(\boldsymbol{x}_T, x_{S\setminus T}, \boldsymbol{x}_{\bar{S}}) \right] && \text{(Definition of expectation)}
\end{aligned}
$$

This result shows that deriving $F(x_{S\setminus T})$ from $F(x)$ or from $F(x_S)$ yields an identical result. We conclude that our definition of F, which marginalizes out missing features with the conditional distribution induced by $q(\boldsymbol{x})$, provides the unique subset extension of f that is consistent with $q(\boldsymbol{x})$. □

Proposition A.2. *For a regression model $f \in \mathcal{F}$ that estimates a real-valued $\boldsymbol{y}$'s conditional expectation, there is a unique subset extension $F \in \mathfrak{F}$ that is consistent with $q(\boldsymbol{x})$,*

$$
F(x_S) = \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}} \mid x_S)} [f(x_S, \boldsymbol{x}_{\bar{S}})],
$$

where $q(\boldsymbol{x}_{\bar{S}} \mid x_S)$ is the conditional distribution induced by $q(\boldsymbol{x})$.

Proof. Unlike the previous proof, F does not directly define some $q(\boldsymbol{y} \mid x_S)$. However, F represents an estimate of the conditional expectation $\mathbb{E}[\boldsymbol{y} \mid x_S]$ for each $S \subseteq [d]$, so we can assume the existence of conditional distributions $q(\boldsymbol{y} \mid x_S)$ that satisfy

$$F(x_S) = \mathop{\mathbb{E}}_{q(\boldsymbol{y}|x_S)}[\boldsymbol{y}].$$

We show that our probability laws are sufficient to constrain the conditional expectation represented by F to have a unique definition for each $S \subset [d]$.

Consider how the probability laws can be used to compute $q(\boldsymbol{y} \mid x_S)$ given only $q(\boldsymbol{x})$ and the assumed $q(\boldsymbol{y} \mid x)$. For this proof, we consider the case of discrete $\boldsymbol{x}$ and $\boldsymbol{y}$, but a similar argument can be used to prove the same result for continuous $\boldsymbol{x}$ and $\boldsymbol{y}$. Below, we provide a step-by-step derivation of $q(\boldsymbol{y} \mid x_S)$ that indicates which axioms or definitions are used in each step. (Axiom 1 refers to the countable additivity property and Axiom 2 refers to Bayes rule.)

$$\begin{aligned}
q(y \mid x_S) &= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} q(y, x_{\bar{S}} \mid x_S) && \text{(Axiom 1)} \\
&= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} \frac{q(y, x_S, x_{\bar{S}})}{q(x_S)} && \text{(Axiom 2)} \\
&= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} q(y \mid x_S, x_{\bar{S}}) \frac{q(x_S, x_{\bar{S}})}{q(x_S)} && \text{(Axiom 2)} \\
&= \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} q(y \mid x_S, x_{\bar{S}}) \cdot q(x_{\bar{S}} \mid x_S) && \text{(Axiom 2)} \\
&= \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}}|x_S)} \left[q(y \mid x_S, \boldsymbol{x}_{\bar{S}}) \right] && \text{(Definition of expectation)}
\end{aligned}$$

This derivation shows that in order to be consistent with $q(\boldsymbol{x})$, the conditional distribution $q(\boldsymbol{y} \mid x_S)$ must be defined as follows:

$$q(\boldsymbol{y} \mid x_S) = \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}}|x_S)} [q(x_S, \boldsymbol{x}_{\bar{S}})].$$

Since F represents the expectation of these distributions, it can be derived as follows:

$$\begin{aligned}
F(x_S) &= \mathop{\mathbb{E}}_{q(\boldsymbol{y}|x_S)}[\boldsymbol{y}] && \text{(Definition of } F\text{)} \\
&= \sum_{y\in\mathcal{Y}} y \cdot q(y \mid x_S) && \text{(Definition of expectation)} \\
&= \sum_{y\in\mathcal{Y}} \sum_{x_{\bar{S}}\in\mathcal{X}_{\bar{S}}} y \cdot q(y \mid x_S, x_{\bar{S}}) \cdot q(x_{\bar{S}} \mid x_S) && \text{(Previous derivation)} \\
&= \sum_{x_{\bar{S}}\in\mathcal{X}_{\bar{S}}} \mathbb{E}[\boldsymbol{y} \mid x_S, x_{\bar{S}}] \cdot q(x_{\bar{S}} \mid x_S) && \text{(Interchanging order of sums)} \\
&= \sum_{x_{\bar{S}}} f(x_S, x_{\bar{S}}) \cdot q(x_{\bar{S}} \mid x_S) && \text{(Definition of } f\text{)} \\
&= \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}}|x_S)} \left[f(x_S, \boldsymbol{x}_{\bar{S}})\right] && \text{(Definition of expectation)}
\end{aligned}$$

According to this result, the probability laws imply that F must be defined as follows:

$$F(x_S) \equiv \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}}|x_S)} \left[f(x_S, \boldsymbol{x}_{\bar{S}})\right]. \tag{A.3}$$

To complete the proof, we consider whether there are other ways of deriving F's behavior that may demonstrate inconsistency. The only other case to consider is whether we can derive a unique definition for $F(x_{S\setminus T})$ when beginning from $F(x)$ and when beginning from $F(x_S)$ for $T \subset S \subset [d]$. The first result is given by eq. A.3, and we now derive the second

result. To begin, we derive $q(\boldsymbol{y} \mid \boldsymbol{x}_{S\setminus T})$ from $q(\boldsymbol{y} \mid \boldsymbol{x}_S)$:

$$
\begin{aligned}
q(y \mid x_{S\setminus T}) &= \sum_{x_T \in \mathcal{X}_T} q(y, x_T \mid x_{S\setminus T}) && \text{(Axiom 1)} \\
&= \sum_{x_T \in \mathcal{X}_T} \frac{q(y, x_T, x_{S\setminus T})}{q(x_{S\setminus T})} && \text{(Axiom 2)} \\
&= \sum_{x_T \in \mathcal{X}_T} q(y \mid x_T, x_{S\setminus T}) \frac{q(x_T, x_{S\setminus T})}{q(x_{S\setminus T})} && \text{(Axiom 2)} \\
&= \sum_{x_T \in \mathcal{X}_T} q(y \mid x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Axiom 2)} \\
&= \mathop{\mathbb{E}}_{q(\boldsymbol{x}_T \mid \boldsymbol{x}_{S\setminus T} = x_{S\setminus T})} \left[q(y \mid x_T, x_{S\setminus T}) \right] && \text{(Definition of expectation)}
\end{aligned}
$$

We can now derive $F(x_{S\setminus T})$ by taking the expectation of this distribution:

$$
\begin{aligned}
F(x_{S\setminus T}) &= \mathop{\mathbb{E}}_{q(\boldsymbol{y} \mid \boldsymbol{x}_{S\setminus T} = x_{S\setminus T})} [\boldsymbol{y}] && \text{(Definition of } F\text{)} \\
&= \sum_{y \in \mathcal{Y}} y \cdot q(y \mid x_{S\setminus T}) && \text{(Definition of expectation)} \\
&= \sum_{y \in \mathcal{Y}} \sum_{x_T \in \mathcal{X}_T} y \cdot q(y \mid x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Previous derivation)} \\
&= \sum_{x_T \in \mathcal{X}_T} \mathbb{E}[\boldsymbol{y} \mid x_T, x_{S\setminus T}] \cdot q(x_T \mid x_{S\setminus T}) && \text{(Interchanging order of sums)} \\
&= \sum_{x_T \in \mathcal{X}_T} F(x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Definition of } F\text{)} \\
&= \sum_{x_T \in \mathcal{X}_T} \sum_{x_{\bar{S}} \in \mathcal{X}_{\bar{S}}} f(x_T, x_{S\setminus T}, x_{\bar{S}}) \cdot q(x_{\bar{S}} \mid x_T, x_{S\setminus T}) \cdot q(x_T \mid x_{S\setminus T}) && \text{(Definition of } F\text{)} \\
&= \mathop{\mathbb{E}}_{q(\boldsymbol{x}_T, \boldsymbol{x}_{\bar{S}} \mid \boldsymbol{x}_{S\setminus T} = x_{S\setminus T})} \left[f(\boldsymbol{x}_T, x_{S\setminus T}, \boldsymbol{x}_{\bar{S}}) \right]] && \text{(Definition of expectation)}
\end{aligned}
$$

This result shows that deriving $F(x_{S\setminus T})$ from $F(x)$ or from $F(x_S)$ yields an identical result. We conclude that our definition of F, which marginalizes out missing features with the conditional distribution induced by $q(\boldsymbol{x})$, provides the unique subset extension of f that is consistent with $q(\boldsymbol{x})$. □

A.5 Conditional Distribution Approximations

We now describe how several approaches to removing features can be understood as approximations of marginalizing out missing features using their conditional distribution.

A.5.1 Separate Models

Shapley Net Effects (Lipovetsky and Conklin, 2001), SPVIM (Williamson and Feng, 2020) and the original IME (Štrumbelj et al., 2009) require training separate models for each subset of features. As in the main text, we denote these models as $\{f_S : S \subseteq [d]\}$. Similarly, the univariate predictors approach requires training models with individual features, and feature ablation requires training models with individual features held out. These models are used to make predictions with missing features, and they can be represented by the subset function $F(x_S) = f_S(x_S)$.

Note that this F satisfies the necessary properties to be a subset extension of $f_{[d]}$ (invariance to missing features and agreement with $f_{[d]}$ in the presence of all features) despite the fact that its predictions with held out features do not explicitly reference $f_{[d]}$. However, under the assumption that each f_S optimizes the population risk, we show that each f_S can be understood in relation to $f_{[d]}$.

For a regression task where each model f_S is trained with MSE loss, the model that optimizes the population risk is the conditional expectation $f_S(x_S) = \mathbb{E}[\boldsymbol{y} \mid x_S]$. Similarly, if the prediction task is classification and the loss function is cross entropy (or another strictly proper scoring function, see Gneiting and Raftery, 2007) then the model that optimizes the population risk is the conditional probability function or Bayes classifier $f_S(x_S) = p(\boldsymbol{y} \mid x_S)$. In both cases, if each f_S for $S \subseteq [d]$ optimizes the population risk, then we observe the following relationship between F and $f_{[d]}$:

$$F(x_S) = f_S(x_S) = \mathbb{E}\left[f_{[d]}(\boldsymbol{x}) \mid x_S\right].$$

This is precisely the approach of marginalizing out missing features from $f_{[d]}$ using their conditional distribution.

A.5.2 Missingness During Training

INVASE (Yoon et al., 2018) is based on a strategy of introducing missing features during model training (Appendix A.1). It trains a model where zeros (or potentially other values) take the place of removed features, so that the model can recognize these as missing values and make the best possible prediction given the available information.

We show here how this approach can be understood as an approximation of marginalizing out features with their conditional distribution. First, we note that replacing features with a default value is problematic if that value is observed in the dataset, because the model then faces ambiguity about whether the value is real or represents missingness. This issue can be resolved either by ensuring that the replacement value does not occur in the dataset, or by providing a mask vector $m \in \{0,1\}^d$ indicating missingness as an additional input to the model.

We assume for simplicity that this binary vector is provided as an additional input, and we let $x \odot m$ (the element-wise product) represent feature masking and $f(x \odot m, m)$ denote the model's prediction. This can be viewed as a technique for parameterizing a subset function $F \in \mathfrak{F}$ because it ensures invariance to the features that are not selected. Specifically, we can write

$$F(x_S) = f\big(x \odot m(S), m(S)\big),$$

where $m(S)$ is a binary vector with ones corresponding to the members in S. If we let $\boldsymbol{m}$ denote a random binary mask variable representing feature subsets, then the loss for training this model is:

$$\min_f \mathop{\mathbb{E}}_{p(\boldsymbol{m},\boldsymbol{x},\boldsymbol{y})} \Big[\ell\big(f(\boldsymbol{x} \odot \boldsymbol{m}, \boldsymbol{m}), \boldsymbol{y}\big)\Big].$$

Then, if $\boldsymbol{m}$ is independent from $(\boldsymbol{x}, \boldsymbol{y})$, we can decompose the loss as follows:

$$\begin{aligned}\mathop{\mathbb{E}}_{p(\boldsymbol{m},\boldsymbol{x},\boldsymbol{y})}\Big[\ell\big(f(\boldsymbol{x}\odot\boldsymbol{m},\boldsymbol{m}),\boldsymbol{y}\big)\Big] &= \mathop{\mathbb{E}}_{p(\boldsymbol{m})}\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\ell\big(f(\boldsymbol{x}\odot\boldsymbol{m},\boldsymbol{m}),\boldsymbol{y}\big)\Big] \\ &= \sum_{m\in\{0,1\}^d} p(m)\cdot\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\ell\big(f(\boldsymbol{x}\odot m,m),\boldsymbol{y}\big)\Big].\end{aligned}$$

For each value of m, we can regard $f(x\odot m, m)$ as a separate function on the specified subset of features $\{x_i : m_i = 1\}$. Then, for classification tasks using cross entropy loss, the objective is optimized by the function f^* such that

$$f^*(x\odot m, m) = p(\boldsymbol{y}\mid x_S),$$

where $S = \{i \in [d] : m_i = 1\}$. A similar result holds for other strictly proper scoring functions (Gneiting and Raftery, 2007) and for regression tasks trained with MSE loss (with the conditional probability replaced by the conditional expectation). In these cases, the result is equivalent to marginalizing out missing features according to their conditional distribution.

One issue with INVASE, noted by Jethani et al. (2021a), is that the mask variable $\boldsymbol{m}$ is not independent from the input vector $\boldsymbol{x}$. Intuitively, this means that the selected features can communicate information about the held out features, which then inform the model's prediction about the label $\boldsymbol{y}$. The INVASE model therefore may not approximate $p(\boldsymbol{y}\mid x_S)$, potentially invalidating its information-theoretic interpretation in terms of KL divergence minimization (Section 3.9.3).

Nonetheless, it is possible to learn a model with missingness introduced at training time that approximates marginalizing out features using their conditional distribution. It suffices to sample masks during training independently from the model input, e.g., by sampling masks uniformly at random (Jethani et al., 2021a) or according to the distribution described below.

A.5.3 Surrogate Models

To remove features from an existing model f, we can train a surrogate model to match its predictions when features are held out. This technique, described by Frye et al. (2021) and implemented by L2X (Chen et al., 2018a) and REAL-X (Jethani et al., 2021a), can also approximate marginalizing out features using their conditional distribution.

To train such a model, we require a mechanism for removing features. We let f denote the original model and g the surrogate, and similar to the model trained with missingness, we can remove features using a mask variable $m \in \{0,1\}^d$. Frye et al. (2021) suggest replacing held out features using a value that does not occur in the training set, but we can also provide the mask variable as an input to the surrogate. We therefore represent the surrogate's predictions as $g(x \odot m, m)$, where g can be understood as a subset function $F \in \mathfrak{F}$ (with m defined as in the case of training with missingness):

$$F(x_S) = g\big(x \odot m(S), m(S)\big).$$

The surrogate is then trained using the following objective function:

$$\min_g \mathop{\mathbb{E}}_{p(\boldsymbol{m},\boldsymbol{x})} \Big[\ell\big(g(\boldsymbol{x} \odot \boldsymbol{m}, \boldsymbol{m}), f(\boldsymbol{x})\big)\Big]. \tag{A.4}$$

If $\boldsymbol{m}$ is sampled independently from $\boldsymbol{x}$, then we can decompose the loss as follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{p(\boldsymbol{m},\boldsymbol{x})} \Big[\ell\big(g(\boldsymbol{x} \odot \boldsymbol{m}, \boldsymbol{m}), f(\boldsymbol{x})\big)\Big] &= \mathop{\mathbb{E}}_{p(\boldsymbol{m})} \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \Big[\ell\big(g(\boldsymbol{x} \odot \boldsymbol{m}, \boldsymbol{m}), f(\boldsymbol{x})\big)\Big] \\
&= \sum_{m \in \{0,1\}^d} p(m) \cdot \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \Big[\ell\big(g(\boldsymbol{x} \odot m, m), f(\boldsymbol{x})\big)\Big].
\end{aligned}$$

For each value of m, we can regard $g(x \odot m, m)$ as a separate function on the specified subset of features $\{x_i : m_i = 1\}$. Then, for specific loss functions, we can reason about the optimal surrogate that matches the original model f most closely. For models that are compared using MSE loss, we point to a result from Covert et al. (2020) which shows that to match

f's predictions in the sense of MSE loss, the optimal surrogate function g^* is given by

$$g^*(x \odot m, m) = \mathbb{E}[f(\boldsymbol{x}) \mid x_S], \tag{A.5}$$

where $S = \{i \in [d] : m_i = 1\}$. This result justifies the approach used by Frye et al. (2021). However, while Frye et al. (2021) focus on MSE loss, we also find that a cross entropy loss can be used for classification tasks. When the original model f and g both output discrete probability distributions, their predictions can be compared through a soft cross entropy loss, which we define as follows:

$$H(a, b) = -\sum_j a_j \log b_j.$$

Now, for classifications models that are compared via cross entropy loss, we point to a similar result from Covert et al. (2020) showing that the optimal surrogate model is once again given by marginalizing out the missing features using their conditional distribution. Notably, these results require that $\boldsymbol{x}$ and $\boldsymbol{m}$ are sampled independently during training, which is a property that is satisfied by Frye et al. (2021) and Jethani et al. (2021a) (REAL-X), but not Chen et al. (2018a) (L2X).

The only detail left to specify is the distribution for $\boldsymbol{m}$ in eq. A.4. Any distribution that places mass on all $m \in \{0, 1\}^d$ should suffice, at least in principle. Masks could be sampled uniformly at random, but for models with large numbers of features, this places nearly all the probability mass on subsets with approximately half of the features included. We therefore opt to use a distribution that samples the *subset size* uniformly at random. In our experiments, we sample masks m as follows:

1. Sample $k \in \{0, 1, \ldots, d\}$ uniformly at random.

2. Sample k indices $(i_1, \ldots, i_k)$ from $[d]$ at random and without replacement.

3. Set m with $m_i = \mathbb{1}\big(i \in \{i_1, \ldots, i_k\}\big)$.

A.6 Information-Theoretic Connections in Regression

Here, we describe probabilistic interpretations of each explanation method's underlying set function (Section 3.6) in the context of regression models rather than classification models (Section 3.9.3). We assume that the models are evaluated using MSE loss, and, as in the main text, we assume model optimality. The different set functions have the following interpretations.

- The set function $u_x(S) = F(x_S)$ quantifies the response variable's conditional expectation:

$$u_x(S) = \mathbb{E}[\boldsymbol{y} \mid x_S]. \tag{A.6}$$

  This set function lets us examine each feature's true relationship with the response variable.

- The set function $v_{xy}(S) = -\ell\big(F(x_S), y\big)$ quantifies the squared distance between the model output and the correct label:

$$v_{xy}(S) = -\Big(\mathbb{E}[\boldsymbol{y} \mid x_S] - y\Big)^2. \tag{A.7}$$

  Under the assumption that the response variable's conditional distribution is Gaussian, this represents the pointwise mutual information between x_S and y (up to factors that depend on S):

$$\begin{aligned} \mathrm{I}(y; x_S) = & -\log p(y) - \frac{1}{2}\log 2\pi - \frac{1}{2}\log \mathrm{Var}(\boldsymbol{y} \mid x_S) \\ & - \frac{1}{2}\underbrace{\Big(\mathbb{E}[\boldsymbol{y} \mid x_S] - y\Big)^2}_{v_{xy}(S)} / \mathrm{Var}(\boldsymbol{y} \mid x_S). \end{aligned} \tag{A.8}$$

- The set function $v_x(S) = -\,\mathbb{E}_{p(\boldsymbol{y}|x)}\big[\ell\big(F(x_S), \boldsymbol{y}\big)\big]$ quantifies the squared difference of

the conditional expectation from the model output, up to a constant value:

$$v_x(S) = -\Big(\mathbb{E}[\boldsymbol{y} \mid x] - \mathbb{E}[\boldsymbol{y} \mid x_S]\Big)^2 + c. \tag{A.9}$$

Under the assumption that the response variable's distribution conditioned on $\boldsymbol{x} = x$ and $\boldsymbol{x}_S = x_S$ are both Gaussian, this quantity has a relationship with the KL divergence between $p(\boldsymbol{y} \mid x)$ and $p(\boldsymbol{y} \mid x_S)$:

$$\begin{aligned} & D_{\mathrm{KL}}\big(p(\boldsymbol{y} \mid x) \,\big|\big|\, p(\boldsymbol{y} \mid x_S)\big) \\ = & \frac{1}{2}\log\frac{\mathrm{Var}(\boldsymbol{y} \mid x_S)}{\mathrm{Var}(\boldsymbol{y} \mid x)} - \frac{1}{2} \\ & + \Big(\mathrm{Var}(\boldsymbol{y} \mid x) + \underbrace{\big(\mathbb{E}[\boldsymbol{y} \mid x] - \mathbb{E}[\boldsymbol{y} \mid x_S]\big)^2}_{v_x(S)}\Big) \Big/ \Big(2 \cdot \mathrm{Var}(\boldsymbol{y} \mid x_S)\Big). \end{aligned} \tag{A.10}$$

- The set function $v(S) = -\,\mathbb{E}_{\boldsymbol{xy}}\Big[\ell\big(F(\boldsymbol{x}_S), \boldsymbol{y}\big)\Big]$ quantifies the explained variance in the response variable $\boldsymbol{y}$, up to a constant value:

$$v(S) = \mathrm{Var}(\boldsymbol{y}) - \mathbb{E}[\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_S)] + c. \tag{A.11}$$

Using the entropy maximizing property of the Gaussian distribution (Cover and Thomas, 2012), we see that the explained variance has the following relationship with the mutual information between $\boldsymbol{x}_S$ and the response variable $\boldsymbol{y}$:

$$\begin{aligned} \mathrm{I}(\boldsymbol{y}; \boldsymbol{x}_S) &= H(\boldsymbol{y}) - \mathbb{E}\left[H(\boldsymbol{y} \mid \boldsymbol{x}_S)\right] \\ &\geq H(\boldsymbol{y}) - \frac{1}{2}\,\mathbb{E}\left[\log\left(2\pi e \cdot \mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_S)\right)\right] \\ &\geq H(\boldsymbol{y}) - \frac{1}{2}\log 2\pi e - \frac{1}{2}\log\underbrace{\mathbb{E}\left[\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_S)\right]}_{v(S)}. \end{aligned} \tag{A.12}$$

Equality is achieved in the first bound if the distribution $p(\boldsymbol{y} \mid x_S)$ is Gaussian. The

second bound is due to Jensen's inequality.

- The set function $w_x(S) = -\ell\big(F(x_S), f(x)\big)$ quantifies the squared difference between the model output and the expected model output when conditioned on a subset of features:

$$w_x(S) = -\Big(f(x) - \mathbb{E}[f(\boldsymbol{x}) \mid x_S]\Big)^2. \tag{A.13}$$

If we assume that $p(f(\boldsymbol{x}) \mid \boldsymbol{x}_S)$ is a Gaussian distribution, then we can view this as a component in the KL divergence between $p(f(\boldsymbol{x}) \mid \boldsymbol{x})$ (a deterministic distribution) and $p(f(\boldsymbol{x}) \mid x_S)$:

$$\begin{aligned}
&D_{\mathrm{KL}}\big(p(f(\boldsymbol{x}) \mid x) \,\big|\big|\, p(f(\boldsymbol{x}) \mid x_S)\big) \\
&= \lim_{\sigma \to 0} \frac{1}{2} \log \frac{\mathrm{Var}(f(\boldsymbol{x}) \mid x_S)}{\sigma^2} - \frac{1}{2} \\
&\qquad + \Big(\sigma^2 + \underbrace{\big(f(x) - \mathbb{E}[f(\boldsymbol{x}) \mid x_S]\big)^2}_{w_{\boldsymbol{x}}(S)}\Big) \Big/ \Big(2\mathrm{Var}(f(\boldsymbol{x}) \mid x_S)\Big) \\
&= \infty.
\end{aligned} \tag{A.14}$$

This result is somewhat contrived, however, because the KL divergence is ultimately infinite due to $p(f(\boldsymbol{x}) \mid \boldsymbol{x})$ being deterministic. An alternative interpretation may be possible if we consider models that predict both the mean and standard deviation for a Gaussian response distribution.

- The set function $w(S) = -\,\mathbb{E}_{\boldsymbol{x}}\Big[\ell\big(F(\boldsymbol{x}_S), f(\boldsymbol{x})\big)\Big]$ quantifies the explained variance in the model output, up to a constant value:

$$w(S) = \mathrm{Var}\Big(f(\boldsymbol{x})\Big) - \mathbb{E}\Big[\mathrm{Var}\big(f(\boldsymbol{x}) \mid \boldsymbol{x}_S\big)\Big] + c. \tag{A.15}$$

This is the variance decomposition result presented for Shapley Effects (Owen, 2014). Using the same entropy maximizing property of the Gaussian distribution, we can also

see that the explained variance is related to the mutual information with the model output, which can be viewed as a random variable $f(\boldsymbol{x})$:

$$\begin{aligned}
\mathrm{I}\big(f(\boldsymbol{x}); \boldsymbol{x}_S\big) &= H\big(f(\boldsymbol{x})\big) - \mathbb{E}\left[H(f(\boldsymbol{x}) \mid \boldsymbol{x}_S)\right] \\
&\geq H\big(f(\boldsymbol{x})\big) - \frac{1}{2}\,\mathbb{E}\left[\log\big(2\pi e \cdot \mathrm{Var}(f(\boldsymbol{x}) \mid \boldsymbol{x}_S)\big)\right] \\
&\geq H\big(f(\boldsymbol{x})\big) - \frac{1}{2}\log 2\pi e - \frac{1}{2}\log \underbrace{\mathbb{E}\left[\mathrm{Var}\big(f(\boldsymbol{x}) \mid \boldsymbol{x}_S\big)\right]}_{w(S)}
\end{aligned} \tag{A.16}$$

Equality is achieved in the first bound if $f(\boldsymbol{x})$ has a Gaussian distribution when conditioned on $\boldsymbol{x}_S = x_S$. The second bound is due to Jensen's inequality.

These results are analogous to those from the classification case, and they show that each explanation method's set function has an information-theoretic interpretation even in the context of regression tasks. However, the regression case requires stronger assumptions about the data distribution to yield these information-theoretic links (i.e., Gaussian distributions). To avoid strong distributional assumptions, it is more conservative to interpret these quantities in terms of Euclidean distances (eqs. A.7, A.9 and A.13) and conditional variances (eqs. A.11 and A.15).

A.7 Experiment Details

This section provides additional details about models, datasets and hyperparameters, as well as some additional results.

A.7.1 Hyperparameters

The original models used for each dataset are:

- For the census income dataset, we trained a LightGBM model with a maximum of 10 leaves per tree and a learning rate of 0.05 (Ke et al., 2017).

- For MNIST, we trained a 14-layer CNN consisting of convolutional layers with kernel size 3, max pooling layers, and ELU activations (Clevert et al., 2015). The output was produced by flattening the convolutional features and applying two fully connected layers, similar to the VGG architecture (Simonyan and Zisserman, 2014). We trained the model with Adam using a learning rate of 10^{-3} (Kingma and Ba, 2014).

- For the BRCA dataset, we trained a ℓ_1 regularized logistic regression model and selected the regularization parameter using a validation set.

For the BRCA dataset, to avoid overfitting, we also randomly selected a subset of 100 genes to analyze out of 17,814 total. To ensure that a sufficient number BRCA-associated genes were selected, we tried 10 random seeds for the gene selection step and selected the seed whose 100 genes achieved the best performance (displayed in Table A.1). A small portion of missing expression values were imputed with their mean, and the data was centered and normalized prior to fitting the regularized logistic regression model.

When generating explanations using feature removal approaches that required sampling multiple values for the missing features (marginalizing with uniform, product of marginals, or joint marginal), we used 512 samples for the census income and MNIST datasets and 372 for the BRCA dataset (the size of the train split).

As described in the main text (Section 3.11) and in Appendix A.5, we trained surrogate models to represent marginalizing out features according to the conditional distribution. Our surrogate models were trained as follows:

- For the census income data, the surrogate was a MLP with a masking layer and four hidden layers of size 128 followed by ELU activations. During training, the mask variable was sampled according to the procedure described in Appendix A.5. Our masking layer replaced missing values with -1 (a value that did not occur in the dataset) and also appended the mask as an additional set of features, which improved its ability to match the original model's predictions.

Table A.1: List of genes analyzed.

Genes 1-17	Genes 18-34	Genes 35-51	Genes 52-68	Genes 69-85	Genes 86-100
OSTbeta	NBR2	TSHR	HPS4	GRINA	C20orf111
STATH	CCDC64	C7	ZFPM1	YTHDF3	OMA1
MAPK10	NUP210	CRYBB2	OAS2	TMCC1	NCAPH2
PLEKHG5	HEMGN	PPAPDC3	TUBA1C	UBE1DC1	GPX2
ERO1L	SLC25A3	TXNL4B	OR8K5	C6orf15	BPY2C
ZNF711	LEF1	CHST9	THSD3	PDE6A	ZNF324
ZNF385	MVD	HACE1	ATP6V0C	PEO1	CDC27
OR52E8	OTUD3	AYTL1	RAB22A	TMEM52	CCNB2
SLC5A11	KIAA1949	PRSS35	AP1B1	PARP1	CNOT7
P4HA3	SLC44A3	ZNF408	CTAGE6	GSS	BIRC3
LHFPL4	ZNF775	DDC	C6orf26	RDH11	GAL3ST3
MGC33657	THY1	CSTL1	ESR1	STXBP1	PLEKHM1
CAPZB	DYNC1I2	OR2F1	UPK3B	ACLY	SPOCD1
RBM15B	CYP1A1	C12orf50	ROBO4	TMSB10	PENK
C1orf176	SPTA1	SH3YL1	TMEFF1	TUBB	TAS2R9
KLF3	CLEC4M	SNUPN	KIAA1279	LIPK	
OLFM4	RXFP3	COL25A1	ZFP36L1	HRC	

- For MNIST, the surrogate was a CNN with an identical architecture to the original model (see above) except for a masking layer at the input. The masking layer replaced missing values with zeros and appended the mask along the channels dimension.

- For the BRCA data, the surrogate was an MLP with two hidden layers of size 64 followed by ELU activations. The masking layer replaced missing values with their mean and appended the mask as an additional set of features.

A.7.2 Additional Results

We present two supplementary results for the experiments described in Section 3.11. Figure A.1 shows more examples of MNIST explanations using the combination of the conditional distribution and Shapley value (i.e., LossSHAP). These explanations consistently

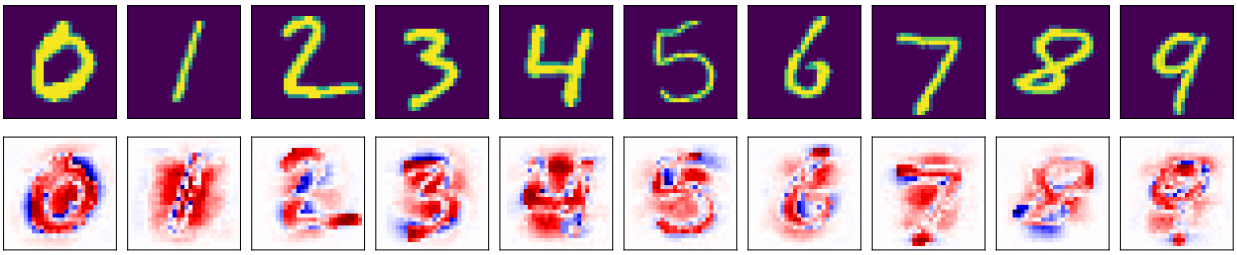


Figure A.1: MNIST prediction loss explanations using LossSHAP (feature removal with the conditional distribution and summary with the Shapley value).

highlight important pixels within the digit as well as empty regions that distinguish the digit from other possible classes (e.g., see the digits 3, 4 and 9).

Figure A.2 quantifies the similarity between dataset loss explanations for the BRCA dataset using their correlation (top) and Spearman rank correlation (bottom). We see patterns in these explanations that are similar to those seen with the census dataset: explanations generated using Shapley values are relatively similar to those that remove individual features or use Banzhaf values, while the include individual technique tends to differ most from the others. Because these plots visualize correlation rather than Euclidean distance, we can also see that Banzhaf value explanations are correlated with those that use the mean when included technique, which was predicted by our theory (Section 3.8). Interestingly, the Shapley value explanations are more strongly correlated across different removal strategies than they are to explanations that use other summarization strategies but that remove features in the same way (see bottom row of Figure A.2).

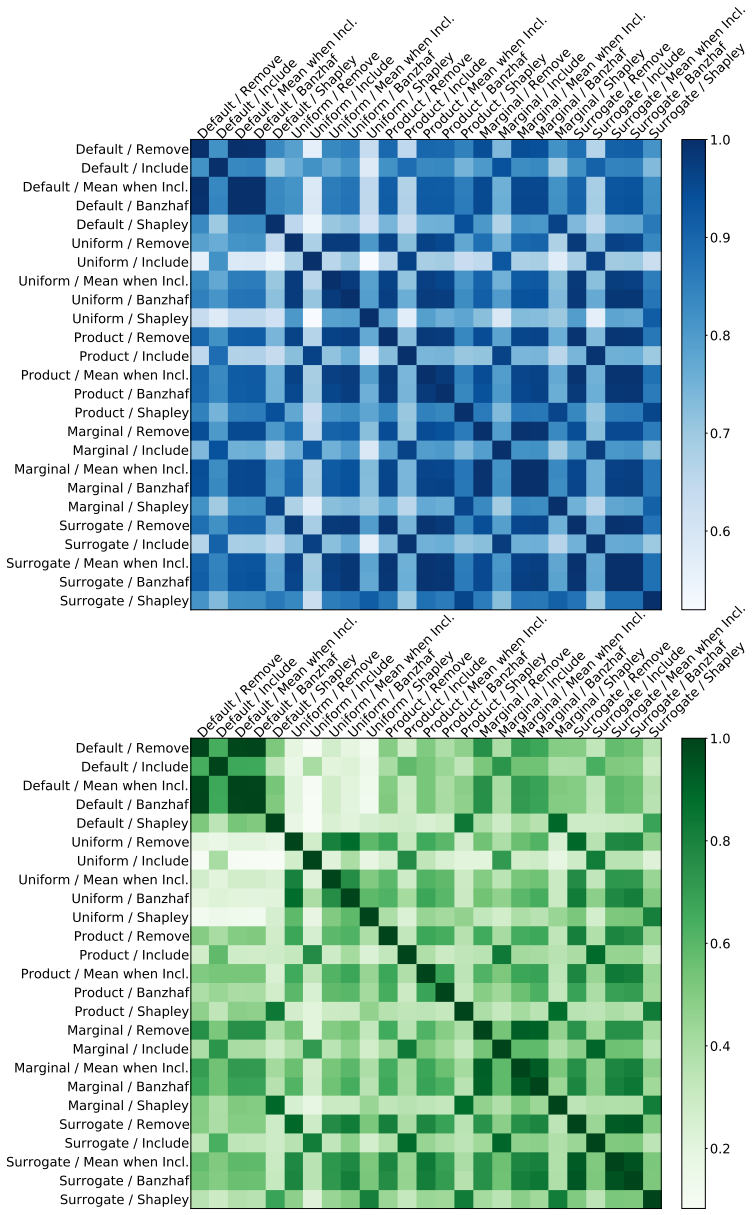


Figure A.2: Mean correlation (top) and Spearman rank correlation (bottom) between different explanation methods on the BRCA dataset.

Appendix B

ROBUSTNESS OF REMOVAL-BASED EXPLANATIONS

B.1 Proofs: Prediction Robustness to Input Perturbations

We now re-state and prove our results from Section 4.4 related to input perturbations.

Lemma B.1. *When removing features using either the* ***baseline*** *or* ***marginal*** *approaches, the prediction function $f(\boldsymbol{x}_S)$ for any feature set $\boldsymbol{x}_S$ is L-Lipschitz continuous:*

$$|f(x_S) - f(x'_S)| \leq L \cdot \|x_S - x'_S\|_2 \quad \forall\, x_S, x'_S \in \mathbb{R}^{|S|}.$$

Proof. The key insight for this result is that removing features using either the baseline or marginal approach is equivalent to using a distribution $q(\boldsymbol{x}_{\bar{S}})$ that does not depend on the observed features. Because of that, we have:

$$\begin{aligned}
|f(x_S) - f(x'_S)| &= \Big| \int f(x_S, x_{\bar{S}}) q(x_{\bar{S}}) dx_{\bar{S}} - \int f(x'_S, x_{\bar{S}}) q(x_{\bar{S}}) dx_{\bar{S}} \Big| \\
&\leq \int \big| f(x_S, x_{\bar{S}}) - f(x'_S, x_{\bar{S}}) \big| \cdot q(x_{\bar{S}}) dx_{\bar{S}} \\
&\leq L \cdot \|x_S - x'_S\|_2 \cdot \int q(x_{\bar{S}}) dx_{\bar{S}} \\
&= L \cdot \|x_S - x'_S\|_2.
\end{aligned}$$

□

Lemma B.2. *When removing features using the* ***conditional*** *approach, the prediction function $f(\boldsymbol{x}_S)$ for a feature set $\boldsymbol{x}_S$ satisfies*

$$|f(x_S) - f(x'_S)| \leq L \cdot \|x_S - x'_S\|_2 + 2B \cdot d_{TV}\Big(p(\boldsymbol{x}_{\bar{S}} \mid x_S), p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big),$$

where the total variation distance is defined via the L^1 functional distance as

$$d_{TV}\Big(p(\boldsymbol{x}_{\bar{S}} \mid x_S), p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big) \equiv \frac{1}{2}\Big\|p(\boldsymbol{x}_{\bar{S}} \mid x_S) - p(\boldsymbol{x}_{\bar{S}} \mid x'_S)\Big\|_1.$$

Proof. In this case, we must be careful to account for the different distributions used when marginalizing out the removed features.

$$\begin{aligned}
|f(x_S) - f(x'_S)| &= \Big|\int f(x_S, x_{\bar{S}})p(x_{\bar{S}} \mid x_S)dx_{\bar{S}} - \int f(x'_S, x_{\bar{S}})p(x_{\bar{S}} \mid x'_S)dx_{\bar{S}}\Big| \\
&\leq \int \big|f(x_S, x_{\bar{S}})p(x_{\bar{S}} \mid x_S) - f(x'_S, x_{\bar{S}})p(x_{\bar{S}} \mid x'_S)\big|dx_{\bar{S}} \\
&\leq \int \big|f(x_S, x_{\bar{S}})p(x_{\bar{S}} \mid x_S) - f(x'_S, x_{\bar{S}})p(x_{\bar{S}} \mid x_S)\big|dx_{\bar{S}} \\
&\quad + \int \big|f(x'_S, x_{\bar{S}})p(x_{\bar{S}} \mid x_S) - f(x'_S, x_{\bar{S}})p(x_{\bar{S}} \mid x'_S)\big|dx_{\bar{S}} \\
&\leq \int \big|f(x_S, x_{\bar{S}}) - f(x'_S, x_{\bar{S}})\big| \cdot p(x_{\bar{S}} \mid x_S)dx_{\bar{S}} \\
&\quad + \int \big|f(x'_S, x_{\bar{S}})\big| \cdot \big|p(x_{\bar{S}} \mid x_S) - p(x_{\bar{S}} \mid x'_S)\big|dx_{\bar{S}} \\
&\leq L \cdot \|x_S - x'_S\|_2 \cdot \int p(x_{\bar{S}} \mid x_S)dx_{\bar{S}} + B \cdot \int \big|p(x_{\bar{S}} \mid x_S) - p(x_{\bar{S}} \mid x'_S)\big|dx_{\bar{S}} \\
&= L \cdot \|x_S - x'_S\|_2 + 2B \cdot d_{TV}\Big(p(x_{\bar{S}} \mid x_S), p(x_{\bar{S}} \mid x'_S)\Big).
\end{aligned}$$

□

Example B.1. *For a Gaussian random variable $\boldsymbol{x} \sim \mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d\times d}$, Assumption 4.3 holds with $M = \frac{1}{2}\sqrt{\lambda_{\max}(\Sigma^{-1}) - \lambda_{\min}(\Sigma^{-1})}$. If $\boldsymbol{x}$ is assumed to be standardized, this captures the case where independent features yield $M = 0$. Intuitively, it also means that if one dimension is roughly a linear combination of the others, or if $\lambda_{\max}(\Sigma^{-1})$ is large, the conditional distribution can change quickly as a function of the conditioning variables.*

Proof. Consider a Gaussian random variable $\boldsymbol{x}$ with mean $\mu \in \mathbb{R}^d$ and positive definite covariance $\Sigma \in \mathbb{R}^{d\times d}$. We can partition the variable into two parts, denoted $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and

then partition the variable and parameters as follows:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} \qquad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}.$$

We can then consider the distribution for $\boldsymbol{x}_1$ conditioned on a specific x_2 value. The conditional distribution is $\boldsymbol{x}_1 \mid x_2 \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma})$ with the parameters defined as

$$\bar{\mu} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \qquad \bar{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}.$$

Our goal is to understand the total variation distance when conditioning on different values x_2, x_2'. Pinsker's inequality gives us the following upper bound based on the KL divergence:

$$d_{TV}\Big(p(\boldsymbol{x}_1 \mid x_2), p(\boldsymbol{x}_1 \mid x_2')\Big) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}\Big(p(\boldsymbol{x}_1 \mid x_2) \mid\mid p(\boldsymbol{x}_1 \mid x_2')\Big)}.$$

The KL divergence between Gaussian distributions has a closed-form solution, and in the case with equal covariance matrices reduces to the following:

$$D_{\mathrm{KL}}\Big(p(\boldsymbol{x}_1 \mid x_2) \mid\mid p(\boldsymbol{x}_1 \mid x_2')\Big) = \frac{1}{2}(x_2 - x_2')^\top \Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1}(x_2 - x_2')$$

To provide an upper bound on this term based on $\|x_2 - x_2'\|_2$, it suffices to find the maximum eigenvalue of the above matrix. To characterize the matrix, we observe that it is present in the formula for the block inversion of the joint covariance matrix:

$$\Sigma^{-1} = \begin{pmatrix} \left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1} & -\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1} \\ -\Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1} & \Sigma_{22}^{-1} + \Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1} \end{pmatrix}$$

We therefore have the following, because the matrix we're interested in is a principal minor:

$$\lambda_{\max}\Big(\Sigma_{22}^{-1} + \Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Big) \leq \lambda_{\max}(\Sigma^{-1}).$$

Weyl's inequality also gives us:

$$\begin{aligned}&\lambda_{\min}(\Sigma_{22}^{-1}) + \lambda_{\max}\Big(\Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Big)\\ \leq\ &\lambda_{\max}\Big(\Sigma_{22}^{-1} + \Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Big)\end{aligned}$$

Combining this with the previous inequality, we arrive at the following:

$$\lambda_{\max}\Big(\Sigma_{22}^{-1}\Sigma_{21}\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Big) \leq \lambda_{\max}(\Sigma^{-1}) - \lambda_{\min}(\Sigma_{22}^{-1})$$

For the subtractive term, we can write:

$$\lambda_{\min}(\Sigma_{22}^{-1}) = \frac{1}{\lambda_{\max}(\Sigma_{22})} \geq \frac{1}{\lambda_{\max}(\Sigma)} = \lambda_{\min}(\Sigma^{-1}).$$

We therefore have the following upper bound on the KL divergence:

$$D_{\mathrm{KL}}\Big(p(\boldsymbol{x}_1 \mid x_2) \mid\mid p(\boldsymbol{x}_1 \mid x_2')\Big) \leq \frac{1}{2}\cdot\|x_2 - x_2'\|_2^2 \cdot \Big(\lambda_{\max}(\Sigma^{-1}) - \lambda_{\min}(\Sigma^{-1})\Big).$$

Combining this with Pinsker's inequality, we arrive at the final result:

$$d_{TV}\big(p(\boldsymbol{x}_1 \mid x_2), p(\boldsymbol{x}_1 \mid x_2')\big)\big) \leq \frac{1}{2}\cdot\|x_2 - x_2'\|_2 \cdot \sqrt{\lambda_{\max}(\Sigma^{-1}) - \lambda_{\min}(\Sigma^{-1})}.$$

This means that we have $M = \frac{1}{2}\sqrt{\lambda_{\max}(\Sigma^{-1}) - \lambda_{\min}(\Sigma^{-1})}$.

Notice that this does not imply $M = 0$ when the features $\boldsymbol{x}$ are independent: it only captures this for the case of isotropic variance. However, if we assume that $\boldsymbol{x}$ is standardized, so that each dimension has variance equal to 1, $M = 0$ is implied for independent features because we have $\Sigma = I$.

□

Lemma B.3. *Under Assumption 4.3, the prediction $f(\boldsymbol{x}_S)$ defined using the **conditional** approach is Lipschitz continuous with constant $L + 2BM$ for any feature set $\boldsymbol{x}_S$.*

Proof. The result follows directly from Lemma 4.2 and Assumption 4.3:

$$\begin{aligned}|f(x_S) - f(x'_S)| &\leq L \cdot \|x_S - x'_S\|_2 + 2B \cdot d_{TV}\Big(p(x_{\bar{S}} \mid x_S), p(x_{\bar{S}} \mid x'_S)\Big) \\ &\leq L \cdot \|x_S - x'_S\|_2 + 2BM \cdot \|x_S - x'_S\|_2 \\ &= (L + 2BM) \cdot \|x_S - x'_S\|_2.\end{aligned}$$

□

B.2 Proofs: Prediction Robustness to Model Perturbations

We now re-state and prove our results from Section 4.4 related to model perturbations.

Lemma B.4. *For two models $f, f' : \mathbb{R}^d \mapsto \mathbb{R}$ and a subdomain $\mathcal{X}' \subseteq \mathbb{R}^d$, the prediction functions $f(\boldsymbol{x}_S), f'(\boldsymbol{x}_S)$ for any feature set x_S satisfy*

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|_{\infty,\mathcal{X}'} \cdot Q_{x_S}(\mathcal{X}') + 2B \cdot \big(1 - Q_{x_S}(\mathcal{X}')\big),$$

where $Q_{x_S}(\mathcal{X}')$ is the probability of imputed samples lying in $\mathcal{X}'$ based on the distribution $q(\boldsymbol{x}_{\bar{S}})$:

$$Q_{x_S}(\mathcal{X}') \equiv \mathop{\mathbb{E}}_{q(\boldsymbol{x}_{\bar{S}})}\left[\mathbb{I}\left\{(x_S, \boldsymbol{x}_{\bar{S}}) \in \mathcal{X}'\right\}\right].$$

Proof. The main idea here is that when integrating over the difference in function outputs, we can separate the integral over $(x_S, \boldsymbol{x}_{\bar{S}})$ values falling in the domains $\mathcal{X}'$ and $\mathbb{R}^d \setminus \mathcal{X}'$. Because we have a single value for the observed variables x_S, the distribution $q(\boldsymbol{x}_{\bar{S}})$ is identical for

both terms regardless of the removal technique.

$$
\begin{aligned}
|f(x_S) - f'(x_S)| &= \Big| \int f(x_S, x_{\bar{S}}) q(x_{\bar{S}}) dx_{\bar{S}} - \int f'(x_S, x_{\bar{S}}) q(x_{\bar{S}}) dx_{\bar{S}} \Big| \\
&\leq \int \big| f(x_S, x_{\bar{S}}) - f'(x_S, x_{\bar{S}}) \big| \cdot q(x_{\bar{S}}) dx_{\bar{S}} \\
&= \int \big| f(x_S, x_{\bar{S}}) - f'(x_S, x_{\bar{S}}) \big| \cdot \mathbb{I}\left\{(x_S, \boldsymbol{x}_{\bar{S}}) \in \mathcal{X}'\right\} \cdot q(x_{\bar{S}}) dx_{\bar{S}} \\
&\quad + \int \big| f(x_S, x_{\bar{S}}) - f'(x_S, x_{\bar{S}}) \big| \cdot \mathbb{I}\left\{(x_S, \boldsymbol{x}_{\bar{S}}) \notin \mathcal{X}'\right\} \cdot q(x_{\bar{S}}) dx_{\bar{S}} \\
&\leq \|f - f'\|_{\infty, \mathcal{X}'} \int \mathbb{I}\left\{(x_S, \boldsymbol{x}_{\bar{S}}) \in \mathcal{X}'\right\} \cdot q(x_{\bar{S}}) dx_{\bar{S}} \\
&\quad + 2B \cdot \int \mathbb{I}\left\{(x_S, \boldsymbol{x}_{\bar{S}}) \notin \mathcal{X}'\right\} \cdot q(x_{\bar{S}}) dx_{\bar{S}} \\
&= \|f - f'\|_{\infty, \mathcal{X}'} \cdot Q_{x_S}(\mathcal{X}') + 2B \cdot \big(1 - Q_{x_S}(\mathcal{X}')\big)
\end{aligned}
$$

□

Lemma B.5. *When removing features using the* ***conditional*** *approach, the prediction functions $f(\boldsymbol{x}_S), f'(\boldsymbol{x}_S)$ for two models f, f' and any feature set x_S satisfy*

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|_{\infty, \mathcal{X}}.$$

Proof. The result follows directly from Lemma 4.4 when we set $\mathcal{X}' = \mathcal{X}$. Note that when we remove features using the conditional distribution, we have $Q_{x_S}(\mathcal{X}) = 1$ for all x_S lying on the data manifold, or where there exists $x_{\bar{S}}$ such that $p(x_S, x_{\bar{S}}) > 0$. □

Lemma B.6. *When removing features using the* ***baseline*** *or* ***marginal*** *approach, the prediction functions $f(\boldsymbol{x}_S), f'(\boldsymbol{x}_S)$ for two models f, f' and any feature set x_S satisfy*

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|_{\infty}.$$

Proof. The result follows directly from Lemma 4.4 when we set $\mathcal{X}' = \mathbb{R}^d$, because we have $Q_{x_S}(\mathbb{R}^d) = 1$ for all $x_S \in \mathbb{R}^{|S|}$. □

B.3 Proofs: Summary Technique Robustness

We now re-state and prove our results from Section 4.4 related to summary techniques.

Proposition B.1. *The attributions for each method can be calculated by applying a linear operator $A \in \mathbb{R}^{d \times 2^d}$ to a vector $v \in \mathbb{R}^{2^d}$ representing the predictions with each feature set, or*

$$\phi(f, x) = Av,$$

where the linear operator A for each method is listed in Table 4.3, and v is defined as $v_S = f(x_S)$ for each $S \subseteq [d]$ based on the chosen feature removal technique.

Proof. This result follows directly from the definition of each method. Following the approach of Covert et al. (2021), we disentangle between each method's feature removal implementation and how it generates attribution scores. Assuming a fixed input x where $f(x_S)$ denotes the prediction with a feature set, each method's attributions are defined as follows.

- **Occlusion** (Zeiler and Fergus, 2014): this method simply compares the prediction given all features and the prediction with a single missing feature. The attribution score $\phi_i(f, x)$ is defined as

  $$\phi_i(f, x) = f(x) - f(x_{[d] \setminus i}).$$

  We refer to this summary technique as "leave-one-out" in Table 4.3.

- **Shapley value** (Shapley, 1953; Lundberg and Lee, 2017): this method calculates the impact of removing a single feature, and then averages this over all possible preceding subsets. The attribution score $\phi_i(f, x)$ is defined as

  $$\phi_i(f, x) = \frac{1}{d} \sum_{S \subseteq [d] \setminus i} \binom{d-1}{|S|}^{-1} \Big(f(x_{S \cup i}) - f(x_S) \Big).$$

- **Banzhaf value** (Banzhaf, 1964; Chen and Jordan, 2020): this method is similar to the Shapley value, but it applies a uniform weighting when averaging over all preceding subsets. The attribution score $\phi_i(f, x)$ is defined as

$$\phi_i(f, x) = \frac{1}{2^{d-1}} \sum_{S \subseteq [d] \setminus i} \Big(f(x_{S \cup i}) - f(x_S) \Big).$$

- **RISE** (Petsiuk et al., 2018): this method is similar to the Banzhaf value, but it does not subtract the value for subsets where a feature is not included. The attribution score $\phi_i(f, x)$ is defined as

$$\phi_i(f, x) = \frac{1}{2^{d-1}} \sum_{S \subseteq [d] \setminus i} f(x_{S \cup i}).$$

  We refer to this summary technique as "mean when included" in Table 4.3

- **LIME** (Ribeiro et al., 2016): this method is more flexible than the others and generally involves solving a regression problem that treats the features as inputs and the predictions as labels. LIME can be viewed as a removal-based explanation when the features are represented in a binary fashion, indicating whether they are provided to the model or not (Covert et al., 2021). There is also a choice of weighting kernel $w(S) \geq 0$, of a regularization term, and of whether to fit an intercept term.

  We assume that the regularization is zero and that we fit an intercept term. Denoting the model's coefficients as $(\beta_0, \ldots, \beta_d)$, the problem is the following:

$$\min_{\beta_0, \ldots, \beta_d} \sum_{S \subseteq [d]} w(S) \Big(\beta_0 + \sum_{i \in S} \beta_i - f(x_S) \Big)^2.$$

  We can rewrite this problem in a simpler fashion to derive its solution. First, we let $W \in [0, 1]^{2^d \times 2^d}$ be a diagonal matrix containing the weights $w(S)$ for each subset. Next, we let the predictions be represented by a vector $v \in \mathbb{R}^{2^d}$ where $v_S = f(x_S)$, and

we let the matrix $Z \in \{0,1\}^{2^d \times d}$ enumerate all subsets in a binary fashion. Finally, to accommodate the intercept term, we let $\beta = (\beta_0, \ldots, \beta_d) \in \mathbb{R}^{d+1}$ represent the parameters, and we prepend a columns of ones to Z, resulting in $Z' \in \{0,1\}^{2^d \times (d+1)}$. The problem then becomes:

$$\min_{\beta} (Z'\beta - v)^\top W (Z'\beta - v') = \left\| Z'\beta - v \right\|_W^2.$$

Assuming that $Z'^\top W Z'$ is invertible, which is guaranteed when $w(S) > 0$ for all $S \subseteq [d]$, this problem has a closed-form solution:

$$\beta^* = (Z'^\top W Z')^{-1} Z'^\top W v.$$

LIME's attribution scores are given by $\phi_i(f, x) = \beta_i^*$, for $i = 1, ..., d$, where we discard the intercept term. We can therefore conclude that each attribution is a linear function of v, with coefficients given by all but the top row of $(Z'^\top W Z')^{-1} Z'^\top W$. If we denote all but the first row of $(Z'^\top W Z')^{-1}$ as $(Z'^\top W Z')^{-1}_{[1:]}$, then we have $A = (Z'^\top W Z')^{-1}_{[1:]} Z'^\top W$.

The formula for the attributions remains roughly the same if we do not fit an intercept term: if the intercept is zero, we have $A = (Z^\top W Z)^{-1} Z^\top W$. We can similarly allow for ridge regularization: given a penalty with parameter $\lambda > 0$, we then have $A = (Z^\top W Z + \lambda I_{2^d})^{-1} Z^\top W$.

These coefficients depend on the specific choice of weighting kernel $w(S)$, and previous work has shown that leave-one-out, Shapley and Banzhaf values all correspond to specific weighting kernels (Covert et al., 2021). Our analysis of those methods therefore pertains to LIME in a limited sense. However, LIME often uses a heuristic exponential kernel in practice, which is more difficult to characterize analytically. We therefore omit the specific entries and norms for LIME in Table 4.3, but show norms for the exponential kernel in our computed results.

We refer to this summary technique as "weighted least squares" in Table 4.3.

We can therefore see that in each method, $\phi_i(f, x)$ is a linear function of the predictions for each $S \subseteq [d]$. Across all methods, we can let the predictions be represented by a vector $v \in \mathbb{R}^{2^d}$ and the coefficients by a matrix $A \in \mathbb{R}^{d \times 2^d}$. □

Lemma B.7. *The difference in attributions given the same summary technique A and different model outputs v, v' can be bounded as*

$$\|Av - Av'\|_2 \leq \|A\|_2 \cdot \|v - v'\|_2 \qquad or \qquad \|Av - Av'\|_2 \leq \|A\|_{1,\infty} \cdot \|v - v'\|_\infty,$$

where $\|A\|_2$ is the spectral norm, and the operator norm $\|A\|_{1,\infty}$ is the square root of the sum of squared row 1-norms, with values for each A given in Table 4.3.

Proof. We first derive the two matrix inequalities. The first inequality follows from the definition of the spectral norm, which is defined as the maximum singular value:

$$\|A\|_2 \equiv \sup_{u \neq 0} \frac{\|Au\|_2}{\|u\|_2} = \sigma_{\max}(A) = \sqrt{\lambda_{\max}(AA^\top)}.$$

We prove the second bound using Holder's inequality as follows, where A_i denotes the ith row:

$$\|Au\|_2 = \sqrt{\sum_{i=1}^{d} (A_i^\top u)^2} \leq \sqrt{\sum_{i=1}^{d} \|A_i\|_1^2 \cdot \|u\|_\infty^2} = \|u\|_\infty \cdot \sqrt{\sum_{i=1}^{d} \|A_i\|_1^2} = \|u\|_\infty \cdot \|A\|_{1,\infty}.$$

In the special case where all of A's row 1-norms are equal, or where we have $\|A_i\|_1 = \|A_j\|_1$ for all $i, j \in [d]$, we can use the simplified expression $\|A\|_{1,\infty} = \|A_1\|_1 \cdot \sqrt{d}$.

Next, we derive the specific norm values for each linear operator in Table 4.3.

Beginning with the operator norm $\|A\|_{1,\infty}$, we remark that several methods have equal 1-norms across all rows: following their interpretation as *semivalues* (Monderer et al., 2002), leave-one-out, Shapley and Banzhaf all have $\|A_i\|_1 = 2$ for all $i \in [d]$. Similarly, RISE has $\|A_i\|_1 = 1$ for all $i \in [d]$. This yields the norms $\|A\|_{1,\infty} = 2\sqrt{d}$ for leave-one-out, Banzhaf and Shapley, and $\|A\|_{1,\infty} = \sqrt{d}$ for RISE.

For the spectral norm, we consider each method separately. We find that for all methods, it is simplest to calculate $AA^\top$ and then $\lambda_{\max}(AA^\top)$ to find the maximum singular value $\sigma_{\max}(A)$. Note that the entries of the matrix are determined by the inner product $(AA^\top)_{ij} = A_i^\top A_j$.

For leave-one-out, we derive $AA^\top$ as follows:

$$(AA^\top)_{ij} = \begin{cases} 2 & \text{if } i = j \\ 1 & \text{otherwise.} \end{cases}$$

The matrix $AA^\top$ is therefore the sum of a rank-one component and an identity term, or we have $AA^\top = \mathbf{1}_d\mathbf{1}_d + I_d$. Recognizing that $\mathbf{1}_d$ is the leading eigenvector, we get the following result:

$$\sigma_{\max}(A) = \sqrt{\lambda_{\max}(AA^\top)} = \sqrt{\frac{(\mathbf{1}_d\mathbf{1}_d + I_d)\mathbf{1}_d}{\|\mathbf{1}_d\|_2}} = \sqrt{\frac{\mathbf{1}_d d + \mathbf{1}_d}{\|\mathbf{1}_d\|_2}} = \sqrt{d+1}.$$

For the Banzhaf value, we derive $AA^\top$ as follows:

$$(AA^\top)_{ij} = \begin{cases} 1/2^{2d-2} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

The max eigenvalue is therefore $\lambda_{\max}(AA^\top) = 1/2^{d-2}$, which yields $\|A\|_2 = 1/2^{d/2-1}$.

For the mean when included technique used by RISE, we derive $AA^\top$ as follows:

$$(AA^\top)_{ij} = \begin{cases} 1/2^{d-1} & \text{if } i = j \\ 1/2^d & \text{otherwise.} \end{cases}$$

The matrix $AA^\top$ is therefore the sum of a rank-one component and an identity term, or we have $AA^\top = \mathbf{1}_d\mathbf{1}_d/2^d + I_d/2^d$. Recognizing that $\mathbf{1}_d$ is the leading eigenvector, we get the

following result:

$$\sigma_{\max}(A) = \sqrt{\lambda_{\max}(AA^\top)} = \sqrt{\frac{(\mathbf{1}_d\mathbf{1}_d/2^d + I_d/2^d)\mathbf{1}_d}{\|\mathbf{1}_d\|_2}} = \sqrt{\frac{\mathbf{1}_d d/2^d + \mathbf{1}_d/2^d}{\|\mathbf{1}_d\|_2}} = \sqrt{\frac{d+1}{2^d}}.$$

For the Shapley value, we derive $AA^\top$ by first considering the diagonal entries:

$$\begin{aligned}
(AA^\top)_{ii} = A_i^\top A_i &= \sum_{S:i\in S}\left(\frac{(|S|-1)!(d-|S|)!}{d!}\right)^2 + \sum_{S:i\notin S}\left(\frac{|S|!(d-|S|-1)!}{d!}\right)^2 \\
&= \sum_{k=0}^{d}\left[\binom{d-1}{k-1}\left(\frac{(k-1)!(d-k)!}{d!}\right)^2 + \binom{d-1}{k}\left(\frac{k!(d-k-1)!}{d!}\right)^2\right] \\
&= \frac{1}{d}\sum_{k=1}^{d}\frac{(k-1)!(d-k)!}{d!} + \frac{1}{d}\sum_{k=0}^{d-1}\frac{k!(d-k-1)!}{d!} \\
&= \sum_{k=1}^{d-1}\frac{(k-1)!(d-k-1)!}{d!} + \underbrace{\frac{2}{d^2}}_{k=0,k=d} \\
&= \frac{1}{d(d-1)}\sum_{k=1}^{d-1}\binom{d-2}{k-1}^{-1} + \frac{2}{d^2}
\end{aligned}$$

We next consider the off-diagonal entries:

$$
\begin{aligned}
(AA^\top)_{ij} = A_i^\top A_j = &\sum_{S : i \in S, j \in S} \frac{(|S|-1)!(d-|S|)!}{d!} \frac{(|S|-1)!(d-|S|)!}{d!} \\
&- \sum_{S : i \in S, j \notin S} \frac{(|S|-1)!(d-|S|)!}{d!} \frac{|S|!(d-|S|-1)!}{d!} \\
&- \sum_{S : i \notin S, j \in S} \frac{|S|!(d-|S|-1)!}{d!} \frac{(|S|-1)!(d-|S|)!}{d!} \\
&+ \sum_{S : i \notin S, j \notin S} \frac{|S|!(d-|S|-1)!}{d!} \frac{|S|!(d-|S|-1)!}{d!} \\
= &\sum_{k=0}^{d} \binom{d-2}{k-2} \frac{(k-1)!(d-k)!}{d!} \frac{(k-1)!(d-k)!}{d!} \\
&- 2 \sum_{k=0}^{d} \binom{d-2}{k-1} \frac{(k-1)!(d-k)!}{d!} \frac{k!(d-k-1)!}{d!} \\
&+ \sum_{k=0}^{d} \binom{d-2}{k} \frac{k!(d-k-1)!}{d!} \frac{k!(d-k-1)!}{d!} \\
= &\sum_{k=2}^{d} \frac{(k-1)}{d(d-1)} \frac{(k-1)!(d-k)!}{d!} \\
&- 2 \sum_{k=1}^{d-1} \frac{1}{d(d-1)} \frac{k!(d-k)!}{d!} \\
&+ \sum_{k=0}^{d-2} \frac{(d-k-1)}{d(d-1)} \frac{k!(d-k-1)!}{d!}
\end{aligned}
$$

We first focus on the case where $d > 2$, and we later return to the case where $d = 2$. We can

group terms that share the same k values to simplify the above:

$$\begin{aligned} A_i^\top A_j &= \frac{1}{d(d-1)}\left[-\sum_{k=2}^{d-2}\frac{(k-1)!(d-k-1)!}{(d-1)!} + \underbrace{\frac{2(d-1)}{d}}_{k=0,k=d} - \underbrace{\frac{2}{d-1}}_{k=1,k=d-1}\right] \\ &= -\frac{1}{d(d-1)}\sum_{k=2}^{d-2}\frac{(k-1)!(d-k-1)!}{(d-1)!} + \frac{2}{d^2} - \frac{2}{d(d-1)^2} \\ &= -\frac{1}{d(d-1)^2}\sum_{k=2}^{d-2}\binom{d-2}{k-1}^{-1} + \frac{2}{d^2} - \frac{2}{d(d-1)^2}. \end{aligned}$$

By convention, we let the summation above evaluate to zero for $d = 3$. We therefore have the following result for $AA^\top$:

$$(AA^\top)_{ij} = \begin{cases} \frac{1}{d(d-1)}\sum_{k=1}^{d-1}\binom{d-2}{k-1}^{-1} + \frac{2}{d^2} & \text{if } i = j \\ -\frac{1}{d(d-1)^2}\sum_{k=2}^{d-2}\binom{d-2}{k-1}^{-1} + \frac{2}{d^2} - \frac{2}{d(d-1)^2} & \text{otherwise.} \end{cases}$$

From this, we can see that $AA^\top$ is the sum of a rank-one component and an identity term, or that we have $AA^\top = \mathbf{1}_d\mathbf{1}_d b + I_d(c-b)$ with values for $b = (AA^\top)_{ij}$ and $c = (AA^\top)_{ii}$ given above. This matrix has two eigenvalues, $c - b$ and $db + (c - b)$, and we must consider which of the two is larger. Under our assumption of $d > 2$, we can observe that $b > 0$:

$$\begin{aligned} b = (AA^\top)_{ij} &= \frac{2}{d^2} - \frac{1}{d(d-1)^2}\sum_{k=2}^{d-2}\binom{d-2}{k-1}^{-1} - \frac{2}{d(d-1)^2} \\ &\geq \frac{2}{d^2} - \frac{1}{d(d-1)^2}\sum_{k=2}^{d-2}\binom{d-2}{1}^{-1} - \frac{2}{d(d-1)^2} \\ &= \frac{2}{d^2} - \frac{d-3}{d(d-1)^2(d-2)} - \frac{2}{d(d-1)^2} \\ &= \frac{2(d-1)^2(d-2) - d(d-3) - 2d(d-2)}{d^2(d-1)^2(d-2)} \end{aligned}$$

It can be verified that the numerator of the above fraction is positive for $d > 2$. This implies

that $db + (c - b)$ is the larger eigenvalue in this case. We can therefore calculate $\lambda_{\max}(AA^\top)$ as follows:

$$\begin{aligned}\lambda_{\max}(AA^\top) &= db + (c-b) = c + (d-1)b \\ &= \frac{2}{d^2} + \underbrace{\frac{2}{d(d-1)}}_{k=1,k=d-1} + \frac{2(d-1)}{d^2} - \frac{2}{d(d-1)} \\ &= \frac{2(d-1) + 2(d-1)^2}{d^2(d-1)} \\ &= \frac{2}{d}\end{aligned}$$

On the other hand, when we have $d = 2$ we have the following entries for $AA^\top$:

$$(AA^\top)_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

This yields $\lambda_{\max}(AA^\top) = 1$, which coincides with the general formula $\lambda_{\max}(AA^\top) = 2/d$ derived above. Finally, this let us conclude that the spectral norm for the Shapley value is the following:

$$\|A\|_2 = \sqrt{\lambda_{\max}(AA^\top)} = \sqrt{\frac{2}{d}}.$$

□

Lemma B.8. *When the linear operator A satisfies the **boundedness** property, we have $\|A\|_{1,\infty} = 2\sqrt{d}$.*

Proof. Following Theorem 1 in Monderer et al. (2002), solution concepts satisfying linearity and the boundedness property (also known as the Milnor axiom) are *probabilistic values*. This means that each attribution is the average of a feature's marginal contributions, or that for each feature $i \in [d]$ there exists a distribution $p_i(S)$ with support on the power set of

$[d] \setminus i$, such that the attributions are given by

$$\phi_i(f, x) = \mathop{\mathbb{E}}_{p_i(S)}\big[f(x_{S\cup i}) - f(x_S)\big] = \sum_{S\subseteq[d]\setminus i} p_i(S)\big(f(x_{S\cup i}) - f(x_S)\big). \tag{B.1}$$

The corresponding matrix $A \in \mathbb{R}^{d\times 2^d}$ therefore has rows A_i with entries $A_{iS} \geq 0$ when $i \in S$ and $A_{iS} \leq 0$ when $i \notin S$, where the positive and negative entries each sum to 1. We can therefore conclude that each row satisfies $\|A_i\|_1 = 2$, and that $\|A\|_{1,\infty} = 2\sqrt{d}$.

□

Lemma B.9. *When the linear operator A satisfies the **boundedness** and **symmetry** properties, the spectral norm is bounded as follows,*

$$\frac{1}{2^{d/2-1}} \leq \|A\|_2 \leq \sqrt{d+1},$$

with $1/2^{d/2-1}$ achieved by the Banzhaf value and $\sqrt{d+1}$ achieved by leave-one-out.

Proof. Following Monderer et al. (2002), solution concepts satisfying linearity, boundedness and symmetry are *semivalues*. This means that they are *probabilistic values* for which the probability distribution $p_i(S)$ depends only on the cardinality $|S|$ (see the proof above). As a result, we can parameterize the summary technique by how much weight it places on each cardinality.

Let $\alpha \in \mathbb{R}^d$ denote a probability distribution over the cardinalities $k = 1, \ldots, d$. We can write the attributions generated by any semivalue as follows:

$$\phi_i(f, x) = \sum_{k=1}^{d} \alpha_k \sum_{\substack{S\subseteq[d]\setminus i \\ |S|=k-1}} \binom{d-1}{k-1}^{-1} \Big(f(x_{S\cup i}) - f(x_S)\Big).$$

Note that this resembles eq. B.1, only with coefficients that are shared between all subsets with the same cardinality. Due to each attribution being a linear combination of the predictions, we can associate a matrix $A \in \mathbb{R}^{d\times 2^d}$ with each semivalue, similar to Proposition 4.1.

Furthermore, we can let A be a linear combination of basis elements corresponding to each cardinality. We define the basis matrix $A^{(k)} \in \mathbb{R}^{d \times 2^d}$ for each cardinality $k = 1, \ldots, d$ as follows:

$$A_{iS}^{(k)} = \begin{cases} \binom{d-1}{|S|-1}^{-1} & \text{if } i \in S \text{ and } |S| = k \\ -\binom{d-1}{|S|}^{-1} & \text{if } i \notin S \text{ and } |S| = k-1 \\ 0 & \text{otherwise.} \end{cases}$$

With this, we can write the matrix associated with any semivalue as a function of the cardinality weights α:

$$A(\alpha) = \sum_{k=1}^{d} \alpha_k A^{(k)}.$$

An important insight is that by composing a linear function with the convex spectral norm function (Boyd et al., 2004), we can see that $\|A(\alpha)\|_2$ is a convex function of α. This lets us find both the upper and lower bound of $\|A(\alpha)\|_2$ subject to α being a valid probability distribution.

For the lower bound, we refer readers to Theorem C.3 from Wang and Jia (2022), who prove that the Banzhaf value achieves the minimum with α^* such that $\alpha_k^* = \binom{d-1}{k-1}/2^{d-1}$ and spectral norm $\|A(\alpha^*)\|_2 = 1/2^{d/2-1}$. Notice that this is a valid probability distribution because we have $\sum_{k=1}^{d} \binom{d-1}{k-1} = 2^{d-1}$. Also note that the proof in (Wang and Jia, 2022) parameterizes α slightly differently.

For the upper bound, we can leverage convexity to make the following argument. The fact that spectral norm is convex with α representing probabilities lets us write the following for any α value:

$$\sigma_{\max}\Big(A(\alpha)\Big) = \sigma_{\max}\left(\sum_{k=1}^{d} \alpha_k \cdot A^{(k)}\right) \leq \sum_{k=1}^{d} \alpha_k \cdot \sigma_{\max}\big(A^{(k)}\big) \leq \max_k \Big\{\sigma_{\max}\big(A^{(k)}\big)\Big\}.$$

Finding the upper bound therefore reduces to finding the basis element with the largest spectral norm. For each basis element $A^{(k)}$, we can derive the spectral norm by first finding the maximum eigenvalue of $A^{(k)}{A^{(k)}}^\top \in \mathbb{R}^{d\times d}$. In deriving this matrix, we first consider the diagonal entries:

$$\begin{aligned}\left(A^{(k)}{A^{(k)}}^\top\right)_{ii} &= \sum_{S:i\in S,|S|=k}\binom{d-1}{|S|-1}^{-2} + \sum_{S:i\notin S,|S|=k-1}\binom{d-1}{|S|}^{-2} \\ &= \binom{d-1}{k-1}\binom{d-1}{k-1}^{-2} + \binom{d-1}{k-1}\binom{d-1}{k-1}^{-2} \\ &= 2\binom{d-1}{k-1}^{-1}.\end{aligned}$$

Next, we consider the off-diagonal entries:

$$\begin{aligned}\left(A^{(k)}{A^{(k)}}^\top\right)_{ij} &= \sum_{S:i\in S,j\in S,|S|=k}\binom{d-1}{|S|-1}^{-2} + \sum_{S:i\notin S,j\notin S,|S|=k-1}\binom{d-1}{|S|}^{-2} \\ &= \binom{d-2}{k-2}\binom{d-1}{k-1}^{-2} + \binom{d-2}{k-1}\binom{d-1}{k-1}^{-2} \\ &= \frac{k-1}{d-1}\binom{d-1}{k-1}^{-1} + \frac{d-k}{d-1}\binom{d-1}{k-1}^{-1} \\ &= \binom{d-1}{k-1}^{-1}.\end{aligned}$$

To summarize, the entries of this matrix are the following:

$$\left(A^{(k)}{A^{(k)}}^\top\right)_{ij} = \begin{cases} 2\binom{d-1}{k-1}^{-1} & \text{if } i=j \\ \binom{d-1}{k-1}^{-1} & \text{otherwise.}\end{cases}$$

We therefore see that $A^{(k)}{A^{(k)}}^\top$ is equal to a rank-one matrix plus an identity matrix, or $A^{(k)}{A^{(k)}}^\top = b\mathbf{1}_d\mathbf{1}_d^\top + (c-b)I_d$, with values for $b = \left(A^{(k)}{A^{(k)}}^\top\right)_{ij}$ and $c = \left(A^{(k)}{A^{(k)}}^\top\right)_{ii}$ given above. Recognizing that $b > 0$ and therefore that $\mathbf{1}_d$ is the leading eigenvector, we have the

following expression for the maximum eigenvalue:

$$\begin{aligned}\lambda_{\max}\left(A^{(k)}A^{(k)\top}\right) &= c + (d-1)b \\ &= \left(A^{(k)}A^{(k)\top}\right)_{ii} + (d-1)\cdot\left(A^{(k)}A^{(k)\top}\right)_{ij} \\ &= (d+1)\binom{d-1}{k-1}^{-1}.\end{aligned}$$

Finally, we must consider how to maximize this as a function of k. Setting $k = 1$ or $k = d$ achieves the same value of $d + 1$, which yields the following spectral norm:

$$\|A^{(d)}\|_2 = \sqrt{\lambda_{\max}\left(A^{(d)}A^{(d)\top}\right)} = \sqrt{d+1}.$$

The case with $k = d$ corresponds to leave-one-out (see Lemma 4.7), and $k = 1$ corresponds to an analogous *leave-one-in* approach that has been discussed in prior work (Covert et al., 2020, 2021). □

Lemma B.10. *When the linear operator A satisfies the* ***boundedness*** *and* ***efficiency*** *properties, the spectral norm is bounded as follows,*

$$\sqrt{\frac{2}{d}} \leq \|A\|_2 \leq \sqrt{2 + 2\cdot\cos\left(\frac{\pi}{d+1}\right)},$$

with $\sqrt{2/d}$ achieved by the Shapley value.

Proof. Following Monderer et al. (2002), solution concepts satisfying linearity, boundedness and efficiency are *random-order values* (or *quasivalues*). This means that the attributions are the average marginal contributions across a distribution of feature orderings. Reasoning about such orderings requires new notation, so we use π to denote an ordering over $[d]$, and $\pi(i)$ to denote the position of index i within the ordering. We also let $\text{Pre}(\pi, i)$ denote the set of all elements preceding i in the ordering, or $\text{Pre}(\pi, i) \equiv \{j : \pi(j) < \pi(i)\}$. Similarly, we let $\text{Pre}(\pi, i, j)$ be an indicator of whether j directly precedes i in the ordering, or $\text{Pre}(\pi, i, j) \equiv$

$\mathbb{I}\{\pi(j) = \pi(i) - 1\}$. Finally, we use Π to denote the set of all orderings, where $|\Pi| = d!$.

Now, given a distribution over orderings $p(\pi)$, we can write the attributions for any random-order value as follows:

$$\begin{aligned}\phi_i(f, x) &= \sum_{\pi \in \Pi} p(\pi)\Big(f(x_{\mathrm{Pre}(\pi,i)\cup i}) - f(x_{\mathrm{Pre}(\pi,i)})\Big) \\ &= \sum_{S \subseteq [d]\setminus i} \Big(\sum_{\pi:\mathrm{Pre}(\pi,i)=S} p(\pi)\Big)\Big(f(x_{S\cup i}) - f(x_S)\Big).\end{aligned}$$

Based on this formulation, we can associate a matrix $A \in \mathbb{R}^{d\times 2^d}$ with each random-order value based on the coefficients applied to each prediction. The entries are defined as follows:

$$A_{iS} = \begin{cases} \sum_{\pi:\mathrm{Pre}(\pi,i)=S\setminus i} p(\pi) & \text{if } i \in S \\ -\sum_{\pi:\mathrm{Pre}(\pi,i)=S} p(\pi) & \text{otherwise.} \end{cases}$$

Furthermore, we can view A as a linear combination of basis elements $A^{(\pi)} \in \mathbb{R}^{d\times 2^d}$ corresponding to each individual ordering. We define the basis matrix $A^{(\pi)}$ associated with each $\pi \in \Pi$ as follows:

$$A^{(\pi)}_{iS} = \begin{cases} 1 & \text{if } \mathrm{Pre}(\pi, i) = S \setminus i \\ -1 & \text{if } \mathrm{Pre}(\pi, i) = S \\ 0 & \text{otherwise.} \end{cases}$$

With this, we can write the linear operator A corresponding to a distribution $p(\pi)$ as the following linear combination:

$$A(p) = \sum_{\pi \in \Pi} p(\pi) \cdot A^{(\pi)}.$$

By composing a linear operation with the convex spectral norm function (Boyd et al., 2004), we can see that $\|A(p)\|_2$ is a convex function of the probabilities $p(\pi)$ assigned to each

ordering. This will allow us to derive a form for both the upper and lower bound on $\|A(p)\|_2$.

Beginning with the upper bound, we can use convexity to write the following:

$$\sigma_{\max}\big(A(p)\big) = \sigma_{\max}\Big(\sum_{\pi\in\Pi} p(\pi)\cdot A^{(\pi)}\Big) \leq \sum_{\pi\in\Pi} p(\pi)\cdot\sigma_{\max}\big(A^{(\pi)}\big) \leq \max_{\pi\in\Pi}\Big\{\sigma_{\max}\big(A^{(\pi)}\big)\Big\}.$$

Finding the upper bound therefore reduces to finding the basis element with the largest spectral norm. For an arbitrary ordering π, we can derive the spectral norm via the matrix $A^{(\pi)}{A^{(\pi)}}^\top$, whose entries are given by:

$$\Big(A^{(\pi)}{A^{(\pi)}}^\top\Big)_{ij} = \begin{cases} 2 & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } \mathrm{Pre}(\pi, i, j) \\ -1 & \text{if } i \neq j \text{ and } \mathrm{Pre}(\pi, j, i) \\ 0 & \text{otherwise.} \end{cases}$$

Within these matrices, the diagonal entries are always equal to 2, most rows contain two -1's, and two rows (corresponding to the first and last element in the ordering π) contain a single -1. We can see that the matrices $A^{(\pi)}{A^{(\pi)}}^\top$ are identical up to a permutation of the rows and columns, or that they are similar matrices. They therefore share the same eigenvalues.

Without loss of generality, we choose to focus on the ordering π' that places the indices in their original order, or $\pi'(i) = i$ for $i \in [d]$. The corresponding matrix $A^{(\pi')}{A^{(\pi')}}^\top$ has the

following form:

$$A^{(\pi')}A^{(\pi')\top} = \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix}$$

This is a tridiagonal Toeplitz matrix, a type of matrix whose eigenvalues have a known closed-form solution (Noschese et al., 2013). The maximum eigenvalue is the following:

$$\begin{aligned} \lambda_{\max}\left(A^{(\pi')}A^{(\pi')\top}\right) &= 2 + 2\sqrt{(-1)\cdot(-1)}\cos\left(\frac{\pi}{d+1}\right) \\ &= 2 + 2\cdot\cos\left(\frac{\pi}{d+1}\right). \end{aligned}$$

We therefore have the following upper bound on the spectral norm for a random-order value:

$$\|A\|_2 \leq \sqrt{2 + 2\cdot\cos\left(\frac{\pi}{d+1}\right)}.$$

This value tends asymptotically towards 2, and it is achieved by any random-order value that places all its probability mass on a single ordering. Such a *single-order value* approach has not been discussed in the literature, but it is a special case of a proposed variant of the Shapley value (Frye et al., 2020).

Next, we consider the lower bound. We now exploit the convexity of $\|A(p)\|_2$ to argue that a specific distribution achieves a lower spectral norm than any other distribution.

Consider an arbitrary distribution $p(\pi)$ over orderings, with the associated linear operator $A(p)$ and spectral norm $\|A(p)\|_2$. Our proof technique is to find other distributions that result in different linear operators A that share the same spectral norm. For any ordering $\pi' \in \Pi$,

we can imagine permuting the distribution $p(\pi)$ in the following sense: we generate a new distribution $p_{\pi'}(\pi)$ such that $p_{\pi'}(\pi) = p(\pi(\pi'))$, where $\pi(\pi')$ is a modified ordering with $\pi(\pi')(i) = \pi(\pi'(i))$. That is, the probability mass is reassigned based on a permutation of the indices.

Crucially, the linear operator $A(p_{\pi'})$ is different from $A(p)$ but shares the same spectral norm. This is because $A(p_{\pi'})A(p_{\pi'})^\top$ is similar to $A(p)A(p)^\top$, or they are equal up to a permutation of the rows and columns. We therefore have $\|A(p_{\pi'})\|_2 = \|A(p)\|_2$, and due to convexity we have

$$\big\|A\big(p/2 + p_{\pi'}/2\big)\big\|_2 \leq \frac{1}{2}\|A(p_{\pi'})\|_2 + \frac{1}{2}\|A(p)\|_2 = \|A(p)\|_2.$$

This does not imply that $p/2 + p_{\pi'}/2$ is a minimizer, but we can now generalize this technique to identify a lower bound. Rather than generating a single permuted distribution, we propose generating all possible permuted distributions. That is, we consider the set of all $d!$ permutations $\pi' \in \Pi$ and generate the corresponding distributions $\{p_{\pi'} : \pi' \in \Pi\}$. Notice that when we take the mean of these distributions, the result is identical regardless of the original distributions $p(\pi)$, and that it assigns uniform probability mass to all orderings, because we have:

$$\frac{1}{d!}\sum_{\pi' \in \Pi} p_{\pi'}(\pi) = \frac{1}{d!}\sum_{\pi' \in \Pi} p(\pi') = \frac{1}{d!}.$$

We can then exploit convexity to write the following inequality:

$$\Big\|A\big(\sum_{\pi' \in \Pi} p_{\pi'}/d!\big)\Big\|_2 \leq \|A(p)\|_2.$$

We can make this argument for any original distribution p, so this provides a global lower bound. The random-order value with uniform weighting is the Shapley value, whose spectral norm was derived in Lemma 4.7. We therefore conclude with the following lower bound for

random-order values, which is achieved by the Shapley value:

$$\|A\|_2 \geq \sqrt{\frac{2}{d}}.$$

□

Lemma B.11. *When the linear operator A satisfies the* ***boundedness***, ***symmetry*** *and* ***efficiency*** *properties, the spectral norm is $\|A\|_2 = \sqrt{2/d}$.*

Proof. The result follows from the spectral norm result in Lemma 4.7, and the fact that the Shapley value is the only solution concept to satisfy linearity, boundedness, symmetry and efficiency (Monderer et al., 2002). □

B.4 Proofs: Main Results

We now re-state and prove our results from Section 4.5.

Theorem B.1. *The robustness of removal-based explanations to input perturbations is given by the following meta-formula,*

$$\|\phi(f,x) - \phi(f,x')\|_2 \leq g(\text{removal}) \cdot h(\text{summary}) \cdot \|x - x'\|_2,$$

where the factors for each method are defined as follows:

$$g(\text{removal}) = \begin{cases} L & \text{if removal} = \textbf{\textit{baseline}} \text{ or } \textbf{\textit{marginal}} \\ L + 2BM & \text{if removal} = \textbf{\textit{conditional}}, \end{cases}$$

$$h(\text{summary}) = \begin{cases} 2\sqrt{d} & \text{if summary} = \textbf{\textit{Shapley}}, \textbf{\textit{Banzhaf}}, \text{ or } \textbf{\textit{leave-one-out}} \\ \sqrt{d} & \text{if summary} = \textbf{\textit{mean when included}}. \end{cases}$$

Proof. Given two inputs x and x', the results in Lemma 4.1 and Lemma 4.3 imply the

following bound for any feature set $\boldsymbol{x}_S$,

$$|f(x'_S) - f(x'_S)| \leq L' \cdot \|x_S - x'_S\|_2,$$

where we have $L' = L$ when using the baseline or marginal approach and $L' = L + 2BM$ when using the conditional approach. This bound depends on the specific subset, we also have a bound across all subsets because $\|x_S - x'_S\|_2 \leq \|x - x'\|_2$ for all $S \subseteq [d]$. In terms of the corresponding vectors $v, v' \in \mathbb{R}^{2^d}$, this implies the following bound:

$$\|v - v'\|_\infty = \max_{S \subseteq [d]} |v_S - v'_S| \leq L' \cdot \|x - x'\|_2.$$

Next, Lemma 4.7 shows that we have the following bound on the difference in attributions:

$$\|\phi(f, x) - \phi(f, x')\|_2 = \|Av - Av'\|_2 \leq \|A\|_{1,\infty} \cdot \|v - v'\|_\infty,$$

where the norm $\|A\|_{1,\infty}$ depends on the chosen summary technique. Combining these bounds, we arrive at the following result:

$$\|\phi(f, x) - \phi(f, x')\|_2 \leq \|A\|_{1,\infty} \cdot L' \cdot \|x - x'\|_2.$$

Substituting in the specific values for L' and $\|A\|_{1,\infty}$ completes the proof.

□

Theorem B.2. *The robustness of removal-based explanations to model perturbations is given by the following meta-formula,*

$$\|\phi(f, x) - \phi(f', x)\|_2 \leq h(\text{summary}) \cdot \|f - f'\|,$$

where the functional distance and factor associated with the summary technique are specified

as follows:

$$\|f - f'\| = \begin{cases} \|f - f'\|_\infty & \textit{if removal} = \textbf{\textit{baseline}} \textit{ or } \textbf{\textit{marginal}} \\ \|f - f'\|_{\infty,\mathcal{X}} & \textit{if removal} = \textbf{\textit{conditional}}, \end{cases}$$

$$h(\text{summary}) = \begin{cases} 2\sqrt{d} & \textit{if summary} = \textbf{\textit{Shapley}}, \textbf{\textit{Banzhaf}}, \textit{ or } \textbf{\textit{leave-one-out}} \\ \sqrt{d} & \textit{if summary} = \textbf{\textit{mean when included}}. \end{cases}$$

Proof. Given two models f and f', the results in Lemmas 4.5 and 4.6 imply the following bound for any feature set $\boldsymbol{x}_S$,

$$|f(x_S) - f'(x_S)| \leq \|f - f'\|,$$

where the specific norm is $\|f - f'\|_\infty$ when using the baseline or marginal approach and $\|f - f'\|_{\infty,\mathcal{X}}$ when using the conditional approach. In terms of the corresponding vectors $v, v' \in \mathbb{R}^{2^d}$, this implies the following bound:

$$\|v - v'\|_\infty = \max_{S \subseteq [d]} |v_S - v'_S| \leq \|f - f'\|.$$

Next, Lemma 4.7 showed that we have the following bound on the difference in attributions:

$$\|\phi(f, x) - \phi(f', x)\|_2 = \|Av - Av'\|_2 \leq \|A\|_{1,\infty} \cdot \|v - v'\|_\infty,$$

where the norm $\|A\|_{1,\infty}$ depends on the chosen summary technique. Combining these bounds, we arrive at the following result:

$$\|\phi(f, x) - \phi(f', x)\|_2 \leq \|A\|_{1,\infty} \cdot \|f - f'\|.$$

Substituting in the specific values for $\|f - f\|$ and $\|A\|_{1,\infty}$ completes the proof.

□

Corollary B.1. *The robustness of removal-based explanations to simultaneous input and model perturbations is given by the following meta-formula,*

$$\|\phi(f,x) - \phi(f',x')\|_2 \leq g(\text{removal}) \cdot h(\text{summary}) \cdot \|x - x'\|_2 + h(\text{summary}) \cdot \|f - f'\|,$$

where the functional distance and factors associated with the removal and summary technique are given in Theorems 4.1 and 4.2.

Proof. This result follows from triangle inequality:

$$\|\phi(f,x) - \phi(f',x')\|_2 \leq \|\phi(f,x) - \phi(f,x')\|_2 + \|\phi(f,x') - \phi(f',x')\|_2.$$

The bounds for the first and second terms are given by Theorem 4.1 and Theorem 4.2, respectively. □

B.5 The Role of Sampling When Removing Features

In this section, we present results analyzing the impact of sampling when removing features. Sampling is often required in practice to integrate across the distribution for removed features (see $q(\boldsymbol{x}_{\bar{S}})$ in eq. 4.1), and this can create further discrepancy between attributions for similar inputs.

B.5.1 Theory

Here, we provide a probabilistic bound for the difference between the model prediction $f(x_S)$ with a subset of features x_S and its estimator $\hat{f}(x_S)$ with m independent samples from $q(\boldsymbol{x}_{\bar{S}})$.

Lemma B.1. *Let* $\hat{f}(x_S) \equiv \frac{1}{m}\sum_{i=1}^{m} f(x_S, x_{\bar{S}}^{(i)})$ *be the empirical estimator of* $f(x_S)$ *in eq.* 4.1*, where* $x_{\bar{S}}^{(i)} \overset{i.i.d.}{\sim} q(\boldsymbol{x}_{\bar{S}})$*. For any* $\delta \in (0,1)$ *and* $\varepsilon > 0$*, if the sample size* m *satisfies* $m \geq \frac{2B^2 \log(2/\delta)}{\varepsilon^2}$*, then* $\mathbb{P}\left(|\hat{f}(x_S) - f(x_S)| \leq \varepsilon\right) \geq 1 - \delta$.

Proof. With Assumption 4.2, we have $-\frac{B}{m} \leq \frac{f(x_S, x_{\bar{S}}^{(i)})}{m} \leq \frac{B}{m}$. Also, $\mathbb{E}[\hat{f}(x_{\bar{S}})] = f(x_{\bar{S}})$ because

$x_{\bar{S}}^{(i)}$ are sampled from $q(\boldsymbol{x}_{\bar{S}})$. Applying Hoeffding's inequality (Hoeffding, 1994), we obtain

$$\begin{aligned}\mathbb{P}(|\hat{f}(x_S) - f(x_S)| \geq \varepsilon) &\leq 2\exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{m}\left(\frac{2B}{m}\right)^2}\right) \\ &= 2\exp\left(-\frac{m\varepsilon^2}{2B^2}\right) \leq 2\exp\left(-\frac{\left(\frac{2B^2\log(2/\delta)}{\varepsilon^2}\right)\varepsilon^2}{2B^2}\right) = \delta,\end{aligned}$$

and hence for the complement event we have

$$\mathbb{P}(|\hat{f}(x_S) - f(x_S)| \leq \varepsilon) \geq 1 - \delta.$$

□

Lemma B.1 can be applied to bound the difference between the original and perturbed attributions when the integral in eq. 4.1 is estimated via sampling. For example, under input perturbations, the difference $|\hat{f}(x_S) - \hat{f}(x'_S)|$ decomposes into $|\hat{f}(x_S) - f(x_S)| + |f(x_S) - f(x'_S)| + |f(x'_S) - \hat{f}(x'_S)|$ by the triangle inequality. The first and last terms can be bounded with high probability with Lemma B.1 and the union bound, and the middle term is bounded by Lemma 4.1 or Lemma 4.2. Intuitively, by combining probabilistic bounds over all the feature subsets, it is possible to derive a final probabilistic upper bound with the following meta-formula:

$$\|\phi(f,x) - \phi(f,x')\|_2 \leq g(\text{removal}) \cdot h(\text{summary}) \cdot \|x - x'\|_2 + g_{\text{sampling}}(\text{removal}), \qquad \text{(B.2)}$$

where $g_{\text{sampling}}(\text{removal})$ is a function of δ, ε, m when removal = **marginal** or **conditional**, and $g_{\text{sampling}}(\textbf{baseline}) = 0$ because no sampling is needed. Similarly, for our bound under model perturbations, the term $g_{\text{sampling}}(\text{removal})$ can be included to account for sampling in feature removal.

B.5.2 Empirical Observations

In eq. B.2, the sampling-dependent term $g_{\text{sampling}}(\text{removal})$ does not depend on the input perturbation strength $\|x - x'\|_2$, so we empirically estimate $g_{\text{sampling}}(\text{removal})$ with the intercept of the empirical upper bound (as illustrated in Figure 4.1) when $\|x - x'\|_2 = 0$. As shown in Figure B.1 (left), $g_{\text{sampling}}(\textbf{baseline})$ is indeed zero, whereas $g_{\text{sampling}}(\textbf{marginal})$ and $g_{\text{sampling}}(\textbf{conditional})$ empirically decrease with increasing sample size for feature removal as expected. We observe that the sampling error tends to be lower for the conditional approach compared to the marginal approach, potentially due to lower variances in the conditional rather than marginal distributions. Finally, Figure B.1 (right) confirms that the perturbation-dependent term $g(\text{removal}) \cdot h(\text{summary}) \cdot \|x - x'\|_2$ does not depend on the removal sample size.

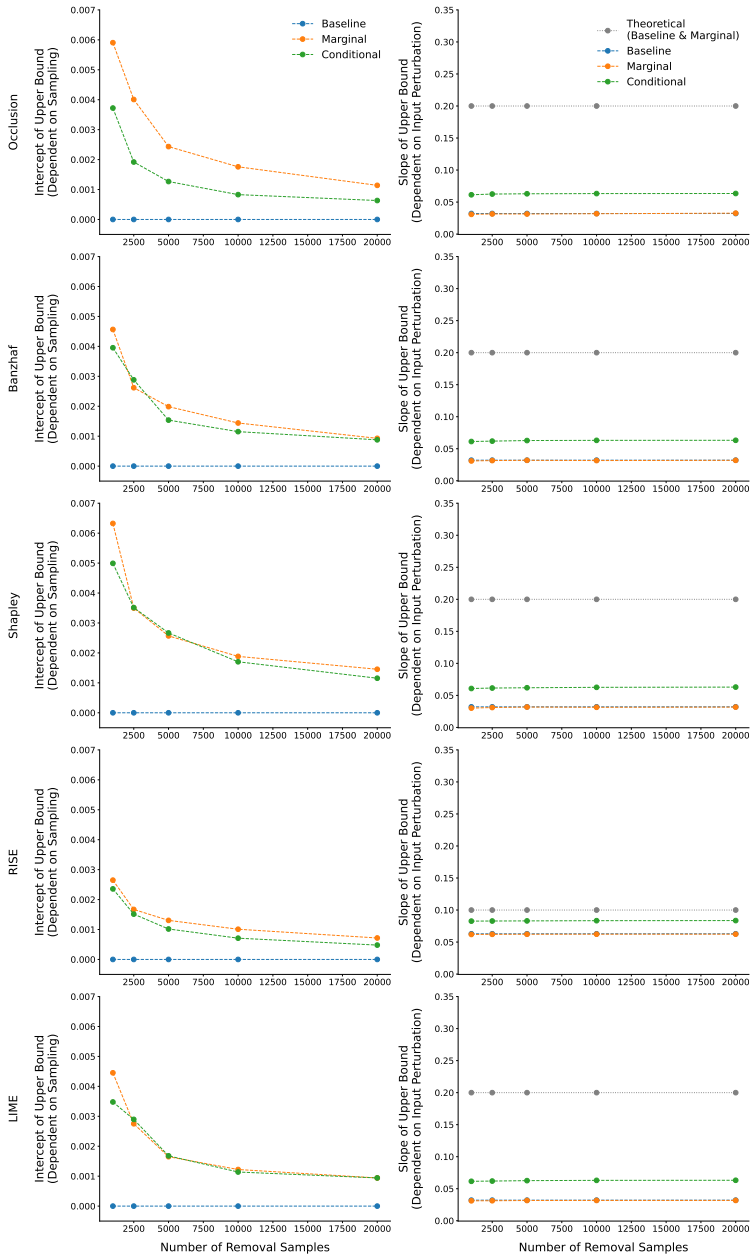


Figure B.1: Intercepts and slopes of empirical and theoretical upper bounds with respect to the input perturbation norm, for Occlusion, Banzhaf, Shapley, RISE, and LIME attribution differences in the synthetic experiment in Section 4.6, and with varying sample size for estimating feature removal.

B.6 Feature Grouping

Rather than generate attribution scores for each feature $\boldsymbol{x}_i$, another option is to partition the feature set and generate group-level scores. This is common in practice for images (Ribeiro et al., 2016; Chen et al., 2022; Covert et al., 2022), where reducing the number of features makes attributions less computationally intensive, and removing individual pixels is also unlikely to produce meaningful prediction differences. For example, LIME pre-computes superpixels using a basic segmentation algorithm (Ribeiro et al., 2016), and ViT Shapley uses grid-shaped patches defined by a vision transformer (Covert et al., 2022).

When computing group-level attributions, we require a set of feature index groups that we denote as $\{G_1, \ldots, G_g\}$, where each group satisfies $G_i \subseteq [d]$. We assume that the g groups are non-overlapping and cover all the indices, so that we have $G_i \cap G_j = \varnothing$ for $i \neq j$ and $G_1 \cup \ldots \cup G_g = [d]$. Under this approach, the attributions are a vector of the form $\phi(f, x) \in \mathbb{R}^g$, and we compute them by querying the model only with subsets that are unions of the groups. For this, we let $T \subseteq [g]$ denote a subset of group indices and denote the corresponding union of groups as $G_T = \bigcup_{i \in T} G_i \subseteq [d]$. Using this notation, the Shapley value for the ith group G_i is defined as

$$\phi_i(f, x) = \frac{1}{g} \sum_{T \subseteq [g] \setminus i} \binom{g-1}{|T|}^{-1} \left(f(x_{G_{T \cup i}}) - f(x_{G_T}) \right),$$

where $f(x_{G_T})$ is defined based on the chosen feature removal approach.

To generalize our earlier results to this setting, we first remark that querying the model with each subset induces a vector $v \in \mathbb{R}^{2^g}$ containing the predictions for each union G_T with $T \subseteq [g]$. The attributions are then calculated according to $\phi(f, x) = Av$, where $A \in \mathbb{R}^{g \times 2^g}$ is the linear operator associated with each summary technique (see Table 4.3). Our approach for bounding the attribution difference under either input or model perturbation is similar here, because we can rely on the same inequalities from Lemma 4.7, or $||A(v - v')||_2 \leq ||A||_2 \cdot ||v - v'||_2$ and $||A(v - v')||_2 \leq ||A||_{1,\infty} \cdot ||v - v'||_\infty$, depending on which norm is

available for the vectors of predictions $||v - v'||$.

Following the same proof techniques used previously, we can guarantee the following bounds in the feature grouping setting.

Input perturbation Lemma 4.1 and Lemma 4.3 guarantee that $f(\boldsymbol{x}_S)$ is Lipschitz continuous for any subset $S \subseteq [d]$. These results automatically apply to the specific subsets induced by feature grouping. If $f(x_S)$ is L'-Lipschitz continuous for all $S \subseteq [d]$, we therefore have the following inequality,

$$|f(x_{G_T}) - f(x'_{G_T})| \leq L' \cdot ||x_{G_T} - x'_{G_T}||_2,$$

for all group subsets $T \subseteq [g]$. As a result, we can guarantee that the induced vectors $v, v' \in \mathbb{R}^{2^g}$ for two inputs x, x' have the following bound:

$$||v - v'||_\infty \leq L' \cdot ||x - x'||_2.$$

Finally, we see that Theorem 4.1 only needs to be changed by replacing d with g in the h(summary) term. This bound may even be tighter than when we do not use feature groups, because it is possible that the Lipschitz constant is smaller when we restrict our attention to a smaller number of subsets.

Model perturbation Lemma 4.5 and Lemma 4.6 guarantee that the predictions $f(\boldsymbol{x}_S)$ and $f'(\boldsymbol{x}_S)$ for two models f, f' can differ by no more than the functional difference $||f - f'||$, where the specific norm depends on the feature removal approach. These results apply to all subsets $S \subseteq [d]$, so they automatically apply to those induced by feature grouping. We therefore have the following inequality,

$$|f(x_{G_T}) - f'(x_{G_T})| \leq ||f - f'||,$$

for all x and all group subsets $T \subseteq [g]$. For the induced vectors $v, v' \in \mathbb{R}^{2^g}$, we also have

$$||v - v'||_\infty \leq ||f - f'||.$$

Finally, we see that Theorem 4.2 applies as well, where the only change required is replacing d with g in the h(summary) term.

B.7 LIME Weighted Least Squares Linear Operator

In this section, we discuss the linear operator $A \in \mathbb{R}^{d \times 2^d}$ associated with LIME (Ribeiro et al., 2016). We begin with a theoretical characterization that focuses on LIME's default implementation choices, and we then corroborate these points with empirical results.

B.7.1 Theory

The observation that LIME is equivalent to a linear operator $A \in \mathbb{R}^{d \times 2^d}$ is discussed in Appendix B.3. We pointed out there that LIME's specific operator depends on several implementation choices, including the choice of weighting kernel, regularization and intercept term. Here, we consider that no regularization is used, that an intercept term is fit, and we focus on the default weighting kernel in LIME's popular implementation.[1] The default is the following exponential kernel,

$$w(S) = \exp\left(-\frac{\left(1 - \sqrt{|S|/d}\right)^2}{\sigma^2}\right), \tag{B.3}$$

where $\sigma^2 > 0$ is a hyperparameter. The user can set this parameter arbitrarily, but LIME also considers default values for each data type: $\sigma^2 = 0.25$ for images, $\sigma^2 = 25$ for language, and $\sigma^2 = 0.75\sqrt{d}$ for tabular data. We find that these choices approach two limiting cases, $\sigma^2 \to \infty$ and $\sigma^2 \to 0$, where LIME reduces to other summary techniques; namely, we find

[1] `https://github.com/marcotcr/lime`

that these limiting cases respectively recover the Banzhaf value (Banzhaf, 1964) and leave-one-out (Zeiler and Fergus, 2014) summary techniques. We argue this first from a theoretical perspective, and we then provide further empirical evidence.

Limiting case $\sigma^2 \to \infty$ In this case, it is easy to see from eq. B.3 that we have $w(S) \to 1$ for all $S \subseteq [d]$. The weighted least squares objective for LIME then reduces to an unweighted least squares problem, which according to results from Hammer and Holzman (1992) recovers the Banzhaf value. We might expect that the default settings of σ^2 approach this outcome in practice (e.g., $\sigma^2 = 25$), and we verify this in our empirical results.

Limiting case $\sigma^2 \to 0$ As we let σ^2 approach 0, increasing weight is applied to $w(S)$ when the cardinality $|S|$ is large. Similar to a softmax activation, the differences between cardinalities become exaggerated with low temperatures; for very low temperatures, the weight applied to $|S| = d$ is much larger than the weight applied to $|S| = d - 1$, which is much larger than the weight for $|S| = d - 2$, etc. In this regime, applying weight only to the subset $S = [d]$ leaves the problem undetermined, so the small residual weight on $|S| = d - 1$ plays an important role. The problem begins to resemble one where only subsets with $|S| \geq d - 1$ are taken into account, which yields a unique solution: following Covert et al. (2021), this reduces LIME to leave-one-out (Occlusion). More formally, if we denote $A_{\text{LIME}}(\sigma^2)$ as the linear operator resulting from a specific σ^2 value, we argue that $\lim_{\sigma^2 \to 0} A_{\text{LIME}}(\sigma^2)$ is the operator associated with leave-one-out. We do not prove this result due to the difficulty of deriving $A_{\text{LIME}}(\sigma^2)$ for a specific σ^2 value, but we provide empirical evidence for this claim below.

B.7.2 Empirical Analysis

Here, we denote the LIME linear operator with the exponential kernel as A_{LIME} instead of $A_{\text{LIME}}(\sigma^2)$ for brevity. Recall from Appendix B.3 that the LIME linear operator has the form $A_{\text{LIME}} = (Z'^{\top} W Z')^{-1}_{[1:]} Z'^{\top} W$, where $Z' = \{0, 1\}^{2^d \times (d+1)}$ is a binary matrix with all ones

in the first column and the other columns indicating whether each feature is in each subset, and $W \in \mathbb{R}^{2^d \times 2^d}$ is a diagonal matrix with $W_{S,S} = w(S)$ for all $S \in [d]$ and zeros elsewhere.

To empirically compare the linear operator of LIME to those of the Banzhaf value and leave-one-out (Occlusion), we first derive closed-form expressions of $Z'^{\top} W Z'$ and $Z'^{\top} W$ as in the following proposition.

Proposition B.2. *Define*

$$
\begin{aligned}
a_0 &\equiv \sum_{k=0}^{d} \binom{d}{k} \exp\left(-\left(1-\sqrt{k/d}\right)^2/\sigma^2\right), \\
a_1 &\equiv \sum_{k=1}^{d} \binom{d-1}{k-1} \exp\left(-\left(1-\sqrt{k/d}\right)^2/\sigma^2\right), \text{ and} \\
a_2 &\equiv \sum_{k=2}^{d} \binom{d-2}{k-2} \exp\left(-\left(1-\sqrt{k/d}\right)^2/\sigma^2\right).
\end{aligned}
$$

Then $(Z'^{\top} W Z')$ *is a* $(d+1) \times (d+1)$ *matrix with the following form:*

$$
(Z'^{\top} W Z') = \begin{pmatrix} a_0 & B \\ B^{\top} & C \end{pmatrix},
$$

where $B = (a_1, ..., a_1) \in \mathbb{R}^{1 \times d}$ *, and* C *is a* $d \times d$ *matrix with* a_1 *on the diagonal entries and* a_2 *on the off-diagonal entries. Furthermore, for all* $S \in [d]$

$$
(Z'^{\top} W)_{iS} = \begin{cases} w(S), & \text{if } i = 1 \\ \mathbb{1}\{i-1 \in S\} w(S) & \text{otherwise.} \end{cases}
$$

Proof. We first find an expression for each element in $Z'^{\top} W Z'$. Consider the first diagonal

entry:

$$\begin{aligned}(Z'^{\top}WZ')_{11} = Z_1'^{\top}WZ_1' &= \mathbf{1}_{2^d}^{\top}W\mathbf{1}_{2^d} \\ &= \sum_{S\subseteq[d]} w(S) \\ &= \sum_{k=0}^{d}\binom{d}{k}\exp\left(-\left(1-\sqrt{k/d}\right)^2/\sigma^2\right) = a_0.\end{aligned}$$

For the rest of the diagonal entries, we have:

$$\begin{aligned}(Z'^{\top}WZ')_{ii} = Z_i'^{\top}WZ_i' &= \sum_{S:i-1\in S} w(S) \\ &= \sum_{k=1}^{d}\binom{d-1}{k-1}\exp\left(-\left(1-\sqrt{k/d}\right)^2/\sigma^2\right) = a_1,\end{aligned}$$

for $i = 2, ..., d+1$. For the off-diagonal entries, we have:

$$(Z'^{\top}WZ')_{1i} = (Z'^{\top}WZ')_{i1} = \mathbf{1}_{2^d}^{\top}WZ_i' = \sum_{S:i-1\in S} w(S) = a_1,$$

for $i = 2, ..., d+1$. Finally, consider the rest of the off-diagonal entries:

$$\begin{aligned}(Z'^{\top}WZ')_{ij} = Z_i'^{\top}WZ_j' &= \sum_{S:i-1\in S, j-1\in S} w(S) \\ &= \sum_{k=2}^{d}\binom{d-2}{k-2}\exp\left(-\left(1-\sqrt{k/d}\right)^2/\sigma^2\right) = a_2,\end{aligned}$$

for $i, j = 2, ..., d+1$ such that $i \neq j$. Taken all these together, $Z'^{\top}WZ'$ has the closed form expression as shown in the proposition.

We now proceed to find an expression for each element in $Z'^{\top}W$, which has the first row entries:

$$(Z'^{\top}W)_{1S} = \mathbf{1}_{2^d}^{\top}W_S = w(S),$$

for all $S \subseteq [d]$, whereas the rest of the rows have entries:

$$(Z'^{\top}W)_{iS} = Z_i'^{\top}W_S = \mathbb{1}\{i-1 \in S\}w(S),$$

for $i = 2, ..., d+1$. Hence the closed form expression of $Z'^{\top}W$ in the proposition follows. □

With closed form expressions for $Z'^{\top}WZ'$ and $Z'^{\top}W$, we numerically compute the LIME linear operator $A_{\text{LIME}} = (Z'^{\top}WZ')^{-1}_{[1:]}Z'^{\top}W$ and compare it to the linear operators of the Banzhaf value (A_{Banzhaf}) and Occlusion ($A_{\text{Occlusion}}$) with both large and small σ^2 values.

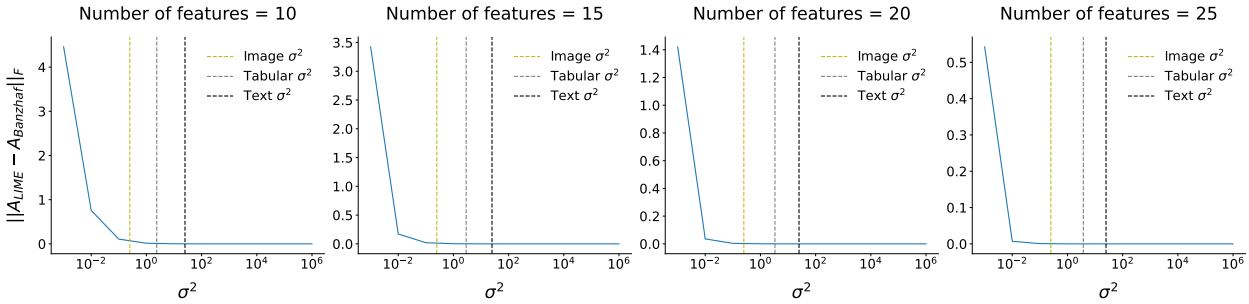


Figure B.2: Frobenius norm of the difference between the LIME linear operator A_{LIME} and the Banzhaf linear operator A_{Banzhaf}, with varying LIME exponential kernel hyperparameter σ^2 and number of features. We include reference lines for the default LIME hyperparameters: $\sigma^2 = 0.25$ for images, $\sigma^2 = 0.75\sqrt{d}$ for tabular data, and $\sigma^2 = 25$ for text data.

As shown in Figure B.2, as σ^2 increases, the Frobenius norm difference between the LIME and Banzhaf linear operators approaches zero, verifying our theoretical argument for the limiting case of $\sigma^2 \to \infty$. Furthermore, we observe that A_{LIME} and A_{Banzhaf} have near zero difference for all default σ^2 values when there are $\{10, 15, 20, 25\}$ features. As σ^2 approaches zero, the Frobenius norm difference between the LIME and Occlusion linear operators empirically approaches zero (Figure B.3), confirming our intuition for the limiting case where $\sigma^2 \to 0$. Based on our computational results, LIME with the default σ^2 values indeed has similar spectral norm $\|\cdot\|_2$ and the operator norm $\|\cdot\|_{1,\infty}$ compared to the Banzhaf value (Figure B.4).

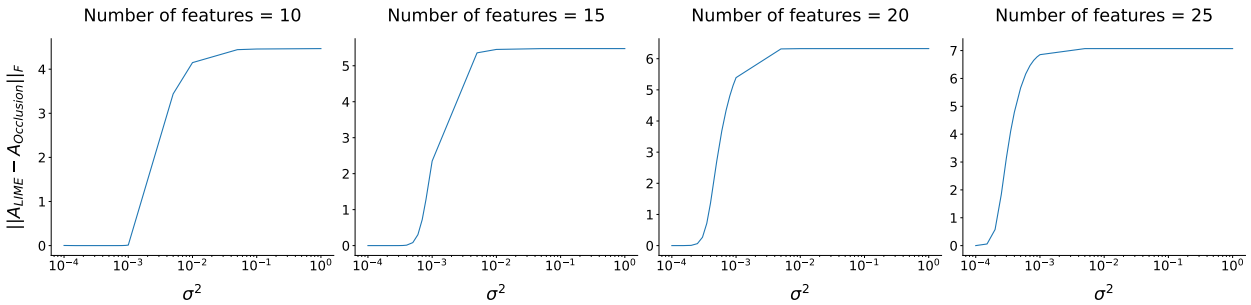


Figure B.3: Frobenius norm of the difference between the LIME linear operator A_{LIME} and the Occlusion linear operator $A_{\text{Occlusion}}$, with varying LIME parameter σ^2 and number of features.

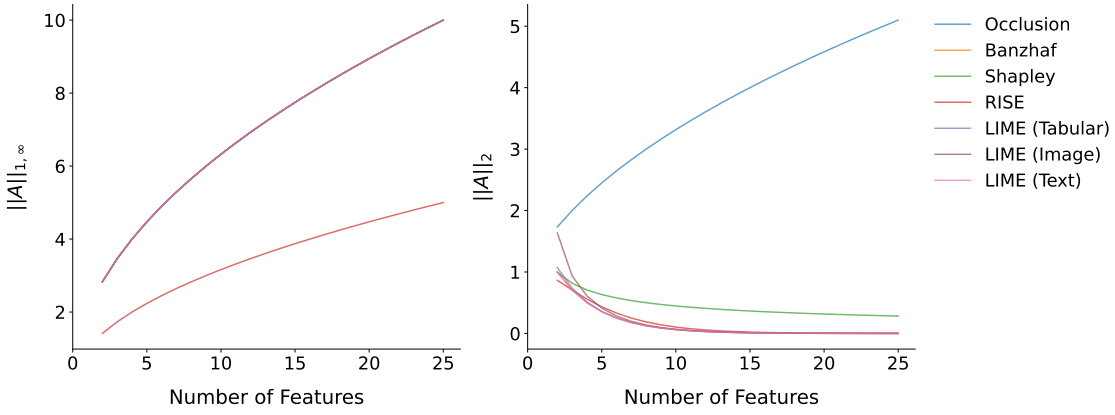


Figure B.4: Computed norms of summary technique linear operators. Note that all lines except for RISE overlap on the left-hand-side showing the operator norm.

B.8 Additional Results for Synthetic Experiments

This section provides several of the additional results described in Section 4.6.

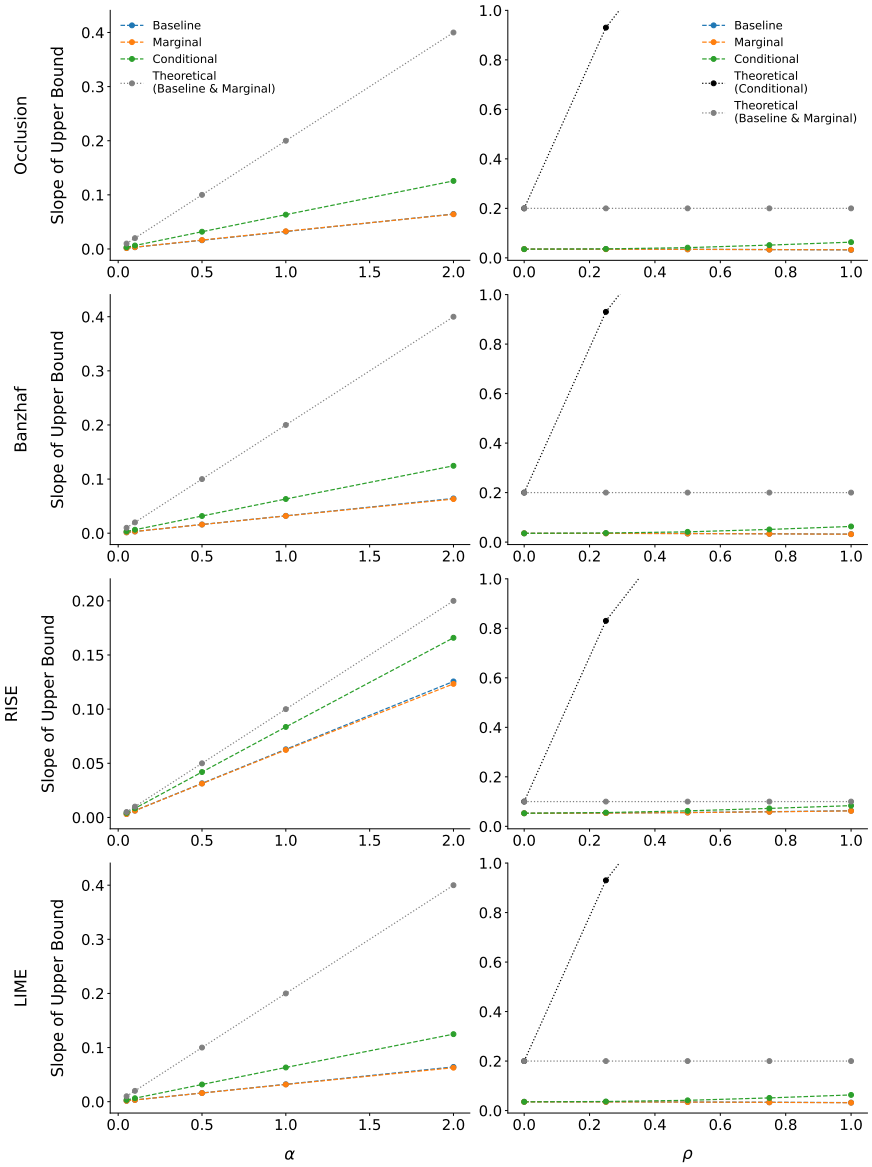


Figure B.6: Slopes of empirical and theoretical upper bounds with respect to the input perturbation norm, for Occlusion, Banzhaf, RISE, and LIME attribution differences in the synthetic experiment. The parameter α varies the logistic regression Lipschitz constant, and ρ varies the correlation between the two synthetic features $\boldsymbol{x}_1, \boldsymbol{x}_2$.

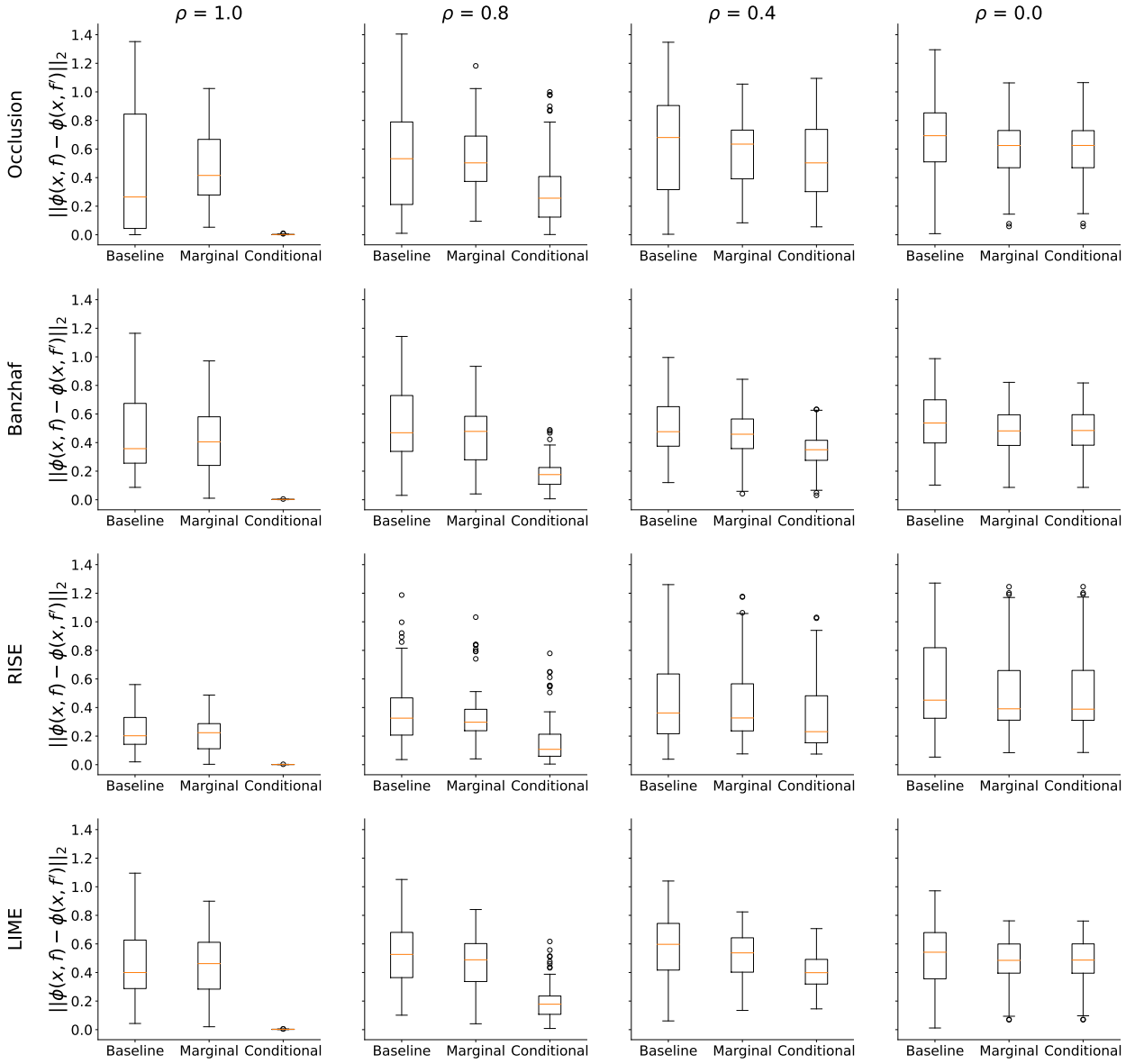


Figure B.7: Occlusion, Banzhaf, RISE, and LIME attribution differences between the logistic regression classifiers f, f' with baseline, marginal, and conditional feature removal, and varying correlation ρ.

B.9 Additional Results for Practical Implications

This section provides several of the additional results described in Section 4.6.

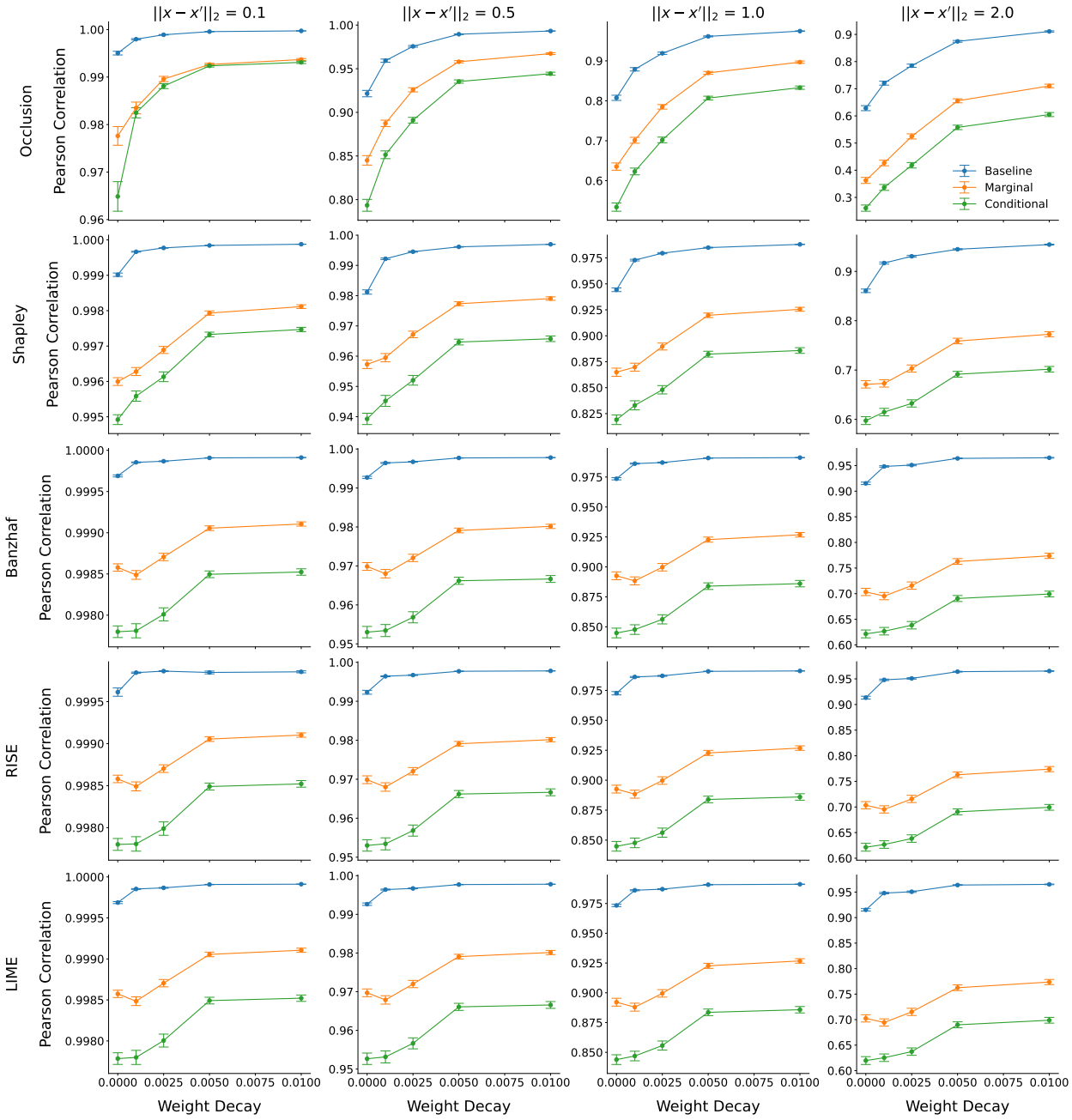


Figure B.8: Pearson correlation of attributions for networks trained with increasing weight decay, under input perturbations with perturbation norms $\{0.1, 0.5, 1, 2\}$. The results include the wine quality dataset with MLPs and baseline (replacing with training set minimums), marginal, and conditional feature removal. Error bars show the means and 95% confidence intervals across explicand-perturbation pairs.

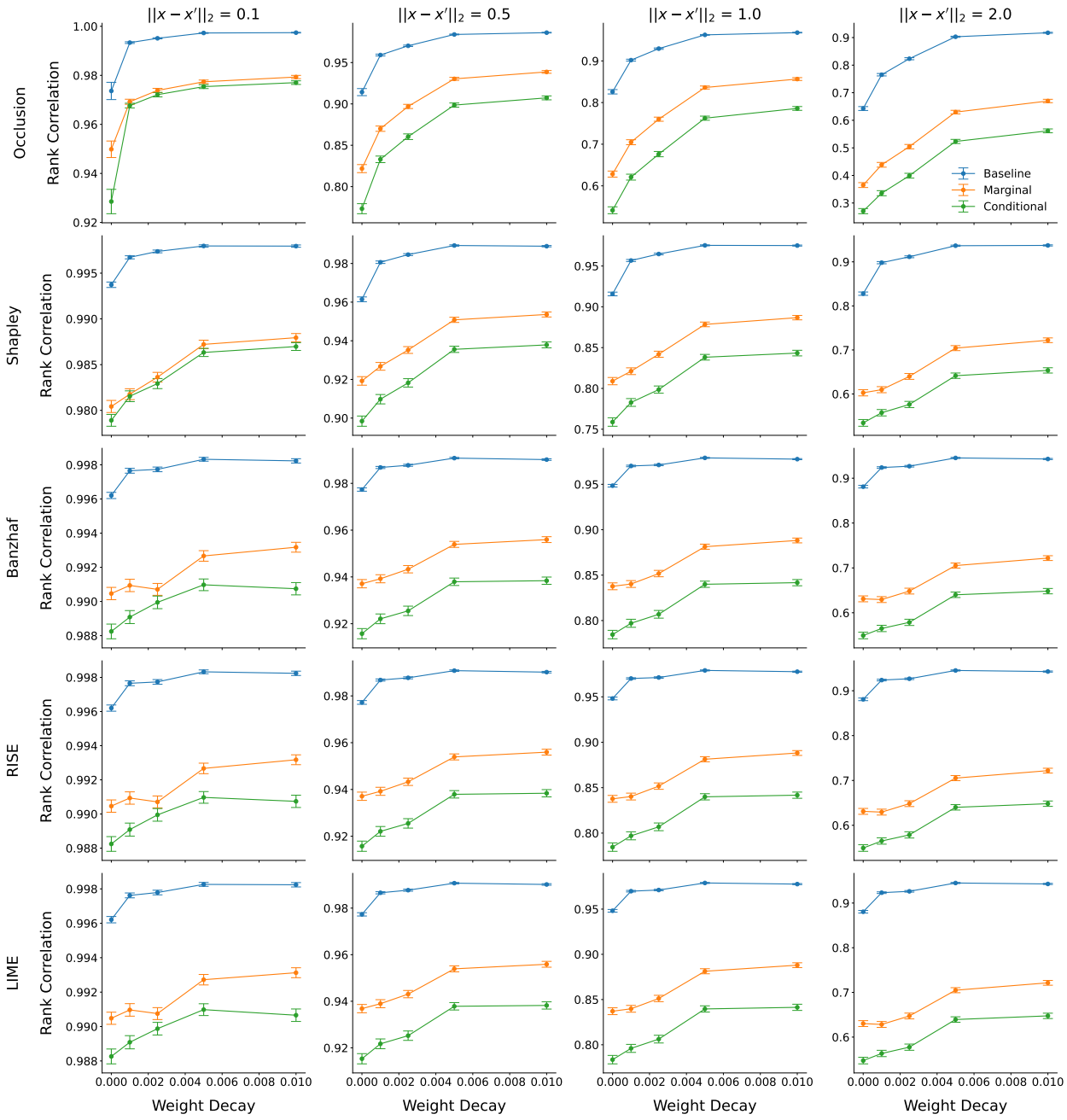


Figure B.9: Spearman rank correlation of attributions for networks trained with increasing weight decay, under input perturbations with perturbation norms $\{0.1, 0.5, 1, 2\}$. The results include the wine quality dataset with MLPs and baseline (replacing with training set minimums), marginal, and conditional feature removal. Error bars show the means and 95% confidence intervals across explicand-perturbation pairs.

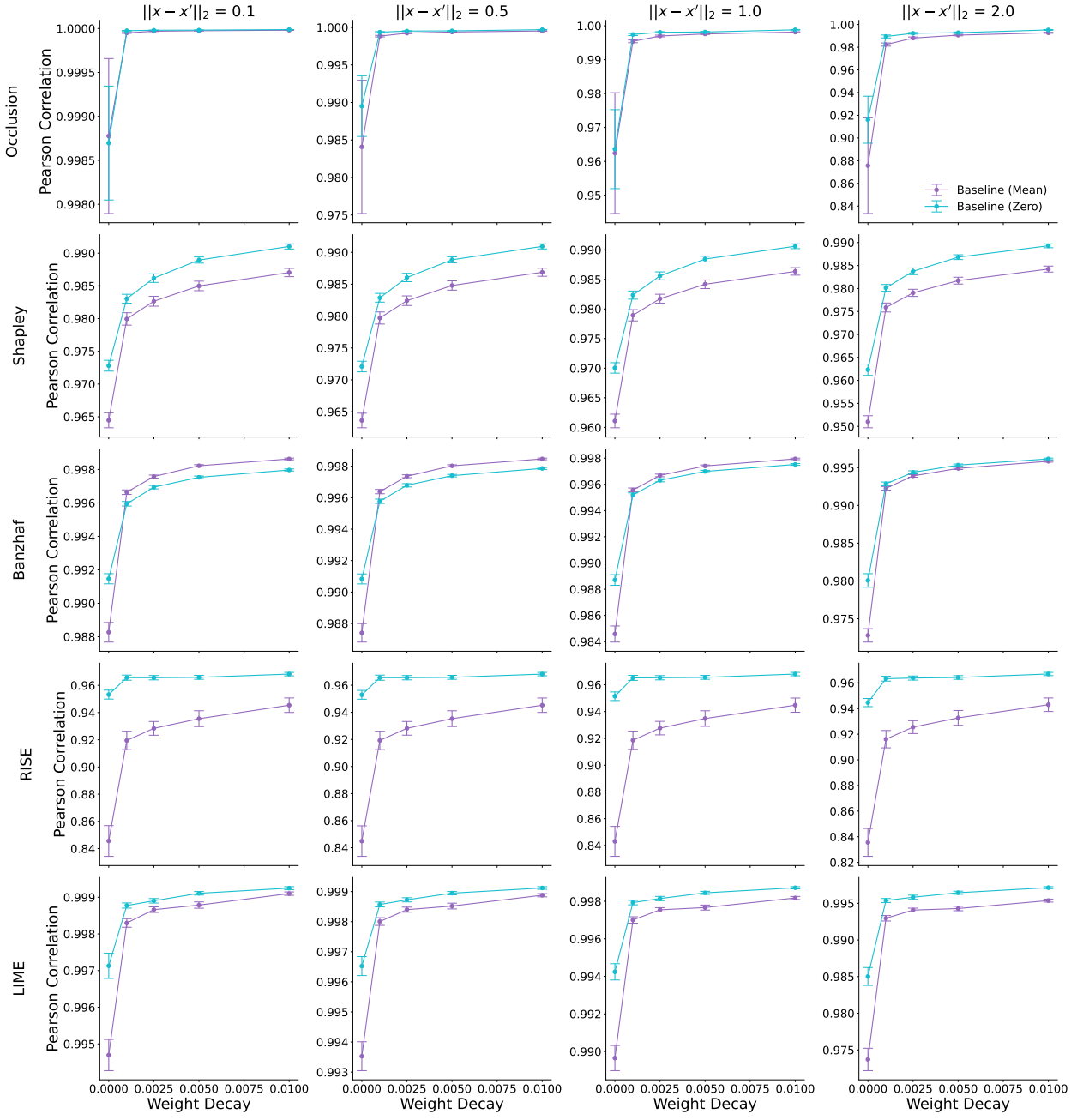


Figure B.10: Pearson correlation of attributions for networks trained with increasing weight decay, under input perturbations with perturbation norms $\{0.1, 0.5, 1, 2\}$. The results include MNIST with CNNs and baseline feature removal with either training set means or zeros. Error bars show the means and 95% confidence intervals across explicand-perturbation pairs.

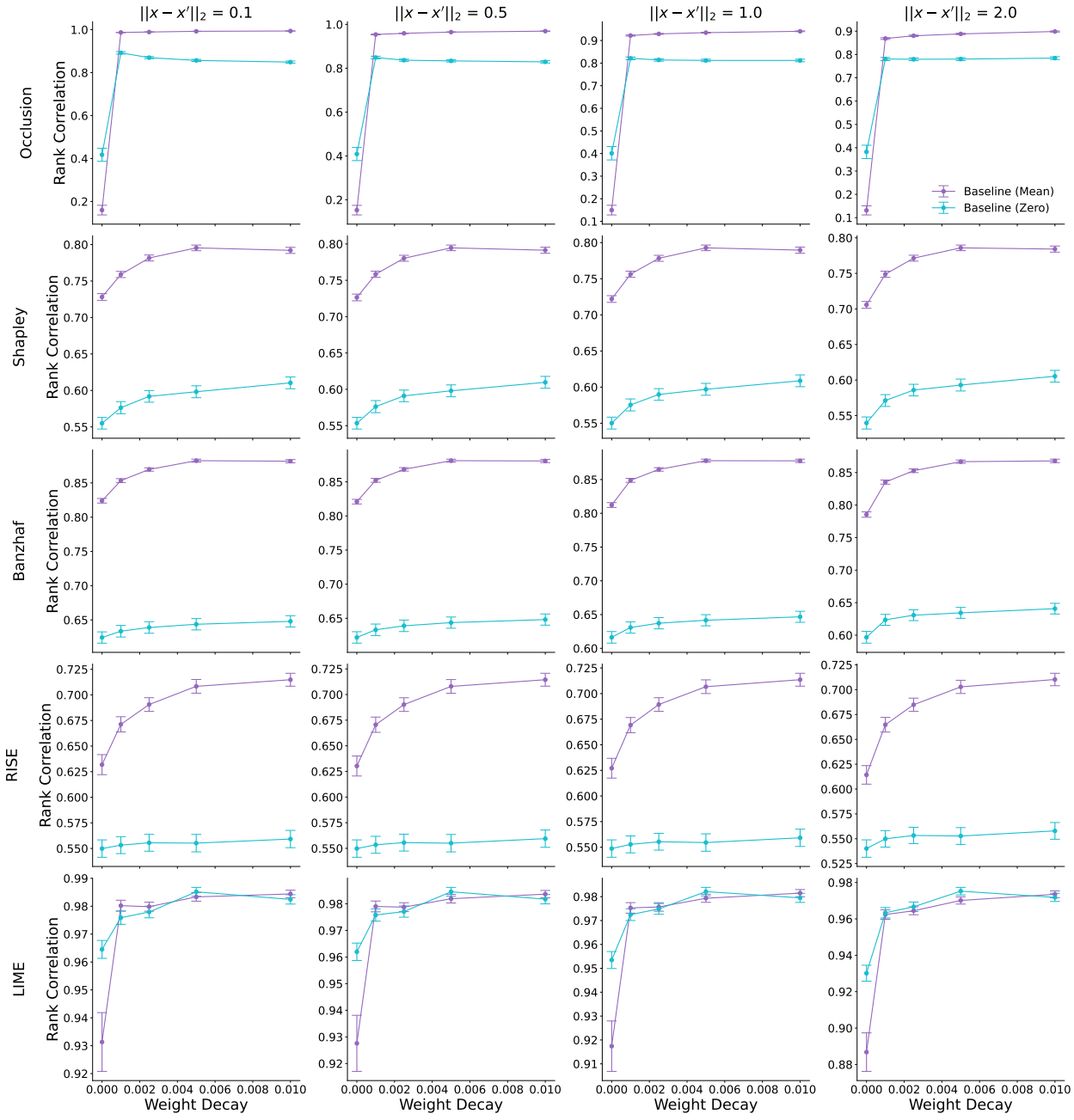


Figure B.11: Spearman rank correlation of attributions for networks trained with increasing weight decay, under input perturbations with perturbation norms $\{0.1, 0.5, 1, 2\}$. The results include MNIST with CNNs and baseline feature removal with either training set means or zeros. Error bars show the means and 95% confidence intervals across explicand-perturbation pairs.

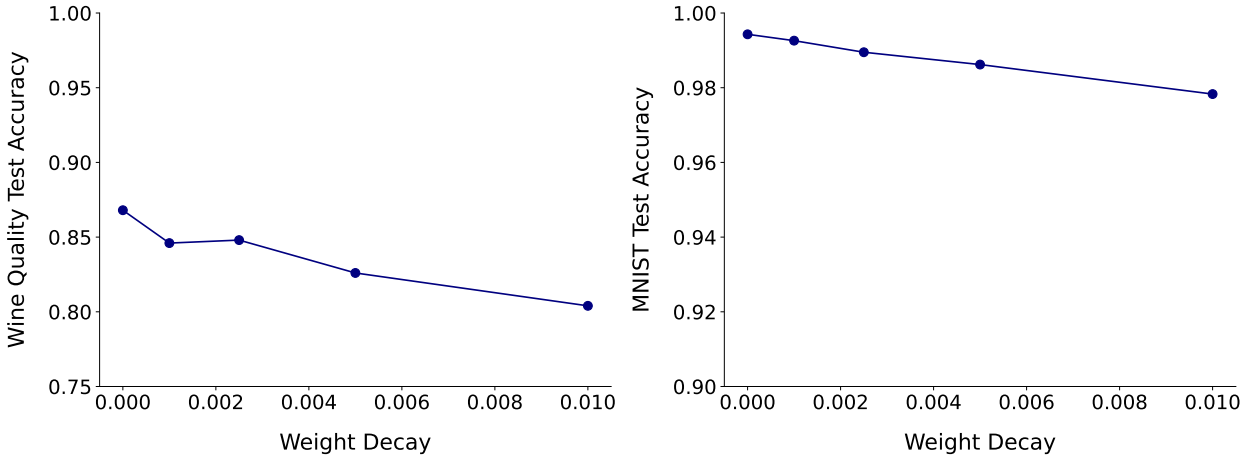


Figure B.12: Test accuracy of MLP trained on the wine quality dataset and CNN trained on MNIST with varying degree of weight decay.

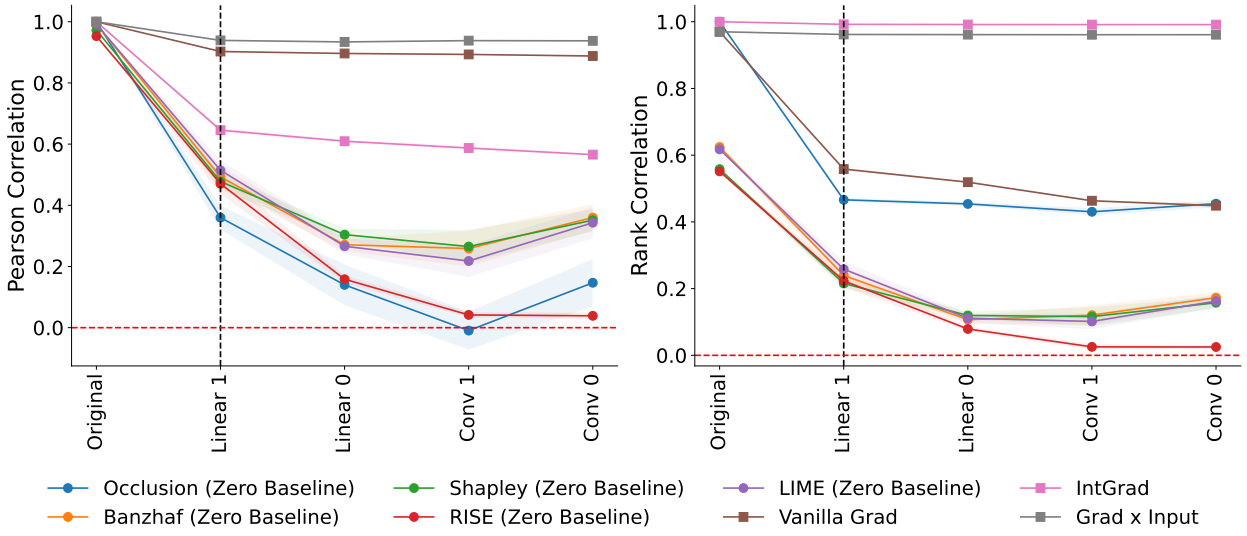


Figure B.13: Sanity checks for attributions using cascading randomization for the CNN trained on MNIST. Features are removed by replacing them with zeros. Attribution similarity is measured by Pearson correlation and Spearman rank correlation. We show the means and 95% confidence intervals across 10 random seeds.

B.10 Implementation Details

Fully connected networks For the wine quality dataset, we trained MLPs with varying levels of weight decay. The architecture consists of 3 hidden layers of width 128 with ReLU activations and was trained with Adam for 50 epochs with learning rate 10^{-3}. We also trained a conditional VAE (CVAE) following the procedure in Frye et al. (2021) to approximate conditional feature removal. The CVAE encoder and decoder both consist of 2 hidden layers of width 64 with ReLU activations and latent dimension 8. We trained the CVAE using a prior regularization strength of 0.5 for at most 500 epochs with Adam and learning rate 10^{-4}, with early stopping if the validation mean squared error did not improve for 100 consecutive epochs. All models were trained with NVIDIA GeForce RTX 2080 Ti GPUs with 11GB memory.

Convolutional networks We trained CNNs for MNIST with varying levels of weight decay. Following the LeNet-like architecture in Adebayo et al. (2018), our network has the following structure: Input $\to$ Conv(5×5, 32) $\to$ MaxPool(2×2) $\to$ Conv(5×5, 64) $\to$ MaxPool(2×2) $\to$ Flatten $\to$ Linear$(1024, 1024)$ $\to$ Linear$(1024, 10)$. We trained the model with Adam for 200 epochs with learning rate 0.001. The same GPUs for MLP training were used to train the CNNs.

Appendix C

PRACTICAL SHAPLEY VALUE ESTIMATION VIA LINEAR REGRESSION

C.1 Main Proofs

We now prove several of our main results for linear regression estimators.

C.1.1 Calculating the Exact Estimator

Recall the definition of A, which is a term in the solution to the Shapley value linear regression problem:

$$A = \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top].$$

The entries of A are straightforward to calculate because $\boldsymbol{z}$ is a random binary vector with a known distribution. Recall that $\boldsymbol{z}$ is distributed according to $p(\boldsymbol{z})$, which is defined as:

$$p(z) = \begin{cases} Q^{-1}\mu_{\text{Sh}}(z) & 0 < \mathbf{1}^\top z < d \\ 0 & \text{otherwise,} \end{cases}$$

where the normalizing constant Q is given by:

$$Q = \sum_{0<\mathbf{1}^\top z<d} \mu_{\text{Sh}}(z) = \sum_{k=1}^{d-1} \binom{d}{k} \frac{d-1}{\binom{d}{k} k(d-k)} = (d-1) \sum_{k=1}^{d-1} \frac{1}{k(d-k)}.$$

Although Q does not have a simple closed-form solution, the expression above can be calculated numerically. The diagonal entries A_{ii} are then given by:

$$\begin{aligned}
A_{ii} &= \mathbb{E}[\boldsymbol{z}_i \boldsymbol{z}_i] = p(\boldsymbol{z}_i = 1) \\
&= \sum_{k=1}^{d-1} p(\boldsymbol{z}_i = 1 \mid \mathbf{1}^\top \boldsymbol{z} = k) p(\mathbf{1}^\top \boldsymbol{z} = k) \\
&= \sum_{k=1}^{d-1} \frac{\binom{d-1}{k-1}}{\binom{d}{k}} \cdot Q^{-1} \binom{d}{k} \frac{d-1}{\binom{d}{k} k(d-k)} \\
&= \frac{\sum_{k=1}^{d-1} \frac{1}{d(d-k)}}{\sum_{k=1}^{d-1} \frac{1}{k(d-k)}}.
\end{aligned}$$

This is equal to $\frac{1}{2}$ regardless of the value of d. To see this, consider the probability $p(\boldsymbol{z}_i = 0)$:

$$\begin{aligned}
p(\boldsymbol{z}_i = 0) &= 1 - p(\boldsymbol{z}_i = 1) \\
&= 1 - \frac{\sum_{k=1}^{d-1} \frac{1}{d(d-k)}}{\sum_{k=1}^{d-1} \frac{1}{k(d-k)}} \\
&= \frac{\sum_{k=1}^{d-1} \frac{1}{d(d-k)}}{\sum_{k=1}^{d-1} \frac{1}{k(d-k)}} \\
&= p(\boldsymbol{z}_i = 1) \\
\Rightarrow A_{ii} &= \frac{1}{2}.
\end{aligned}$$

Next, consider the off-diagonal entries A_{ij} for $i \neq j$:

$$\begin{aligned} A_{ij} &= \mathbb{E}[\boldsymbol{z}_i \boldsymbol{z}_j] = p(\boldsymbol{z}_i = \boldsymbol{z}_j = 1) \\ &= \sum_{k=2}^{d-1} p(\boldsymbol{z}_i = \boldsymbol{z}_j = 1 \mid \mathbf{1}^\top \boldsymbol{z} = k) p(\mathbf{1}^\top \boldsymbol{z} = k) \\ &= \sum_{k=2}^{d-1} \frac{\binom{d-2}{k-2}}{\binom{d}{k}} \cdot Q^{-1} \binom{d}{k} \frac{d-1}{\binom{d}{k} k(d-k)} \\ &= \frac{1}{d(d-1)} \frac{\sum_{k=2}^{d-1} \frac{k-1}{d-k}}{\sum_{k=1}^{d-1} \frac{1}{k(d-k)}}. \end{aligned}$$

The value for off-diagonal entries A_{ij} depends on d, unlike the diagonal entries A_{ii}. Although it does not have a simple closed-form expression, this value can be calculated numerically in $\mathcal{O}(d)$ time.

C.1.2 Variance Reduction Proof

We present a proof for Theorem 5.1, and we then prove that a weaker condition than $G_v \succeq 0$ holds for all cooperative games (the diagonal elements satisfy $(G_v)_{ii} \geq 0$ for all games v).

Main text condition In Section 5.5, we proposed a variance reduction technique that pairs each sample $z_i \sim p(\boldsymbol{z})$ with its complement $\mathbf{1} - z_i$ when estimating b. We now provide a proof for the condition that must be satisfied for the estimator $\check{\beta}_n$ to have lower variance than $\bar{\beta}_n$. As mentioned in the main text, the multivariate CLT asserts that

$$\begin{aligned} \bar{b}_n \sqrt{n} &\xrightarrow{D} \mathcal{N}(b, \Sigma_{\bar{b}}) \\ \check{b}_n \sqrt{n} &\xrightarrow{D} \mathcal{N}(b, \Sigma_{\check{b}}), \end{aligned}$$

where

$$\Sigma_{\bar{b}} = \mathrm{Cov}\big(\boldsymbol{z}v(\boldsymbol{z})\big),$$
$$\Sigma_{\check{b}} = \mathrm{Cov}\Big(\frac{1}{2}\big(\boldsymbol{z}v(\boldsymbol{z}) + (\mathbf{1}-\boldsymbol{z})v(\mathbf{1}-\boldsymbol{z})\big)\Big).$$

We can also apply the multivariate CLT to the Shapley value estimators $\bar{\beta}_n$ and $\check{\beta}_n$. We can see that

$$\bar{\beta}_n\sqrt{n} \xrightarrow{D} \mathcal{N}(\beta^*, \Sigma_{\bar{\beta}})$$
$$\check{\beta}_n\sqrt{n} \xrightarrow{D} \mathcal{N}(\beta^*, \Sigma_{\check{\beta}}),$$

where, due to their multiplicative dependence on b estimators, the covariance matrices are defined as

$$\Sigma_{\bar{\beta}} = C\Sigma_{\bar{b}}C^\top$$
$$\Sigma_{\check{\beta}} = C\Sigma_{\check{b}}C^\top.$$

Next, we examine the relationship between $\Sigma_{\bar{b}}$ and $\Sigma_{\check{b}}$ because they dictate the relationship between $\Sigma_{\bar{\beta}}$ and $\Sigma_{\check{\beta}}$. To simplify our notation, we introduce three jointly distributed random variables, M^0, M^1 and $\bar{M}$, which are all functions of the random variable $\boldsymbol{z}$:

$$M^0 = \boldsymbol{z}v(\boldsymbol{z}) - \mathbb{E}[\boldsymbol{z}]v(\mathbf{0})$$
$$M^1 = (\mathbf{1}-\boldsymbol{z})v(\mathbf{1}-\boldsymbol{z}) - \mathbb{E}[\mathbf{1}-\boldsymbol{z}]v(\mathbf{0})$$
$$\bar{M} = \frac{1}{2}\big(M^0 + M^1\big).$$

To understand $\bar{M}$'s covariance structure, we can decompose it using standard covariance properties and the fact that $p(z) = p(\mathbf{1} - z)$ for all z:

$$\begin{aligned}
\mathrm{Cov}(\bar{M}, \bar{M})_{ij} &= \frac{1}{4}\mathrm{Cov}(M_i^0 + M_i^1, M_j^0 + M_j^1) \\
&= \frac{1}{4}\Big(\mathrm{Cov}(M_i^0, M_j^0) + \mathrm{Cov}(M_i^1, \bar{M}_j^1) + \mathrm{Cov}(M_i^0, M_j^1) + \mathrm{Cov}(M_i^1, M_j^0)\Big) \\
&= \frac{1}{2}\Big(\mathrm{Cov}(M_i^0, M_j^0) + \mathrm{Cov}(M_i^0, M_j^1)\Big).
\end{aligned}$$

We can now compare $\Sigma_{\bar{b}}$ to $\Sigma_{\check{b}}$. To account for each $\bar{M}$ sample requiring twice as many cooperative game evaluations as M^0, we compare the covariance $\mathrm{Cov}(\bar{b}_{2n})$ to the covariance $\mathrm{Cov}(\check{b}_n)$:

$$n\Big(\mathrm{Cov}(\bar{b}_{2n}) - \mathrm{Cov}(\check{b}_n)\Big)_{ij} = -\frac{1}{2}\mathrm{Cov}(M_i^0, M_j^1).$$

Based on this, we define G_v as follows:

$$\begin{aligned}
G_v &= -\mathrm{Cov}(M_i^0, M_j^1) \\
&= -\mathrm{Cov}\Big(\boldsymbol{z}v(\boldsymbol{z}) - \mathbb{E}[\boldsymbol{z}]v(\mathbf{0}), (\mathbf{1} - \boldsymbol{z})v(\mathbf{1} - \boldsymbol{z}) - \mathbb{E}[\mathbf{1} - \boldsymbol{z}]v(\mathbf{0})\Big) \\
&= -\mathrm{Cov}\Big(\boldsymbol{z}v(\boldsymbol{z}), (\mathbf{1} - \boldsymbol{z})v(\mathbf{1} - \boldsymbol{z})\Big).
\end{aligned}$$

This is the matrix referenced in Theorem 5.1. Notice that G_v is the negated cross-covariance between M^0 and M^1, which is the off-diagonal block in the joint covariance matrix for the concatenated random variable (M^0, M^1). This matrix is symmetric, unlike general cross-covariance matrices, and its eigenstructure determines whether our variance reduction approach is effective. In particular, if the condition $G_v \succeq 0$ is satisfied, then we have

$$\mathrm{Cov}(\bar{b}_{2n}) \succeq \mathrm{Cov}(\check{b}_n),$$

which implies that

$$\mathrm{Cov}(\bar{\beta}_{2n}) \succeq \mathrm{Cov}(\check{\beta}_n).$$

Since the inverses of two ordered matrices are also ordered, we get the result:

$$\mathrm{Cov}(\bar{\beta}_{2n})^{-1} \preceq \mathrm{Cov}(\check{\beta}_n)^{-1}.$$

This has implications for quadratic forms involving each matrix. For any vector $a \in \mathbb{R}^d$, we have the inequality

$$a^\top \mathrm{Cov}(\bar{\beta}_{2n})^{-1} a \leq a^\top \mathrm{Cov}(\check{\beta}_n)^{-1} a.$$

The last inequality has a geometric interpretation. It shows that the confidence ellipsoid (i.e., the confidence region, or prediction ellipsoid) for $\check{\beta}_n$ is contained by the corresponding confidence ellipsoid for $\bar{\beta}_{2n}$ since large values of n lead each estimator to converge to its asymptotically normal distribution. This is because the confidence ellipsoids are defined for $\alpha \in (0, 1)$ as

$$\begin{aligned}
\bar{E}_{2n,\alpha} &= \left\{ a \in \mathbb{R}^d : (a - \beta^*)^\top \mathrm{Cov}(\bar{\beta}_{2n})^{-1} (a - \beta^*) \leq \sqrt{\chi^2_d(\alpha)} \right\} \\
\check{E}_{n,\alpha} &= \left\{ a \in \mathbb{R}^d : (a - \beta^*)^\top \mathrm{Cov}(\check{\beta}_n)^{-1} (a - \beta^*) \leq \sqrt{\chi^2_d(\alpha)} \right\},
\end{aligned}$$

where $\chi^2_d(\alpha)$ denotes the inverse CDF of a Chi-squared distribution with d degrees of freedom evaluated at α. More precisely, we have $\check{E}_{n,\alpha} \subseteq \bar{E}_{2n,\alpha}$ because

$$\begin{aligned}
&(a - \beta^*)^\top \mathrm{Cov}(\check{\beta}_n)^{-1} (a - \beta^*) \leq \sqrt{\chi^2_d(\alpha)} \\
\Rightarrow &(a - \beta^*)^\top \mathrm{Cov}(\bar{\beta}_{2n})^{-1} (a - \beta^*) \leq \sqrt{\chi^2_d(\alpha)}.
\end{aligned}$$

This completes the proof.

A weaker condition We now discuss a weaker version of the condition described above. Consider the matrix G_v, which for a game v is defined as

$$G_v = -\mathrm{Cov}\Big(\boldsymbol{z}v(\boldsymbol{z}), (\mathbf{1}-\boldsymbol{z})v(\mathbf{1}-\boldsymbol{z})\Big).$$

A necessary (but not sufficient) condition for $G_v \succeq 0$ is that its diagonal elements are non-negative. We can prove that this weaker condition holds for all games. For an arbitrary game v, the diagonal value $(G_v)_{ii}$ is given by:

$$\begin{aligned}
(G_v)_{ii} &= -\mathrm{Cov}\Big(\boldsymbol{z}_i v(\boldsymbol{z}), (1-\boldsymbol{z}_i)v(\mathbf{1}-\boldsymbol{z})\Big) \\
&= -\,\mathbb{E}\left[\boldsymbol{z}_i(1-\boldsymbol{z}_i)v(\boldsymbol{z})v(\mathbf{1}-\boldsymbol{z})\right] + \mathbb{E}\left[\boldsymbol{z}_i v(\boldsymbol{z})\right]\mathbb{E}\left[(1-\boldsymbol{z}_i)v(\mathbf{1}-\boldsymbol{z})\right] \\
&= \mathbb{E}\left[\boldsymbol{z}_i v(\boldsymbol{z})\right]^2 \\
&= \mathbb{E}\left[v(S) \mid i \in S\right]^2 \\
&\geq 0.
\end{aligned}$$

Geometrically, this condition means that the confidence ellipsoid $\bar{E}_{2n,\alpha}$ extends beyond the ellipsoid $\check{E}_{n,\alpha}$ in the axis-aligned directions. In a probabilistic sense, it means that the variance for each Shapley value estimate is lower when using the paired sampling technique.

C.1.3 Shapley Effects

Shapley Effects is a model explanation method that summarizes the model f's sensitivity to each feature (Owen, 2014). It is based on the cooperative game

$$\tilde{w}(S) = \mathrm{Var}\big(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S]\big). \tag{C.1}$$

To show that Shapley Effects can be viewed as the expectation of a stochastic cooperative game, we reformulate this game as follows, similar to Covert et al. (2020):

$$\begin{aligned}
\tilde{w}(S) &= \mathrm{Var}\big(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S]\big) \\
&= \mathrm{Var}\big(f(\boldsymbol{x})\big) - \mathop{\mathbb{E}}_{\boldsymbol{x}_S}\big[\mathrm{Var}(f(\boldsymbol{x}) \mid \boldsymbol{x}_S)\big] \\
&= c - \mathop{\mathbb{E}}_{\boldsymbol{x}_S}\Big[\mathop{\mathbb{E}}_{\boldsymbol{x}_{[d]\setminus S}|\boldsymbol{x}_S}\big[\big(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S] - f(\boldsymbol{x}_S, \boldsymbol{x}_{[d]\setminus S})\big)^2\big]\Big] \\
&= c - \mathop{\mathbb{E}}_{\boldsymbol{x}}\Big[\big(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S] - f(\boldsymbol{x})\big)^2\Big].
\end{aligned}$$

If we generalize this cooperative game to allow arbitrary loss functions (e.g., cross entropy loss for classification tasks) rather than MSE, then we can ignore the constant value and re-write the game as

$$\tilde{w}(S) = -\mathop{\mathbb{E}}_{\boldsymbol{x}}\Big[\ell\big(\mathbb{E}[f(\boldsymbol{x}) \mid \boldsymbol{x}_S], f(\boldsymbol{x})\big)\Big].$$

Now, it is apparent that Shapley Effects is based on a cooperative game that is the expectation of a stochastic cooperative game, or $\tilde{w}(S) = \mathbb{E}_{\boldsymbol{x}}[\tilde{W}(S, \boldsymbol{x})]$, where $\tilde{W}(S, x)$ is defined as:

$$\tilde{W}(S, x) = -\ell\big(\mathbb{E}[f(\boldsymbol{x}) \mid x_S], f(x)\big).$$

Unlike the stochastic cooperative game implicitly used by SAGE, the exogenous random variable for this game is $\boldsymbol{u} = \boldsymbol{x}$.

C.2 Stochastic Cooperative Game Proofs

For a stochastic cooperative game $V(S, \boldsymbol{u})$, the generalized Shapley values are given by the expression

$$\begin{aligned}\phi_i(V) &= \frac{1}{d}\sum_{S\subseteq[d]\setminus i}\binom{d-1}{|S|}^{-1}\mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(S\cup i,\boldsymbol{u}) - V(S,\boldsymbol{u})\big] \\ &= \frac{1}{d}\sum_{S\subseteq[d]\setminus i}\binom{d-1}{|S|}^{-1}\mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(S\cup i,\boldsymbol{u})\big] - \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(S,\boldsymbol{u})\big].\end{aligned}$$

The second line above shows that the generalized Shapley values are equivalent to the Shapley values of the game's expectation, or $\phi_i(\bar{V})$, where $\bar{V}(S) = \mathbb{E}_{\boldsymbol{u}}[V(S,\boldsymbol{u})]$. Based on this, we can also understand the values $\phi_1(V),\ldots,\phi_d(V)$ as the optimal coefficients for the following weighted least squares problem:

$$\begin{aligned}&\min_{\beta_0,\ldots,\beta_d}\ \sum_z p(z)\Big(\beta_0 + z^\top\beta - \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(z,\boldsymbol{u})\big]\Big)^2 \\ &\text{s.t.}\quad \beta_0 = \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(\mathbf{0},\boldsymbol{u})\big],\quad \mathbf{1}^\top\beta = \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(\mathbf{1},\boldsymbol{u})\big] - \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(\mathbf{0},\boldsymbol{u})\big].\end{aligned}$$

Using our derivation from the main text, we can write the solution as

$$\beta^* = A^{-1}\Big(b - \mathbf{1}\frac{\mathbf{1}^\top A^{-1}b - \mathbb{E}_{\boldsymbol{u}}[V(\mathbf{1},\boldsymbol{u})] + \mathbb{E}_{\boldsymbol{u}}[V(\mathbf{0},\boldsymbol{u})]}{\mathbf{1}^\top A^{-1}\mathbf{1}}\Big),$$

where A and b are given by the expressions

$$\begin{aligned}A &= \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top] \\ b &= \mathop{\mathbb{E}}_{\boldsymbol{z}}\Big[\boldsymbol{z}\big(\mathop{\mathbb{E}}_{\boldsymbol{u}}[V(\boldsymbol{z},\boldsymbol{u})] - \mathop{\mathbb{E}}_{\boldsymbol{u}}[V(\mathbf{0},\boldsymbol{u})]\big)\Big].\end{aligned}$$

Now, we consider our adaptations of KernelSHAP and unbiased KernelSHAP and examine whether these estimators are consistent or unbiased. We begin with the stochastic version of KernelSHAP presented in the main text. Recall that this approach uses the original A

estimator $\hat{A}_n$ and the modified b estimator $\tilde{b}_n$, which is defined as:

$$\tilde{b}_n = \frac{1}{2}\sum_{i=1}^{n} z_i\big(V(z_i, u_i) - \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(\mathbf{0}, \boldsymbol{u})\big]\big).$$

As mentioned in the main text, the strong law of large numbers lets us conclude that $\lim_{n\to\infty} \hat{A}_n = A$. Thus, we can understand the b estimator's expectation as follows:

$$\begin{aligned}
\mathbb{E}\left[\tilde{b}_n\right] &= \mathop{\mathbb{E}}_{\boldsymbol{zu}}\Big[\boldsymbol{z}\big(V(\boldsymbol{z}, \boldsymbol{u}) - \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(\mathbf{0}, \boldsymbol{u})\big]\big)\Big] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z}}\Big[\boldsymbol{z}\big(\mathop{\mathbb{E}}_{\boldsymbol{u}}[V(\boldsymbol{z}, \boldsymbol{u})] - \mathop{\mathbb{E}}_{\boldsymbol{u}}[V(\mathbf{0}, \boldsymbol{u})]\big)\Big] \\
&= b.
\end{aligned}$$

With this, we conclude that $\lim_{n\to\infty} \tilde{b}_n = b$ and that $\tilde{\beta}_n$ are consistent, or

$$\lim_{n\to\infty} \tilde{\beta}_n = \beta^*.$$

To adapt unbiased KernelSHAP to the setting of stochastic cooperative games, we use the same technique of pairing independent samples of $\boldsymbol{z}$ and $\boldsymbol{u}$. To estimate b, we use an estimator $\tilde{\tilde{b}}_n$ defined as:

$$\tilde{\tilde{b}}_n = \frac{1}{n}\sum_{i=1}^{n} z_i V(z_i, u_i) - \mathbb{E}\left[\boldsymbol{z}\right] \mathop{\mathbb{E}}_{\boldsymbol{u}}\big[V(\mathbf{0}, \boldsymbol{u})\big].$$

We then substitute this into a Shapley value estimator as follows:

$$\tilde{\tilde{\beta}}_n = A^{-1}\Big(\tilde{\tilde{b}}_n - \mathbf{1}\frac{\mathbf{1}^\top A^{-1}\tilde{\tilde{b}}_n - v(\mathbf{1}) + v(\mathbf{0})}{\mathbf{1}^\top A^{-1}\mathbf{1}}\Big). \tag{C.2}$$

This is consistent and unbiased because of the linear dependence on $\tilde{\tilde{b}}_n$ and the fact that $\tilde{\tilde{b}}_n$ is unbiased:

$$\begin{aligned}
\mathbb{E}\left[\tilde{\tilde{b}}_n\right] &= \mathop{\mathbb{E}}_{\boldsymbol{zu}}\left[\boldsymbol{z}V(\boldsymbol{z},\boldsymbol{u}) - \mathbb{E}\left[\boldsymbol{z}\right]\mathop{\mathbb{E}}_{\boldsymbol{u}}\left[V(\mathbf{0},\boldsymbol{u})\right]\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z}}\left[\boldsymbol{z}\big(\mathop{\mathbb{E}}_{\boldsymbol{u}}[V(\boldsymbol{z},\boldsymbol{u})] - \mathop{\mathbb{E}}_{\boldsymbol{u}}[V(\mathbf{0},\boldsymbol{u})]\big)\right] \\
&= b.
\end{aligned}$$

With this, we conclude that $\mathbb{E}[\tilde{\tilde{\beta}}_n] = \beta^*$ and $\lim_{n\to\infty} \tilde{\tilde{\beta}}_n = \beta^*$.

C.3 Experiment Details

Here, we provide further details about experiments described in the main text.

C.3.1 Datasets and Hyperparameters

For all three explanation methods considered in our experiments – SHAP (Lundberg and Lee, 2017), SAGE (Covert et al., 2020) and Shapley Effects (Owen, 2014) – we handled removed features by marginalizing them out according to their joint marginal distribution. This is the default behavior for SHAP, but it is an approximation of what is required by SAGE and Shapley Effects. However, this choice should not affect the outcome of our experiments, which focus only on the convergence properties of our Shapley value estimators.

Both SAGE and Shapley Effects require a loss function, and we used cross entropy loss for SAGE and the soft cross entropy loss for Shapley Effects.

For the breast cancer (BRCA) subtype classification dataset, we selected 100 out of 17,814 genes to avoid overfitting on the relatively small dataset size (only 510 patients). These genes were selected at random: we tried ten random seeds and selected the subset that achieved the best performance to ensure that several relevant BRCA genes were included. A small portion of missing expression values were imputed with their mean. The data was centered and normalized prior to fitting a ℓ_1 regularized logistic regression model; the regularization

parameter was chosen using a validation set.

C.3.2 SHAP Run-Time Comparison

To compare the run-time of various SHAP value estimators, we sought to compare the ratio of the mean number of samples required by each method. For a single example x whose SHAP values are represented by β^*, the mean squared estimation error can be decomposed into the variance and bias as follows:

$$\mathbb{E}\left[||\hat{\beta}_n - \beta^*||^2\right] = \mathbb{E}\left[||\hat{\beta}_n - \mathbb{E}[\hat{\beta}_n]||^2\right] + \left|\left|\,\mathbb{E}[\hat{\beta}_n] - \beta^*\right|\right|^2.$$

Since we found that the error is dominated by variance rather than bias (Section 5.5), we can make the following approximation to relate the error to the trace of the covariance matrix:

$$\begin{aligned}\mathbb{E}\left[||\hat{\beta}_n - \beta^*||^2\right] &= \mathbb{E}\left[||\hat{\beta}_n - \mathbb{E}[\hat{\beta}_n]||^2\right] + \left|\left|\,\mathbb{E}[\hat{\beta}_n] - \beta^*\right|\right|^2 \\ &\approx \mathbb{E}\left[||\hat{\beta}_n - \mathbb{E}[\hat{\beta}_n]||^2\right] \\ &= \mathrm{Tr}\left(\mathrm{Cov}(\hat{\beta}_n)\right). \end{aligned} \tag{C.3}$$

If we define convergence based on the mean estimation error falling below a threshold value t, then the convergence condition is

$$\mathbb{E}\left[||\hat{\beta}_n - \beta^*||^2\right] \leq t.$$

Using our approximation (eq. C.3), we can see that this condition is approximately equivalent to

$$\mathbb{E}\left[||\hat{\beta}_n - \beta^*||^2\right] \approx \mathrm{Tr}\left(\mathrm{Cov}(\hat{\beta}_n)\right) \approx \frac{\mathrm{Tr}(\Sigma_{\hat{\beta}})}{n} \leq t.$$

For a given threshold t, the mean number of samples required to explain individual predictions is therefore based on the mean trace of the covariance matrix $\Sigma_{\hat{\beta}}$ (or the analogous covariance matrix for a different estimator). To compare two methods, we simply calculate

the ratio of the mean trace of the covariance matrices. These ratios are reported in Table 5.1, where each covariance matrix is calculated empirically across 100 runs with $n = 2048$ samples.

C.4 Convergence Experiments

In Section 5.5, we empirically compared the bias and variance for the original and unbiased versions of KernelSHAP using a single census income prediction. The results (Figure 5.1) showed that both versions' estimation errors were dominated by variance rather than bias, and that the original version had significantly lower variance. To verify that this result is not an anomaly, we replicated it on multiple examples and across several datasets.

First, we examined several individual predictions for the census income, German credit and bank marketing datasets. To highlight the effectiveness of our paired sampling approach presented in the main text, we added these methods as additional comparisons. Rather than decomposing the error into bias and variance as in the main text, we simply calculated the mean squared error across 100 runs of each estimator. Figure C.1 shows the error for several census income predictions, Figure C.3 for several bank marketing predictions, and Figure C.5 for several credit quality predictions. These results confirm that the original version of KernelSHAP converges significantly faster than the unbiased version, and that the paired sampling technique is effective for both estimators. The dataset sampling approach (original KernelSHAP) appears preferable in practice despite being more difficult to analyze because it converges to the correct result much faster.

Second, we calculated a global measure of the bias and variance for each estimator using the same datasets (Table C.1). Given 100 examples from each dataset, we calculated the mean bias and mean variance for each estimator empirically across 100 runs given $n = 256$ samples. Results show that the bias is nearly zero for all estimators, not just the unbiased ones; they also show that the variance is often significantly larger than the bias. However, when using the original dataset sampling approach in combination with the paired sampling technique, the bias and variance are comparably low ($\approx$0) after 256 samples. The only

Table C.1: Global measures of bias and variance for each SHAP value estimator. Each entry is the mean bias and mean variance calculated empirically across 100 examples (bias/variance, lower is better).

	Census Income	Bank Marketing	German Credit
Unbiased	0.0002/0.0208	0.0001/0.0125	0.0026/0.2561
Unbiased + Paired Sampling	0.0000/0.0068	0.0000/0.0066	0.0000/0.0062
Original (KernelSHAP)	0.0000/0.0007	0.0000/0.0006	0.0000/0.0002
Original + Paired Sampling	0.0000/0.0001	0.0000/0.0001	0.0000/0.0000

exception is the unbiased estimator that does not use paired sampling, but this is due to estimation error because its bias is provably equal to zero.

Finally, Section 5.5 also proposed assuming that the original KernelSHAP estimator's variance reduces at a rate of $\mathcal{O}(\frac{1}{n})$, similar to the unbiased version (for which we proved this rate). Although this result is difficult to prove formally, we find empirically that it holds across multiple predictions and several datasets. In Figures C.2, C.4 and C.6, we display the product of the estimator's variance with the number of samples for the census, bank and credit datasets. Results confirm that the product is roughly constant as the number of samples increases, indicating that the variance for all four estimators (not just the unbiased ones) reduces at a rate of $\mathcal{O}(\frac{1}{n})$.

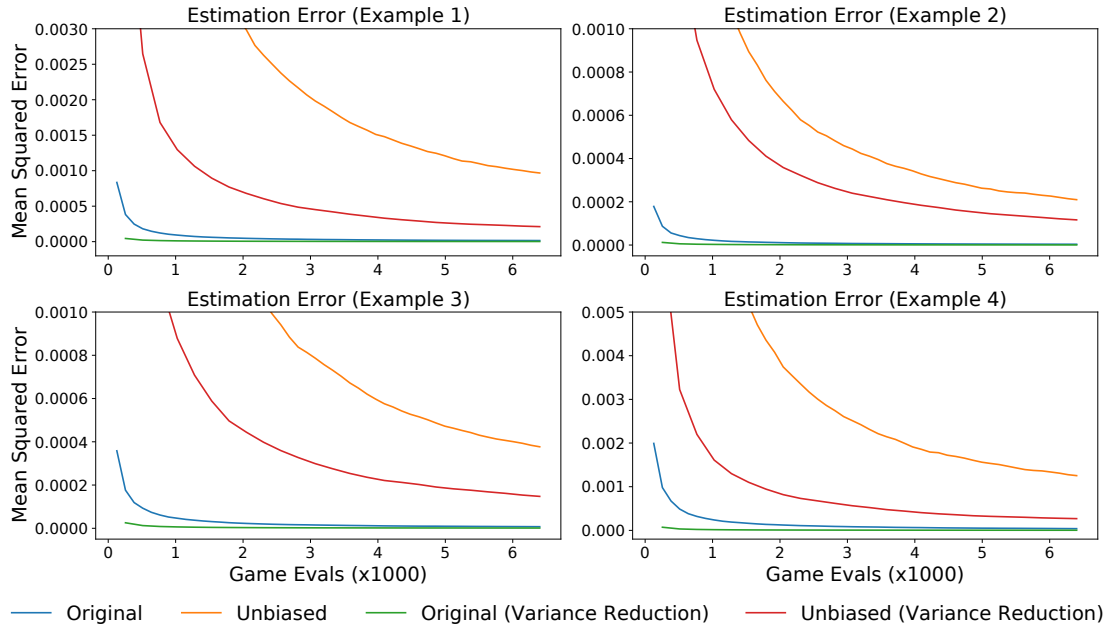


Figure C.1: Census income SHAP value estimation error on four predictions.

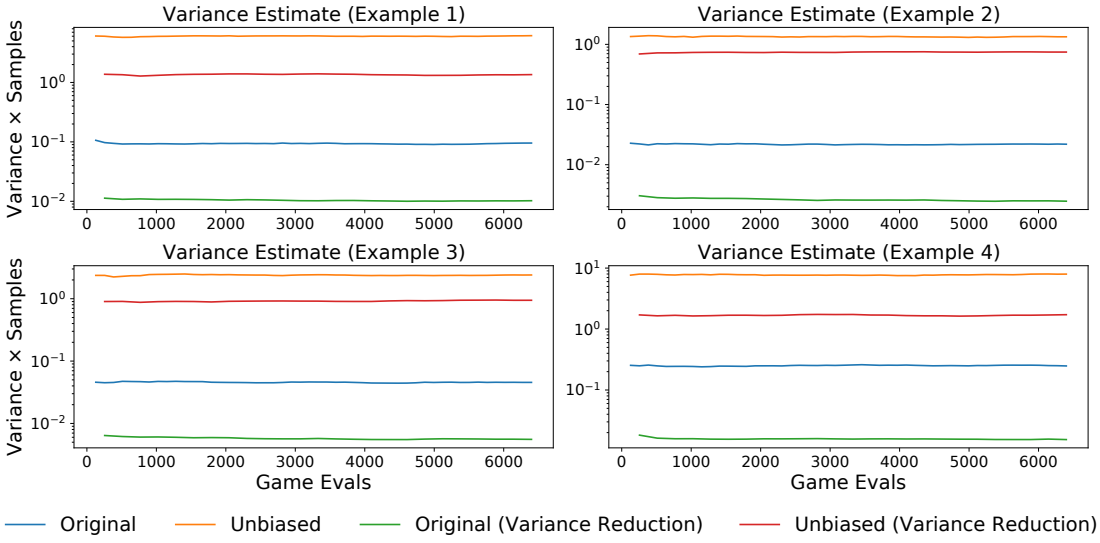


Figure C.2: Census income SHAP value variance estimation on four predictions.

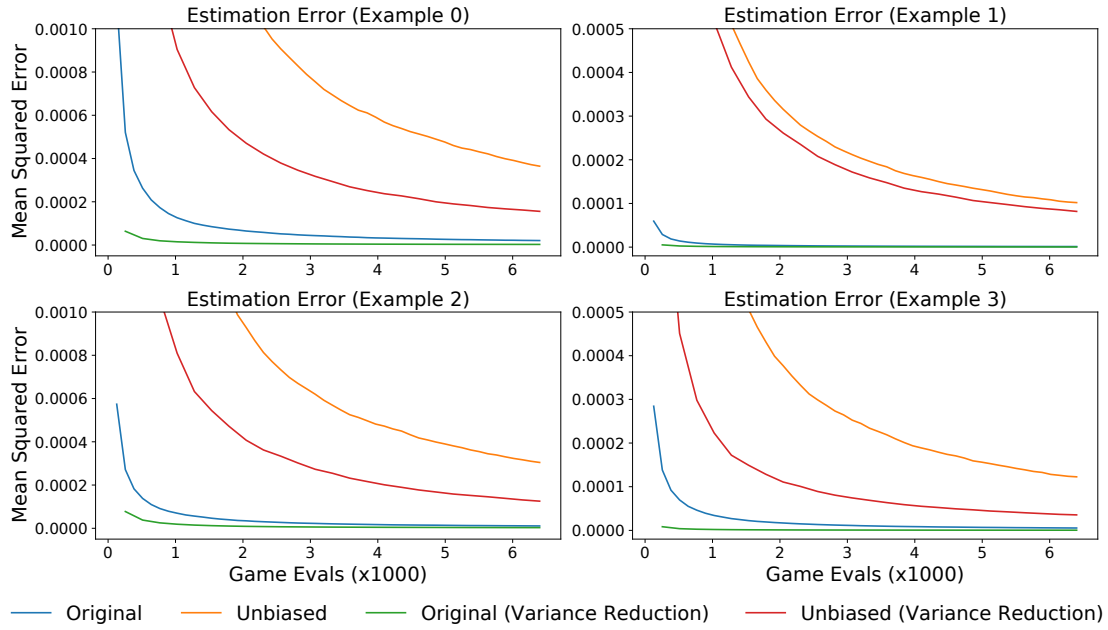


Figure C.3: Bank marketing SHAP value estimation error on four predictions.

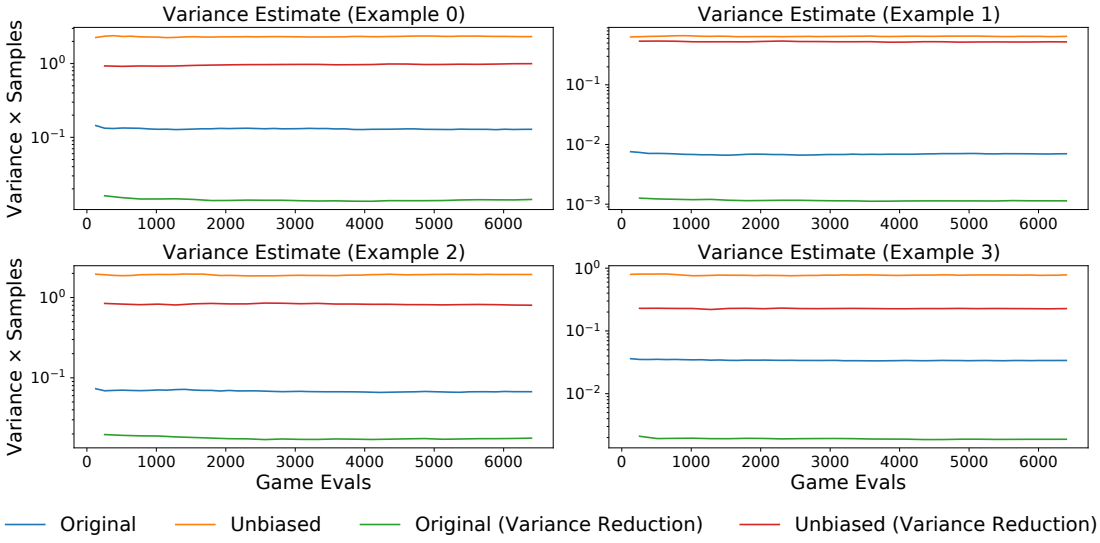


Figure C.4: Bank marketing SHAP value variance estimation on four predictions.

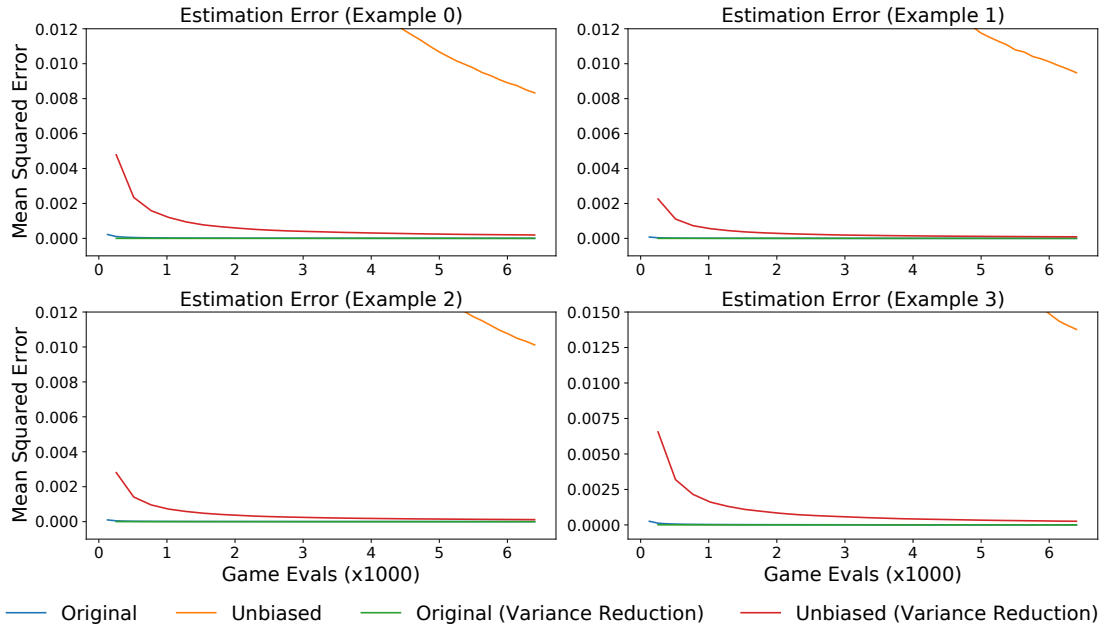


Figure C.5: German credit SHAP value estimation error on four predictions.

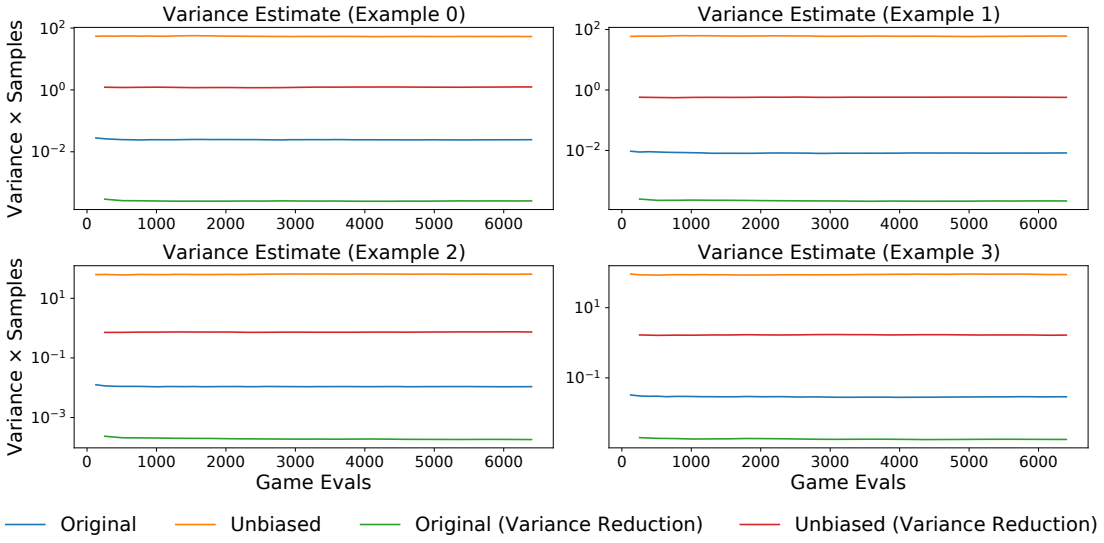


Figure C.6: German credit SHAP value variance estimation on four predictions.

Appendix D

AMORTIZED SHAPLEY VALUE ESTIMATION

D.1 FastSHAP Global Optimizer

Here, we prove that FastSHAP is trained using an objective function whose global optimizer outputs the true Shapley values. Recall that the loss function for the explainer model $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ is

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[\Big(v_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{S}) - v_{\boldsymbol{x},\boldsymbol{y}}(\varnothing) - \boldsymbol{S}^\top \phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta) \Big)^2 \Big].$$

As mentioned in the main text, it is necessary to force the model to satisfy the Efficiency constraint, which means the predictions from $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ must satisfy

$$\mathbf{1}^\top \phi_{\text{fast}}(x, y; \theta) = v_{xy}([d]) - v_{xy}(\varnothing) \quad \forall\, x \in \mathcal{X}, y \in [K].$$

One option for guaranteeing this is to adjust the model outputs using their additive efficient normalization (see Section 6.4). Incorporating this constraint on the predictions, we can then view the loss function as an expectation across $(\boldsymbol{x}, \boldsymbol{y})$, and then write the expected loss for each sample (x, y) as a separate optimization problem over the variable $\phi_{xy} \in \mathbb{R}^d$:

$$\begin{aligned} \min_{\phi_{xy}} \; & \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[\Big(v_{xy}(\boldsymbol{S}) - v_{xy}(\varnothing) - \boldsymbol{S}^\top \phi_{xy} \Big)^2 \Big] \\ \text{s.t.} \quad & \phi_{xy} = v_{xy}([d]) - v_{xy}(\varnothing). \end{aligned} \tag{D.1}$$

This is a constrained weighted least squares problem with a unique global minimizer, and it is in fact precisely the Shapley value's weighted least squares characterization (see eq. 6.3). We can therefore conclude that the optimal prediction for each pair (x, y) is the true Shapley

values, or that $\phi^*_{xy} = \phi(v_{xy})$. As a result, the global optimizer for our objective is a model $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta^*)$ that outputs the true Shapley values almost everywhere in the data distribution $p(\boldsymbol{x}, \boldsymbol{y})$.

Achieving the global optimum requires the ability to sample from $p(\boldsymbol{x})$ (or a sufficiently large dataset), perfect optimization, and a function class for $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ that is expressive enough to contain the global optimizer. The universal approximation theorem (Cybenko, 1989; Hornik, 1991) implies that a sufficiently large neural network can represent the Shapley value function to arbitrary accuracy as long as it is a continuous function. Specifically, we require the function $h_y(x) = \phi(v_{xy})$ to be continuous in x for all y. This holds in practice when we use a surrogate model parameterized by a continuous neural network, because $v_{xy}(S)$ is continuous in x for all (S, y), and the Shapley value is a linear combination of $v_{xy}(S)$ across different values of S (see eq. 6.1).

An alternative approach to enforce the efficiency property is by using an efficiency regularization term, or penalizing the efficiency gap in the explainer model's predictions (Section 6.4). If we incorporate this regularization term with parameter $\gamma > 0$, then our objective yields the following optimization problem for each (x, y) pair:

$$\min_{\phi_{xy}} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \left[\left(v_{xy}(\boldsymbol{S}) - v_{xy}(\varnothing) - \boldsymbol{S}^\top \phi_{xy} \right)^2 \right] + \gamma \left(v_{xy}([d]) - v_{xy}(\varnothing) - \mathbf{1}^\top \phi_{xy} \right)^2.$$

For finite hyperparameter values $\gamma \in [0, \infty)$, this problem relaxes the Shapley value's efficiency property and eliminates the requirement that predictions must sum to the grand coalition's value. However, as we let $\gamma \to \infty$, the penalty term becomes closer to the hard constraint in eq. D.1. Note that in practice, we use finite values for γ and observe sufficiently accurate results, whereas using excessively large γ values could render gradient-based optimization ineffective.

D.2 Additive Efficient Normalization

Here, we provide a geometric interpretation for the additive efficient normalization step and prove that it is guaranteed to yield Shapley value estimates closer to their true values. Consider a game v with Shapley values $\phi(v) \in \mathbb{R}^d$, and assume that we have Shapley values estimates $\hat{\phi} \in \mathbb{R}^d$ that do not satisfy the efficiency property. To force this property to hold, we can project these estimates onto the *efficient hyperplane*, or the subset of $\mathbb{R}^d$ where the efficiency property is satisfied. This corresponds to solving the following optimization problem over $\phi_{\text{eff}} \in \mathbb{R}^d$:

$$\min_{\phi_{\text{eff}}} \; ||\phi_{\text{eff}} - \hat{\phi}||^2 \quad \text{s.t.} \quad \mathbf{1}^\top \phi_{\text{eff}} = v([d]) - v(\varnothing).$$

We can solve the problem via its Lagrangian, denoted by $\mathcal{L}(\phi_{\text{eff}}, \nu)$, with the Lagrange multiplier $\nu \in \mathbb{R}$ as follows:

$$\begin{aligned}
\mathcal{L}(\phi_{\text{eff}}, \nu) &= ||\phi_{\text{eff}} - \hat{\phi}||^2 + \nu\Big(v([d]) - v(\varnothing) - \mathbf{1}^\top \phi_{\text{eff}}\Big) \\
\Rightarrow \phi^*_{\text{eff}} &= \hat{\phi} - \mathbf{1}\frac{v([d]) - v(\varnothing) - \mathbf{1}^\top \hat{\phi}}{d}.
\end{aligned}$$

This transformation, where the efficiency gap is split evenly and added to each estimate, is known as the *additive efficient normalization* (Ruiz et al., 1998). We implement it as an output layer for FastSHAP's predictions to ensure that they satisfy the efficiency property (Section 6.4). This step can therefore be understood as a projection of the network's output onto the efficient hyperplane.

The normalization step is guaranteed to produce corrected estimates ϕ^*_{eff} that are closer to the true Shapley values $\phi(v)$ than the original estimates $\hat{\phi}$. To see this, note that the projection step guarantees that $\hat{\phi} - \phi^*_{\text{eff}}$ and $\phi^*_{\text{eff}} - \phi(v)$ are orthogonal vectors, which implies

the following inequality:

$$\begin{aligned}||\phi(v) - \hat{\phi}||^2 &= ||\phi(v) - \phi^*_{\text{eff}}||^2 + ||\phi^*_{\text{eff}} - \hat{\phi}||^2 \\ &\geq ||\phi(v) - \phi^*_{\text{eff}}||^2.\end{aligned}$$

D.3 Reducing Gradient Variance

Recall that our objective function $\mathcal{L}(\theta)$ is defined as follows:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[\big(v_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{S}) - v_{xy}(\varnothing) - \boldsymbol{S}^\top \phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta) \big)^2 \Big].$$

The objective's gradient is given by

$$\nabla_\theta \mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[\nabla(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S}; \theta) \Big], \tag{D.2}$$

where we define

$$\nabla(x, y, S; \theta) := \nabla_\theta \big(v_{xy}(S) - v_{xy}(\varnothing) - S^\top \phi_{\text{fast}}(x, y; \theta) \big)^2.$$

When FastSHAP is trained with a single sample (x, y, S), the gradient covariance is given by $\text{Cov}\big(\nabla(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S}; \theta)\big)$, which may be too large for effective optimization. We therefore use several strategies to reduce gradient variance. First, given a model that outputs estimates for all classes $y \in [K]$, we calculate the loss jointly for all classes. This yields gradients that we denote as $\nabla(\boldsymbol{x}, \boldsymbol{S}; \theta)$, defined as

$$\nabla(x, S; \theta) := \mathop{\mathbb{E}}_{p(\boldsymbol{y}|x)} [\nabla(x, \boldsymbol{y}, S; \theta)],$$

where we have the relationship

$$\text{Cov}\big(\nabla(\boldsymbol{x}, \boldsymbol{S}; \theta)\big) \preceq \text{Cov}\big(\nabla(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S}; \theta)\big)$$

due to the law of total covariance. Next, we consider minibatches of b independent x samples, which yields gradients $\nabla_b(\boldsymbol{x}, \boldsymbol{S}; \theta)$ with covariance given by

$$\mathrm{Cov}\big(\nabla_b(\boldsymbol{x}, \boldsymbol{S}; \theta)\big) = \frac{1}{b}\mathrm{Cov}\big(\nabla(\boldsymbol{x}, \boldsymbol{S}; \theta)\big).$$

We then consider sampling m independent coalitions S for each input x, resulting in the gradients $\nabla_b^m(\boldsymbol{x}, \boldsymbol{S}; \theta)$ with covariance given by

$$\mathrm{Cov}\big(\nabla_b^m(\boldsymbol{x}, \boldsymbol{S}; \theta)\big) = \frac{1}{mb}\mathrm{Cov}\big(\nabla(\boldsymbol{x}, \boldsymbol{S}; \theta)\big).$$

Finally, we consider a *paired sampling* approach, where each sample $S \sim p(\boldsymbol{S})$ is paired with its complement $\bar{S} = [d] \setminus S$. Paired sampling has been shown to reduce KernelSHAP's variance (Covert and Lee, 2021), and our experiments show that it helps improve FastSHAP's accuracy (Appendix D.4).

The training algorithm in the main text is simplified by omitting these gradient variance reduction techniques, but they are simple to implement in practice.

D.4 FastSHAP Models and Hyperparameters

In this section, we describe the models and architectures used for each dataset, as well as the hyperparameters used when training the FastSHAP explainer model.

D.4.1 Models

Tabular datasets For the original model $f(\boldsymbol{x})$, we use neural networks for the news and marketing datasets and gradient boosted trees for the census (LightGBM Ke et al. 2017) and bankruptcy (XGBoost Chen and Guestrin 2016) datasets. The FastSHAP model $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ and the surrogate model $g(\boldsymbol{x}_S; \beta)$ are implemented using neural networks that consist of 2-3 fully connected layers with 128 units and ReLU activations. The surrogate models use a softmax output layer, while ϕ_{fast} has no output activation. The models are trained using

Adam with a learning rate of 10^{-3}, and we use a learning rate scheduler that multiplies the learning rate by a factor of 0.5 after 3 epochs of no validation loss improvement. Early stopping was triggered after the validation loss ceased to improve for 10 epochs.

Image datasets The models $f(\boldsymbol{x})$ and $g(\boldsymbol{x}_S; \beta)$ are ResNet-50 models pre-trained on Imagenet. We use these without modification to the architecture and fine-tune them on each image dataset. To create the $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ model, we modify the architecture to return a tensor of size $14 \times 14 \times K$. First, the layers after the 4th convolutional block are removed; the output of this block is $14 \times 14 \times 256$. We then append a 2D convolutional layer with K filters, each of size 1×1, so that the output is $14 \times 14 \times K$ and the yth 14×14 slice corresponds to the superpixel-level Shapley values for each class $y \in [K]$. Each model is trained using Adam with a learning rate of 10^{-3}, and we use a learning rate scheduler that multiplies the learning rate by a factor of 0.8 after 3 epochs of no validation loss improvement. Early stopping was triggered after the validation loss ceased to improve for 20 epochs.

D.4.2 FastSHAP Hyperparameters

We now explore various settings of FastSHAP's hyperparameters and observe their impact on FastSHAP's performance. There are two types of hyperparameters: sampling hyperparameters, which affect the number of samples of $\boldsymbol{S}$ taken during training, and efficiency hyperparameters, which control how we enforce the Efficiency constraint. Sampling hyperparameters include: (i) whether to use paired sampling, and (ii) the number of samples of $\boldsymbol{S}$ per $\boldsymbol{x}$ to take during training. Efficiency hyperparameters include: (i) the choice of γ in eq. 6.4, and (ii) whether to perform the additive efficient normalization during training, inference or both.

To understand the effect of sampling hyperparameters, we perform experiments using the same tabular datasets from the main text. We use the in-distribution coalitional game and compute the ground truth SHAP values the same way as in our previous experiments (i.e., by running KernelSHAP to convergence).

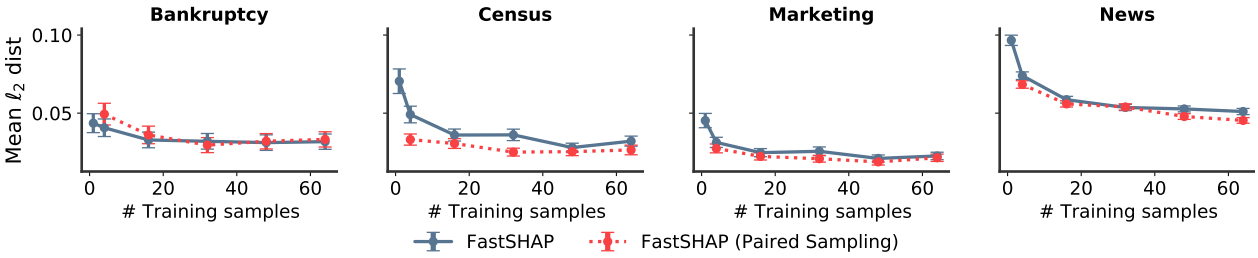


Figure D.1: FastSHAP accuracy as a function of the number of training samples. The results show that using more $\boldsymbol{S}$ samples per $\boldsymbol{x}$ improves FastSHAP's proximity to the ground truth Shapley values, as does the use of paired sampling.

Figure D.1 shows the mean ℓ_2 distance between FastSHAP's estimates and the ground truth. We find that across all four datasets, increasing the number of training samples of $\boldsymbol{S}$ generally improves the mean ℓ_2 distance to ground truth. We also find that for any fixed number of samples (greater than 1), using paired sampling improves FastSHAP's accuracy.

Table D.1 shows the results of an ablation study for the efficiency hyperparameters. *Normalization* (or *Norm.*) refers to the additive efficient normalization step (applied during training and inference, or only during inference), and *penalty* refers to the efficiency regularization technique with the parameter set to $\gamma = 0.1$. We find that using normalization during training uniformly achieves better results than without normalization or with normalization only during inference. The efficiency regularization approach proves to be less effective, generally leading to less accurate Shapley value estimates. Based on these results, we opt to use additive efficient normalization in our tabular data experiments.

D.5 Additional Results for Image Experiments

In this section, we provide additional results for the FastSHAP image experiments.

D.5.1 Inclusion and Exclusion Metrics

Table D.2 shows our inclusion and exclusion metrics when replicated using log-odds rather than accuracy. Similar to our metrics described in the main text, we choose the class pre-

Table D.1: FastSHAP ablation results. The distance to the ground truth Shapley values is displayed for several FastSHAP variations, showing that normalization helps and that the penalty is unnecessary.

	Census		Bankruptcy	
	ℓ_2	ℓ_1	ℓ_2	ℓ_1
Normalization	**0.0229**	**0.0863**	**0.0295**	**0.2436**
Normalization + Penalty	0.0261	0.0971	0.0320	0.2740
Inference Norm.	0.0406	0.1512	0.0407	0.3450
Inference Norm. + Penalty	0.0452	0.1671	0.0473	0.4471
No Norm.	0.0501	0.1933	0.0408	0.3474
No Norm. + Penalty	0.0513	0.1926	0.0474	0.4490

dicted by the original model for each image, and we measure the average log-odds for that class as we include or exclude important features according to the explanations generated by each method. The results confirm roughly the same ordering between methods, with FastSHAP being the only method to achieve strong results on both metrics for both datasets. Figure D.2 shows the raw inclusion and exclusion curves for both the accuracy and log-odds-derived metrics.

Table D.2: Exclusion and Inclusion AUCs calculated using the average log-odds of the predicted class.

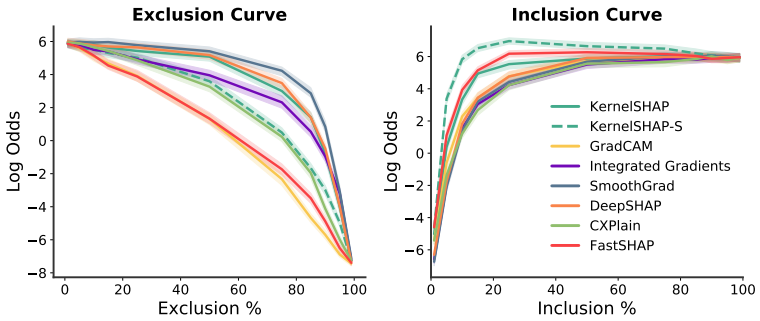
	CIFAR-10		ImageNette	
	Exclusion AUC	Inclusion AUC	Exclusion AUC	Inclusion AUC
FastSHAP	**5.92 (5.62, 6.14)**	5.36 (5.16, 5.63)	**7.98 (7.68, 8.33)**	5.40 (5.16, 5.60)
KernelSHAP	9.88 (9.55, 10.20)	5.36 (5.14, 5.63)	10.68 (10.36, 11.00)	5.07 (4.81, 5.31)
KernelSHAP-S	8.01 (7.68, 8.34)	**6.80 (6.65, 6.96)**	9.39 (9.11, 9.66)	**6.01 (5.78, 6.26)**
GradCAM	7.75 (7.44, 8.09)	4.99 (4.81, 5.26)	**7.77 (7.49, 8.05)**	4.65 (4.40, 4.89)
Integrated Gradients	8.34 (8.03, 8.61)	4.58 (4.37, 4.85)	10.14 (9.79, 10.46)	4.34 (4.10, 4.58)
SmoothGrad	10.99 (10.67, 11.29)	4.30 (4.08, 4.58)	11.19 (10.84, 11.48)	4.47 (4.24, 4.70)
DeepSHAP	9.96 (9.61, 10.24)	5.47 (5.28, 5.76)	10.93 (10.61, 11.20)	4.63 (4.38, 4.85)
CXPlain	8.34 (8.00, 8.58)	4.02 (3.80, 4.31)	9.13 (8.83, 9.41)	4.33 (4.11, 4.57)

D.5.2 FastSHAP Robustness to Limited Data

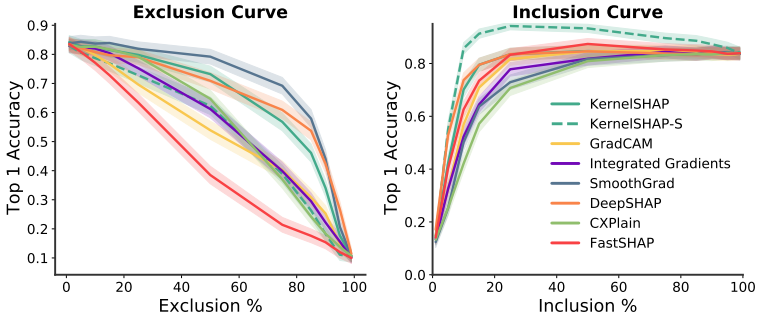
To test FastSHAP's robustness to the size of the training data, we compare its performance when trained with varying amounts of the ImageNette dataset. Figure D.3 plots the change in inclusion and exclusion AUC, calculated using top-1 accuracy, achieved when training FastSHAP with 95%, 85%, 75%, 50%, 25%, 15%, 10%, and 5% of the training dataset. We find that FastSHAP remains competitive when using just 10% of the data, and that it outperforms most baseline methods by a large margin when using just 25%.

D.5.3 Example FastSHAP Image Explanations

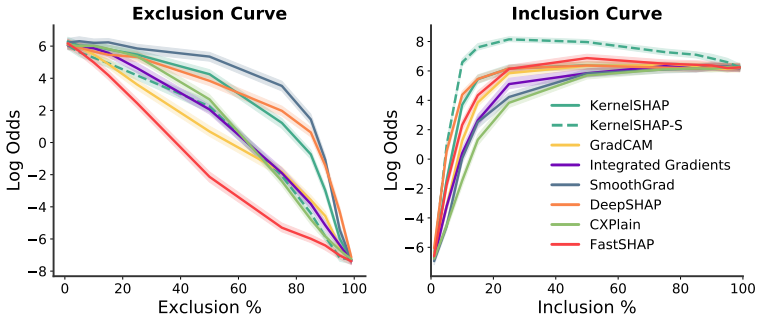
Finally, we show additional explanations generated by FastSHAP and the baseline methods for both CIFAR-10 and ImageNette. Figures D.4 and D.5 show examples for FastSHAP only, and Figures D.6 and D.7) show examples with all the methods.



(a) ImageNette: Log-odds derived inclusion and exclusion curves.



(b) CIFAR-10: Accuracy derived inclusion and exclusion curves.



(c) CIFAR-10: Log-odds derived inclusion and exclusion curves.

Figure D.2: Additional inclusion and exclusion curves. The change in top-1 accuracy or average log-odds of the predicted class as an increasing percentage of the pixels estimated to be important are excluded (left) or included (right) from the set of 1,000 images.

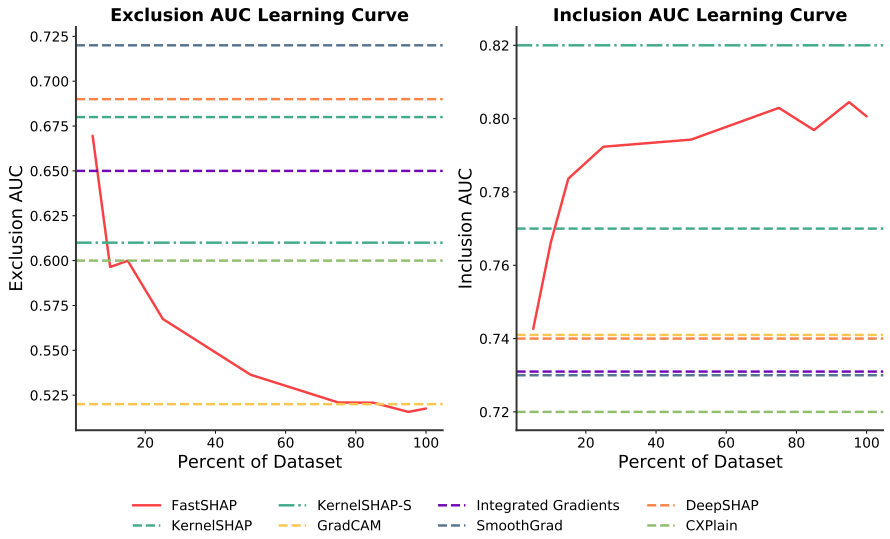


Figure D.3: FastSHAP robustness to limited data. The curves are generated by training FastSHAP with varying portions of the ImageNette dataset and evaluating the Inclusion and Exclusion AUC. Horizontal lines show the Exclusion and Inclusion AUCs for each of the baseline methods, as reported in Table 6.1.

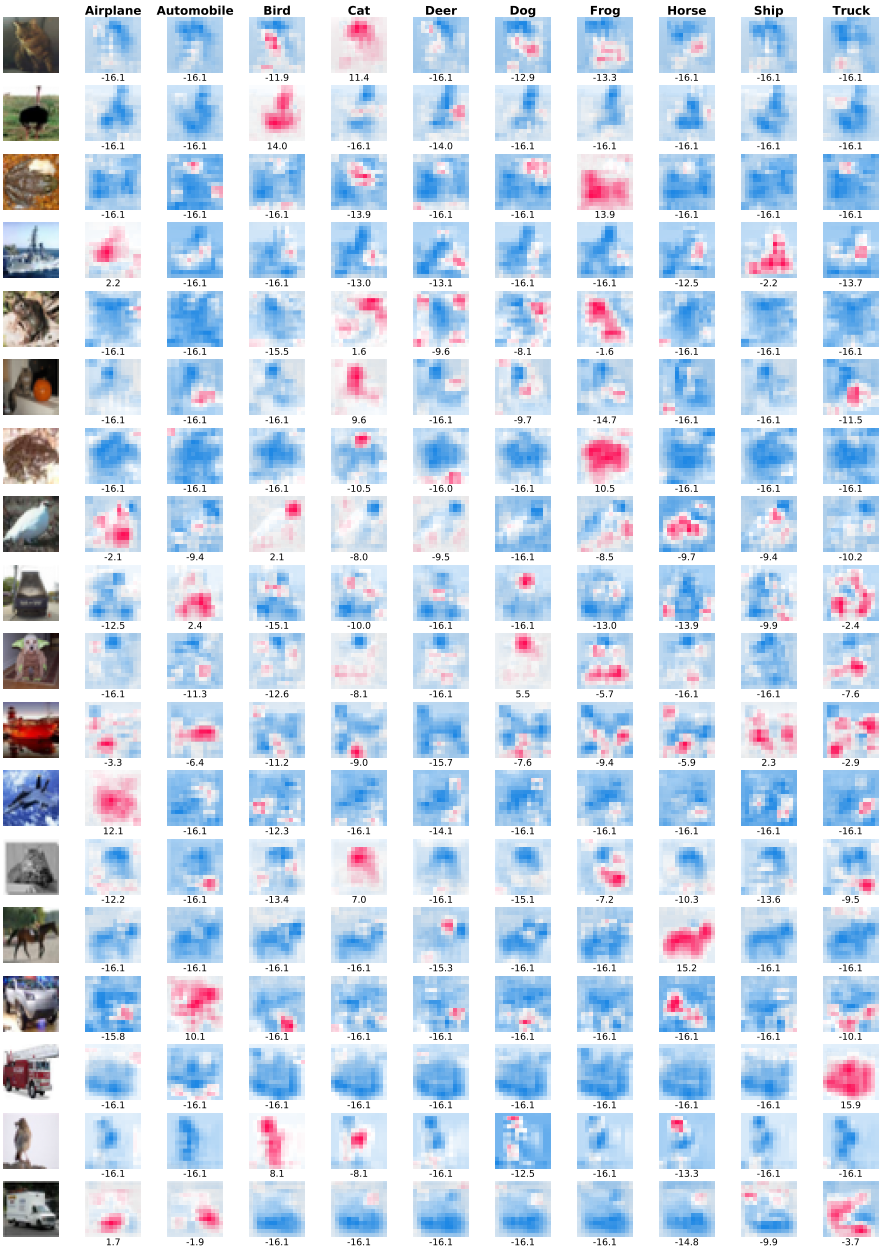


Figure D.4: Explanations generated by FastSHAP for 18 randomly selected CIFAR-10 images. Each column corresponds to a CIFAR-10 class, and the model's prediction (in logits) is provided below each image.



Figure D.5: Explanations generated by FastSHAP for 18 randomly selected ImageNette images. Each column corresponds to an ImageNette class, and the model's prediction (in logits) is provided below each image.

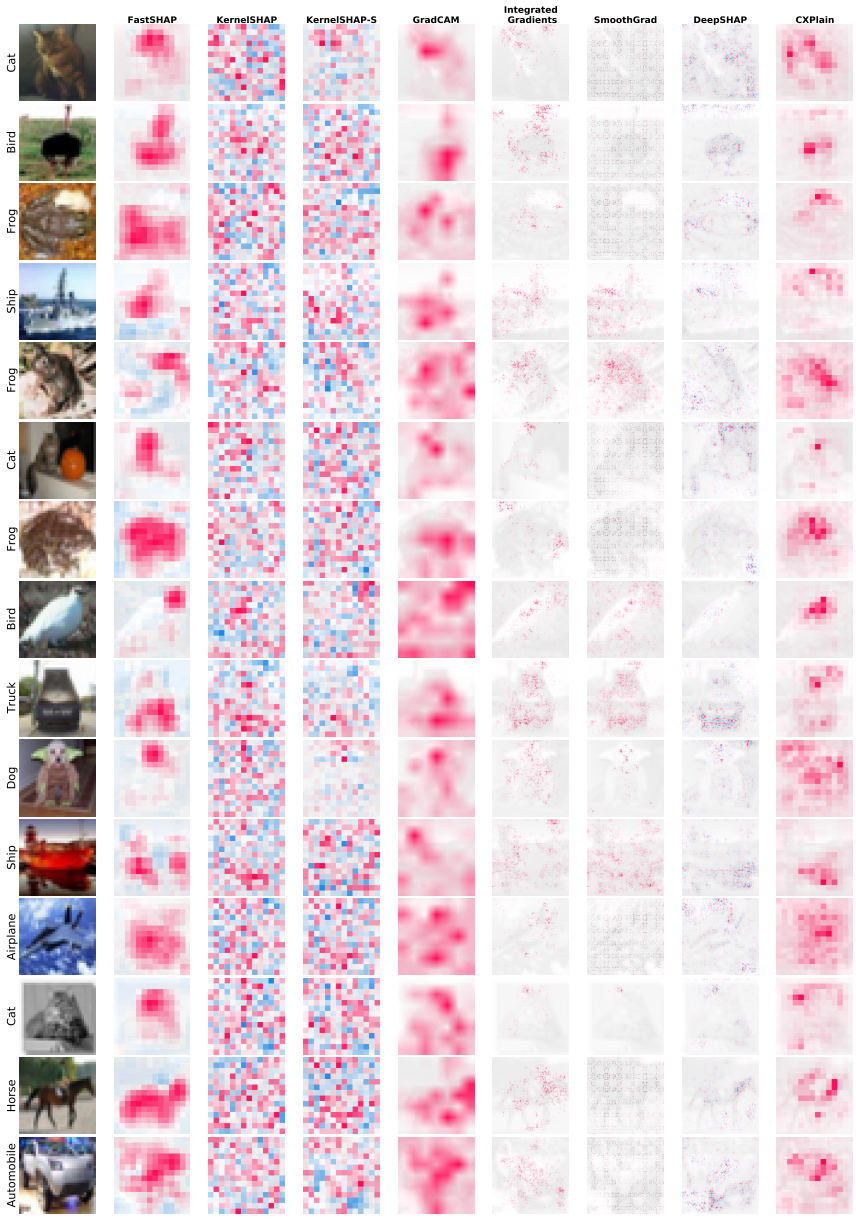


Figure D.6: Explanations generated for the predicted class for 15 randomly selected CIFAR-10 images. Each column corresponds to an explanation method, and each row is labeled with the image's corresponding class.



Figure D.7: Explanations generated for the predicted class for 15 randomly selected ImageNette images. Each column corresponds to an explanation method, and each row is labeled with the image's corresponding class.

Appendix E

LEARNING TO ESTIMATE SHAPLEY VALUES WITH VISION TRANSFORMERS

E.1 Attention Masking

This section describes our attention masking approach in detail. First, recall that ViTs use query-key-value self-attention (Vaswani et al., 2017; Dosovitskiy et al., 2020), which accepts a set of input tokens and produces a weighted sum of learned token values. Given an input $\boldsymbol{z} \in \mathbb{R}^{d \times h}$ and parameters $\mathrm{U}_{qkv} \in \mathbb{R}^{h \times 3h'}$, we compute the self attention output $\mathrm{SA}(\boldsymbol{z})$ for a single head as follows:

$$[\mathbf{Q}, \mathbf{K}, \mathbf{V}] = \boldsymbol{z}\mathrm{U}_{qkv} \tag{E.1}$$

$$\mathbf{A} = \mathrm{softmax}(\mathbf{Q}\mathbf{K}^{\top}/\sqrt{h'}) \tag{E.2}$$

$$\mathrm{SA}(\boldsymbol{z}) = \mathbf{A}\mathbf{V}. \tag{E.3}$$

In multihead self-attention, we perform this operation in parallel over k attention heads and project the concatenated outputs. Denoting each head's output as $\mathrm{SA}_i(\boldsymbol{z})$ and the projection matrix as $\mathrm{U}_{msa} \in \mathbb{R}^{k \cdot h' \times h}$, the multihead self-attention output $\mathrm{MSA}(\boldsymbol{z})$ is

$$\mathrm{MSA}(\boldsymbol{z}) = [\mathrm{SA}_1(\boldsymbol{z}), \ldots, \mathrm{SA}_k(\boldsymbol{z})]\mathrm{U}_{msa}. \tag{E.4}$$

Multihead self-attention can operate with any number of tokens, so given a subset $S \subseteq [d]$ and an input x, we can evaluate a ViT using only tokens for the patches $x_S = \{x_i : i \in S\}$. However, for implementation purposes it is preferable to maintain the same number of tokens within a minibatch. We therefore provide all tokens to the model and achieve the same effect

using attention masking. Our exact approach is described below.

Let $z \in \mathbb{R}^{d \times h}$ represent the full token set for an input x and let S be a subset. At each self-attention layer, we construct a mask matrix $\mathbf{M} = [S, \ldots, S]^\top \in \{0,1\}^{d \times d}$, where with abuse of notation we view the subset as a binary vector $\{0,1\}^d$, and we calculate the masked self-attention output $\mathrm{SA}(\boldsymbol{z}, \mathbf{M})$ as follows:

$$\mathbf{A} = \mathrm{softmax}((\mathbf{Q}\mathbf{K}^\top - (1 - \mathbf{M}) \cdot \infty)/\sqrt{h'}) \tag{E.5}$$

$$\mathrm{SA}(\boldsymbol{z}, \mathbf{M}) = \mathbf{A}\mathbf{V}. \tag{E.6}$$

The masked multihead self-attention output is then calculated similarly to the original version:

$$\mathrm{MSA}(\boldsymbol{z}, \mathbf{M}) = [\mathrm{SA}_1(\boldsymbol{z}, \mathbf{M}), \ldots, \mathrm{SA}_k(\boldsymbol{z}, \mathbf{M})]\mathrm{U}_{msa}. \tag{E.7}$$

Due to the masking in eq. E.5, each output token in $\mathrm{MSA}(\boldsymbol{z}, \mathbf{M})$ is guaranteed not to attend to tokens from $x_{\bar{S}} = \{x_i : i \notin S\}$. We use masked self-attention in all layers of the network, so that the tokens for x_S remain invariant to those for $x_{\bar{S}}$ throughout the entire model, including after the layer norm and fully-connected layers. When the final prediction is calculated using the class token, the output is equivalent to using only the tokens for x_S. If the final prediction is instead produced using global average pooling (Beyer et al., 2022), we can modify the average to account only for tokens we wish to include.

E.2 Masked Training

In this section, we provide proofs to justify training a ViT classifier with held-out tokens, either as part of the original training or as part of a post-hoc fine-tuning procedure (the surrogate model training described in Section 7.5). Our proofs are similar to those in prior work that discusses marginalizing out features using their conditional distribution (Covert et al., 2021).

First, consider a model trained directly with masking. Given a subset distribution $p(\boldsymbol{S})$

and the data distribution $p(\boldsymbol{x}, \boldsymbol{y})$, we can train a model $f(\boldsymbol{x}_{\boldsymbol{S}}; \eta)$ with cross-entropy loss and random masking by minimizing the following:

$$\min_{\eta} \quad \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} [-\log f_{\boldsymbol{y}}(\boldsymbol{x}_{\boldsymbol{S}}; \eta)]. \tag{E.8}$$

To understand the global optimizer for this loss function, consider the expected loss for the prediction given a fixed model input x_S:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{y},\boldsymbol{x}_{\bar{S}}|x_S)} [-\log f_{\boldsymbol{y}}(x_S; \eta)] = \mathop{\mathbb{E}}_{p(\boldsymbol{y}|x_S)} [-\log f_{\boldsymbol{y}}(x_S; \eta)]. \tag{E.9}$$

The expression in eq. E.9 is equal to the KL divergence $D_{\mathrm{KL}}(p(\boldsymbol{y} \mid x_S) \mid\mid f(x_S; \eta))$ up to a constant value, so the prediction that minimizes this loss is $p(\boldsymbol{y} \mid x_S)$. For any subset $S \subseteq [d]$ where $p(S) > 0$, we then have the following result for the model $f(\boldsymbol{x}_{\boldsymbol{S}}; \eta^*)$ that minimizes eq. E.8:

$$f_y(x_S; \eta^*) = p(y \mid x_S) \text{ a.e. in } p(\boldsymbol{x}).$$

Intuitively, this means that training the original model with masking estimates $f(\boldsymbol{x}_S; \eta) \approx p(\boldsymbol{y} \mid \boldsymbol{x}_S)$. In practice, we use a subset distribution $p(\boldsymbol{S})$ where $p(S) > 0$ for all $S \subseteq [d]$: we set $p(\boldsymbol{S})$ by sampling the cardinality uniformly at random and then sampling the members, which is equivalent to defining $p(\boldsymbol{S})$ as

$$p(S) = \frac{1}{\binom{d}{|S|} \cdot (d+1)}.$$

Alternatively, we can use a model $f(\boldsymbol{x}; \eta)$ trained without masking and fine-tune it to better handle held-out features. In our case, this yields a *surrogate model* (Frye et al., 2021) denoted as $g(\boldsymbol{x}_{\boldsymbol{S}}; \beta)$ that we fine-tune by minimizing the following loss:

$$\min_{\beta} \quad \mathop{\mathbb{E}}_{p(\boldsymbol{x})} \mathop{\mathbb{E}}_{p(\boldsymbol{S})} \Big[D_{\mathrm{KL}}\big(f(\boldsymbol{x}; \eta) \mid\mid g(\boldsymbol{x}_{\boldsymbol{S}}; \beta)\big) \Big]. \tag{E.10}$$

To understand the global optimizer for the above loss, we can again consider the expected loss given a fixed input x_S:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x}_{\bar{S}}|x_S)}\Big[D_{\mathrm{KL}}\big(f(\boldsymbol{x};\eta)\;||\;g(x_S;\beta)\big)\Big] = D_{\mathrm{KL}}\big(\,\mathbb{E}[f(\boldsymbol{x};\eta)\mid x_S]\;||\;g(x_S;\beta)\big) + \text{const.} \tag{E.11}$$

The distribution that minimizes this loss is the expected output given the available features, or $\mathbb{E}[f(\boldsymbol{x};\eta)\mid x_S]$. By the same argument presented above, we then have the following result for the optimal surrogate $g(\boldsymbol{x_S};\beta^*)$ that minimizes eq. E.10:

$$g(x_S;\beta^*) = \mathbb{E}[f(\boldsymbol{x};\eta)\mid x_S] \;\text{ a.e. in } p(\boldsymbol{x}).$$

Notice that if the initial model is optimal, or $f(\boldsymbol{x};\eta) = p(\boldsymbol{y}\mid\boldsymbol{x})$, then the optimal surrogate satisfies $g(\boldsymbol{x}_S;\beta^*) = p(\boldsymbol{y}\mid\boldsymbol{x}_S)$.

E.3 Explainer Training Approach

In this section, we summarize our approach for training the explainer model and describe several design choices. Recall that the explainer is a vision transformer $\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta)\in\mathbb{R}^d$ that we train by minimizing the following loss:

$$\begin{aligned}\min_{\theta}\quad & \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\mathop{\mathbb{E}}_{p_{\mathrm{Sh}}(\boldsymbol{S})}\Big[\big(v_{\boldsymbol{xy}}(\boldsymbol{S}) - v_{\boldsymbol{xy}}(\varnothing) - \boldsymbol{S}^\top\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta)\big)^2\Big]\\ \text{s.t.}\quad & \mathbf{1}^\top\phi_{\text{fast}}(x,y;\theta) = v_{xy}([d]) - v_{xy}(\varnothing)\quad\forall\,(x,y).\end{aligned}$$

Additive efficient normalization The constraint on the explainer predictions is necessary to ensure that the global optimizer outputs the exact Shapley values, and we use the same approach as prior work to enforce this constraint (Jethani et al., 2021b). We allow the model to make unconstrained predictions that we then modify using the following

transformation:

$$\phi_{\text{fast}}(x, y; \theta) \leftarrow \phi_{\text{fast}}(x, y; \theta) + \frac{v_{xy}([d]) - v_{xy}(\varnothing) - \mathbf{1}^\top \phi_{\text{fast}}(x, y; \theta)}{d}. \tag{E.12}$$

This operation is known as the *additive efficient normalization* (Ruiz et al., 1998), and it can be interpreted as projecting the predictions onto the hyperplane where the constraint holds (Jethani et al., 2021b). We implement it as an output activation function, similar to how softmax is used to ensure valid probabilistic predictions for classification models.

Subset distribution The specific distribution $p_{\text{Sh}}(\boldsymbol{S})$ in our loss function is motivated by the Shapley value's weighted least squares characterization (Charnes et al., 1988; Lundberg and Lee, 2017). This result states that the Shapley values for a game $v : \mathcal{P}(d) \mapsto \mathbb{R}$ are the solution to the following optimization problem:

$$\begin{aligned} \min_{\phi \in \mathbb{R}^d} \quad & \sum_{0<|S|<d} \frac{d-1}{\binom{d}{|S|}(|S|)(d-|S|)} \Big(v(S) - v(\varnothing) - S^\top \phi \Big)^2 \\ \text{s.t.} \quad & \mathbf{1}^\top \phi = v([d]) - v(\varnothing). \end{aligned}$$

We obtain $p_{\text{Sh}}(\boldsymbol{S})$ by normalizing the weighting term in the summation, and doing so yields a distribution $p_{\text{Sh}}(S) \propto (|S| - 1)!(d - |S| - 1)!$ for $0 < |S| < d$ and $p_{\text{Sh}}([d]) = p_{\text{Sh}}(\varnothing) = 0$. To sample from $p_{\text{Sh}}(\boldsymbol{S})$, we calculate the probability mass on each cardinality, sample a cardinality from this multinomial distribution, and then select the indices uniformly at random.

Stochastic gradient descent As is common in deep learning, we optimize our objective using stochastic gradients rather than exact gradients. To estimate our objective, we require a set of tuples $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S})$ that we obtain as follows. First, we sample an input $x \sim p(\boldsymbol{x})$. Next, we sample multiple subsets $S \sim p_{\text{Sh}}(\boldsymbol{S})$. To reduce gradient variance, we use the paired sampling trick (Covert and Lee, 2021) and pair each subset S with its complement $\bar{S}$.

Then, we use our explainer to output Shapley values simultaneously for all classes $y \in [K]$. Finally, we minibatch this procedure across multiple inputs x and calculate our loss across the resulting set of tuples $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S})$.

Fine-tuning Rather than training the ViT explainer from scratch, we find that fine-tuning an existing model leads to better performance. This is consistent with recent work that finds ViTs challenging to train from scratch (Dosovitskiy et al., 2020). We have several options for initializing the explainer: we can use (i) the original classifier $f(\boldsymbol{x}; \eta)$, (ii) the fine-tuned classifier $g(\boldsymbol{x}_S; \beta)$, or (iii) a ViT pre-trained on another task. We treat this choice as a hyperparameter, selecting the initialization that yields the best performance. We also experiment with freezing certain layers in the model, but we find that training all the parameters leads to the best performance.

Explainer architecture We use standard ViT architectures for the explainer. These typically append a class token to the set of image tokens (Dosovitskiy et al., 2020), and we find it beneficial to preserve this token in pre-trained architectures even though it is unnecessary for Shapley value estimation. We require a separate output head from the pre-trained architecture, and our explainer head consists of one additional self-attention block followed by three fully-connected layers. Each image patch yields one Shapley value estimate per class, and we discard the results for the class token.

Hyperparameter tuning To select hyperparaters related to the learning rate, initialization and architecture, we use a pre-computed set of tuples $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{S})$ to calculate a validation loss. These are generated using inputs x that were not used for training, so our validation loss can be interpreted as an unbiased estimator of the objective function. This approach serves as an inexpensive alternative to comparing with ground truth Shapley values for a large number of samples.

Training pseudocode Algorithm 2 shows a simplified version of our training algorithm, without minibatching, sampling multiple subsets S, or parallelizing across the classes y.

Algorithm 2: Explainer training

Input: Coalitional game $v_{\boldsymbol{xy}}(\boldsymbol{S})$, learning rate α

Output: Explainer $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$

initialize $\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$

while *not converged* **do**

  sample $(x, y) \sim p(\boldsymbol{x}, \boldsymbol{y})$, $S \sim p_{\text{Sh}}(\boldsymbol{S})$

  predict $\phi \leftarrow \phi_{\text{fast}}(x, y; \theta)$

  set $\phi \leftarrow \phi + d^{-1}\left(v_{xy}([d]) - v_{xy}(\varnothing) - \mathbf{1}^\top \phi\right)$

  calculate $\mathcal{L} \leftarrow \left(v_{xy}(S) - v_{xy}(\varnothing) - S^\top \phi\right)^2$

  update $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$

end

E.3.1 Hyperparameter Choices

When training the original classifier and fine-tuned classifier models, we used a learning rate of 10^{-5} and trained for 25 epochs and 50 epochs, respectively. The MURA classifier was trained with an upweighted loss for negative examples to account for class imbalance. The best model was selected based on the validation criterion, where we used 0-1 accuracy for ImageNette and Oxford-IIIT Pets, and Cohen Kappa for MURA.

When training the explainer model, we used the same ViT-Base architecture as the original classifier and initialized using the fine-tuned classifier, as this gave the best results. We used the AdamW optimizer (Loshchilov and Hutter, 2018) with a cosine learning rate schedule and a maximum learning rate of 10^{-4}, and we trained the model for 100 epochs, selecting the best model based on the validation loss. We used standard data augmentation steps: random resized crops, vertical flips, horizontal flips, and color jittering including brightness, contrast, saturation, and hue. We used minibatches of size 64 with 32 subset samples S per x sample, and we found that using a tanh nonlinearity on the explainer

predictions was helpful to stabilize training.

Finally, we modified the ViT architecture to output Shapley values for each token and each class: we removed the classification head and added an extra attention layer, followed by three fully-connected layers with width 4 times the embedding dimension, and we fine-tuned the entire ViT backbone. These choices were determined by an ablation study with different model configurations, and we also compared with training training the ViT from scratch and training a separate U-Net explainer model (Ronneberger et al., 2015) (see Table E.1).

Table E.1: Ablation experiments for ViT Shapley explainer architecture on the ImageNette dataset, with and without fine-tuning.

Fine-tuning config.	Extra attention block	Frozen backbone	Val loss	Test loss
A	False	True	4.332	4.351
B	False	False	4.331	4.351
C	True	True	4.319	4.339
D	True	False	**4.309**	**4.318**
ViT trained from scratch			4.331	4.341
U-Net trained from scratch			4.332	4.338

We used a machine with 2 GeForce RTX 2080Ti GPUs to train the explainer model, and due to GPU memory constraints we loaded the classifier and explainer to separate GPUs and trained with mixed precision using PyTorch Lightning.[1] Training the explainer model required roughly 19 hours for the ImageNette dataset and 60 hours for the MURA dataset.

E.4 Proofs

Here, we re-state and prove our main results from Section 7.6.

Lemma E.1. *For a single input-output pair* (x, y)*, the expected loss under eq.* 7.3 *for the*

[1] `https://github.com/PyTorchLightning/pytorch-lightning`

prediction $\phi_{\text{ViT}}(x, y; \theta)$ *is* μ*-strongly convex with* $\mu = H_{d-1}^{-1}$*, where* H_{d-1} *is the* $(d-1)$*th harmonic number.*

Proof. For an input-output pair (x, y), the expected loss for the prediction $\phi = \phi_{\text{fast}}(x, y; \theta)$ under our objective is given by

$$h_{xy}(\phi) = \phi^\top \mathop{\mathbb{E}}_{p_{\text{Sh}}(\boldsymbol{S})}[\boldsymbol{S}\boldsymbol{S}^\top]\phi - 2\phi^\top \mathop{\mathbb{E}}_{p_{\text{Sh}}(\boldsymbol{S})}\Big[\boldsymbol{S}\big(v_{xy}(\boldsymbol{S}) - v_{xy}(\varnothing)\big)\Big] + \mathop{\mathbb{E}}_{p_{\text{Sh}}(\boldsymbol{S})}\Big[\big(v_{xy}(\boldsymbol{S}) - v_{xy}(\varnothing)\big)^2\Big].$$

This is a quadratic function of ϕ with its Hessian given by

$$\nabla_\phi^2 h_{xy}(\phi) = 2 \cdot \mathop{\mathbb{E}}_{p_{\text{Sh}}(\boldsymbol{S})}[\boldsymbol{S}\boldsymbol{S}^\top].$$

The convexity of $h_{xy}(\phi)$ is determined by the Hessian's eigenvalues, and its entries can be derived from the subset distribution $p_{\text{Sh}}(\boldsymbol{S})$; see similar results in Simon and Vincent (2020) and Covert and Lee (2021). The distribution assigns equal probability to subsets of equal cardinality, so we define the shorthand notation $p_k \equiv p_{\text{Sh}}(S)$ for S such that $|S| = k$. Specifically, we have

$$p_k = Q^{-1}\frac{d-1}{\binom{d}{k}k(d-k)} \qquad \text{and} \qquad Q = \sum_{k=1}^{d-1}\frac{d-1}{k(d-k)}.$$

We can then write $A \equiv \mathbb{E}_{p_{\mathrm{Sh}}(\boldsymbol{S})}[\boldsymbol{S}\boldsymbol{S}^\top]$ and derive its entries as follows:

$$\begin{aligned}
A_{ii} &= \Pr(\boldsymbol{S}_i = 1) = \sum_{k=1}^{d} \binom{d-1}{k-1} p_k \\
&= Q^{-1} \sum_{k=1}^{d-1} \frac{(d-1)}{d(d-k)} \\
&= \frac{\sum_{k=1}^{d-1} \frac{d-1}{d(d-k)}}{\sum_{k=1}^{d-1} \frac{d-1}{k(d-k)}} \\
A_{ij} &= \Pr(\boldsymbol{S}_i = \boldsymbol{S}_j = 1) = \sum_{k=2}^{d} \binom{d-2}{k-2} p_k \\
&= Q^{-1} \sum_{k=2}^{d-1} \frac{k-1}{d(d-k)} \\
&= \frac{\sum_{k=2}^{d-1} \frac{k-1}{d(d-k)}}{\sum_{k=1}^{d-1} \frac{d-1}{k(d-k)}}
\end{aligned}$$

Based on this, we can see that A has the structure $A = (b-c)I_d + c\mathbf{1}\mathbf{1}^\top$ for $b \equiv A_{ii} - A_{ij}$ and $c \equiv A_{ij}$. Following the argument by Simon and Vincent (2020), the minimum eigenvalue is then given by $\lambda_{\min}(A) = b - c$. Deriving the specific value shows that it depends on the $(d-1)$th harmonic number, H_{d-1}:

$$\begin{aligned}
\lambda_{\min}(A) &= b - c = A_{ii} - A_{ij} \\
&= \frac{\frac{1}{d} + \sum_{k=2}^{d-1} \left(\frac{d-1}{d(d-k)} - \frac{k-1}{d(d-k)} \right)}{\sum_{k=1}^{d-1} \frac{d-1}{k(d-k)}} \\
&= \frac{1}{d \sum_{k=1}^{d-1} \frac{1}{k(d-k)}} \\
&= \frac{1}{2 \sum_{k=1}^{d-1} \frac{1}{k}} \\
&= \frac{1}{2H_{d-1}}.
\end{aligned}$$

The minimum eigenvalue is therefore strictly positive, and this implies that $h_{xy}(\phi)$ is μ-

strongly convex with μ given by

$$\mu = 2 \cdot \lambda_{\min}(A) = H_{d-1}^{-1}.$$

Note that the strong convexity constant μ does not depend on (x, y) and is determined solely by the number of features d.

□

Theorem E.1. *For a model $\phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta)$ whose predictions satisfy the constraint in eq. 7.3, the objective value $\mathcal{L}(\theta)$ upper bounds the Shapley value estimation error as follows,*

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\big|\big|\phi_{\text{ViT}}(\boldsymbol{x}, \boldsymbol{y}; \theta) - \phi(v_{\boldsymbol{xy}})\big|\big|_2\Big] \leq \sqrt{2H_{d-1}\Big(\mathcal{L}(\theta) - \mathcal{L}^*\Big)},$$

where $\mathcal{L}^*$ represents the loss achieved by the exact Shapley values.

Proof. We first consider a single input-output pair (x, y) with prediction given by $\phi = \phi_{\text{fast}}(x, y; \theta)$. Rather than writing the expected loss $h_{xy}(\phi)$, we now write the Lagrangian $\mathcal{L}_{xy}(\phi, \gamma)$ to account for the linear constraint in our objective, see eq. 7.3:

$$\mathcal{L}_{xy}(\phi, \gamma) = h_{xy}(\phi) + \gamma\big(v_{xy}([d]) - v_{xy}(\varnothing) - \mathbf{1}^\top\phi\big).$$

Regardless of the Lagrange multiplier value $\gamma \in \mathbb{R}$, the Lagrangian $\mathcal{L}_{xy}(\phi, \gamma)$ is a μ-strongly convex quadratic with the same Hessian as $h_{xy}(\phi)$:

$$\nabla_\phi^2 \mathcal{L}_{xy}(\phi, \gamma) = \nabla_\phi^2 h_{xy}(\phi).$$

Strong convexity enables us to bound ϕ's distance to the global minimizer via the Lagrangian's value. First, we denote the Lagrangian's optimizer as (ϕ^*, γ^*), where ϕ^* is given

by the exact Shapley values (Charnes et al., 1988):

$$\phi^* = \phi(v_{xy}).$$

Next, we use the first-order strong convexity condition to write the following inequality:

$$\mathcal{L}_{xy}(\phi, \gamma^*) \geq \mathcal{L}_{xy}(\phi^*, \gamma^*) + (\phi - \phi^*)^\top \nabla_\phi \mathcal{L}_{xy}(\phi^*, \gamma^*) + \frac{\mu}{2}||\phi - \phi^*||^2.$$

According to the Lagrangian's KKT conditions (Boyd et al., 2004), we have the property that $\nabla_\phi \mathcal{L}_{xy}(\phi^*, \gamma^*) = 0$. The inequality therefore simplifies to

$$\mathcal{L}_{xy}(\phi, \gamma^*) \geq \mathcal{L}_{xy}(\phi^*, \gamma^*) + \frac{\mu}{2}||\phi - \phi^*||_2^2,$$

or equivalently,

$$||\phi - \phi^*||_2^2 \leq \frac{2}{\mu}\Big(\mathcal{L}_{xy}(\phi, \gamma^*) - \mathcal{L}_{xy}(\phi^*, \gamma^*)\Big).$$

If we constrain ϕ to be a feasible solution (i.e., it satisfies our objective's linear constraint), the KKT primal feasibility condition implies that the inequality further simplifies to

$$||\phi - \phi^*||_2^2 \leq \frac{2}{\mu}\Big(h_{xy}(\phi) - h_{xy}(\phi^*)\Big). \tag{E.13}$$

Now, we can consider this bound in expectation over $p(\boldsymbol{x}, \boldsymbol{y})$. First, we denote our full training loss as $\mathcal{L}(\theta)$, which is equal to

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbb{E}}_{p_{\text{Sh}}(\boldsymbol{S})} \Big[\Big(v_{\boldsymbol{xy}}(\boldsymbol{S}) - v_{\boldsymbol{xy}}(\varnothing) - \boldsymbol{S}^\top \phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)\Big)^2\Big] = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})} \Big[h_{\boldsymbol{xy}}\big(\phi_{\text{fast}}(\boldsymbol{x}, \boldsymbol{y}; \theta)\big)\Big].$$

Next, we denote $\mathcal{L}^*$ as the training loss achieved by the exact Shapley values, or

$$\mathcal{L}^* = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[h_{\boldsymbol{xy}}\big(\phi(v_{\boldsymbol{xy}})\big)\Big].$$

Given a network $\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta)$ whose predictions satisfy the linear constraint, taking the bound from eq. E.13 in expectation yields the following bound on the distance between the predicted and exact Shapley values:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\big|\big|\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta) - \phi(v_{\boldsymbol{xy}})\big|\big|_2^2\Big] \leq \frac{2}{\mu}\Big(\mathcal{L}(\theta) - \mathcal{L}^*\Big).$$

Applying Jensen's inequality to the left side, we can rewrite the bound as follows:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\big|\big|\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta) - \phi(v_{\boldsymbol{xy}})\big|\big|_2\Big] \leq \sqrt{\frac{2}{\mu}\Big(\mathcal{L}(\theta) - \mathcal{L}^*\Big)}.$$

Substituting in the strong convexity parameter μ from Lemma 7.1, we arrive at the final bound:

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\big|\big|\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta) - \phi(v_{\boldsymbol{xy}})\big|\big|_2\Big] \leq \sqrt{2H_{d-1}\Big(\mathcal{L}(\theta) - \mathcal{L}^*\Big)}.$$

□

We also present a corollary to Theorem 7.1. This result formalizes the intuition that if we can iteratively optimize the explainer such that its loss approaches the optimum, our Shapley value estimation error will go to zero.

Corollary E.1. *Given a sequence of models $\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta_1), \phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta_2), \ldots$ whose predictions satisfy the constraint in eq. 7.3 and where $\mathcal{L}(\theta_n) \to \mathcal{L}^*$, the Shapley value estimation error goes to zero:*

$$\lim_{n\to\infty} \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\big|\big|\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta_n) - \phi(v_{\boldsymbol{xy}})\big|\big|_2\Big] = 0.$$

Proof. Fix $\epsilon > 0$. By assumption, there exists a value n' such that $\mathcal{L}(\theta_n) - \mathcal{L}^* < \frac{\mu\epsilon^2}{2}$ for

$n > n'$. Following the result in Theorem 7.1, we have $\mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\left[\left|\left|\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta_n) - \phi(v_{\boldsymbol{xy}})\right|\right|_2\right] < \epsilon$ for $n > n'$.

□

Finally, we also consider the role of our loss function in quantifying the Shapley value estimation error, which we define for a given explainer model $\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta)$ as

$$\text{SVE} = \mathop{\mathbb{E}}_{p(\boldsymbol{x},\boldsymbol{y})}\left[\left|\left|\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta) - \phi(v_{\boldsymbol{xy}})\right|\right|_2\right].$$

One natural approach is to use an external dataset (e.g., the test data) consisting of samples (x^i, y^i) for $i = 1, \ldots, n$, calculate their exact Shapley values $\phi(v_{x^i y^i})$, and generate a Monte Carlo estimate as follows:

$$\hat{\text{SVE}}_n = \frac{1}{n}\sum_{i=1}^{n}\left|\left|\phi_{\text{fast}}(x^i, y^i;\theta) - \phi(v_{x^i y^i})\right|\right|_2.$$

While standard concentration inequalities allow us to bound SVE using $\hat{\text{SVE}}_n$, generating the ground truth values can be computationally costly, particularly for large n. Instead, another approach is to use our result from Theorem 7.1, which bypasses the need for ground truth Shapley values. For this, recall that $\mathcal{L}(\theta)$ represents our weighted least squares loss function, where we assume that the explainer $\phi_{\text{fast}}(\boldsymbol{x},\boldsymbol{y};\theta)$ satisfies the constraint in eq. 7.3 for all predictions. If we know $\mathcal{L}(\theta)$ exactly, then Theorem 7.1 yields the following bound with probability 1:

$$\text{SVE} \leq \sqrt{2H_{d-1}\Big(\mathcal{L}(\theta) - \mathcal{L}^*\Big)}.$$

If we do not know $\mathcal{L}(\theta)$ exactly, we can instead form a Monte Carlo estimate $\hat{\mathcal{L}}(\theta)_n$ using samples (x^i, y^i, S^i) for $i = 1, \ldots, n$. Then, using concentration inequalities like Chebyshev or Hoeffding (the latter only applies if we assume bounded errors), we can get probabilistic bounds of the form $\text{P}(|\mathcal{L}(\theta) - \hat{\mathcal{L}}_n| > \epsilon) \leq \delta$. With these, we can say with probability at least $1 - \delta$ that $\mathcal{L}(\theta) \leq \hat{\mathcal{L}}(\theta)_n + \epsilon$. Finally, combining this with the last steps of our Theorem 7.1

proof, we obtain the following bound with probability at least $1 - \delta$:

$$\text{SVE} \leq \sqrt{2H_{d-1}\Big(\hat{\mathcal{L}}(\theta)_n - \mathcal{L}^* + \epsilon\Big)}. \tag{E.14}$$

Naturally, δ is a function of ϵ and the number of samples n used to estimate $\hat{\mathcal{L}}(\theta)_n$, with the rate of convergence to probability 1 depending on the choice of concentration inequality (Chebyshev or Hoeffding). Although this procedure yields an inexpensive upper bound on the Shapley value estimation error, the bound's looseness, as well as the fact that we do not know $\mathcal{L}^*$ a priori, make it unappealing as an evaluation metric. The more important takeaways are (i) that training with the loss $\mathcal{L}(\theta)$ effectively minimizes an upper bound on the Shapley value estimation error, and (ii) that comparing explainer models via their validation loss, which is effectively $\hat{\mathcal{L}}(\theta)_n$, is a principled approach to perform model selection and hyperparameter tuning.

E.5 Datasets

The ImageNette dataset contains 9,469 training examples and 3,925 validation examples, and we split the validation data to obtain validation and test sets containing 1,962 examples each. The MURA dataset contains 36,808 training examples and 3,197 validation examples. We use the validation examples as a test set, and we split the training examples to obtain train and validation sets containing 33,071 and 3,737 examples, ensuring that images from the same patient belong to a single split. The Oxford-IIIT Pets dataset contains 7,349 examples for 37 classes, and we split the data to obtain train, validation, and test sets containing 5,879, 735, and 735 examples, respectively. For all datasets, the training and validation data were used to train the original classifiers, fine-tuned classifiers and explainer models, and the test data was used only when calculating performance metrics.

E.6 Baseline Methods

This section provides implementation details for the baseline explanation methods. We used a variety of attention-, gradient- and removal-based methods as comparisons for ViT Shapley, and we modified several approaches to arrive at patch-level feature attribution scores.

Attention last This approach calculates the attention directed from each image token into the class token in the final self-attention layer, summed across attention heads (Abnar and Zuidema, 2020; Chefer et al., 2021). The results are provided at the patch-level, but they are not generated separately for each output class.

Attention rollout This approach accounts for the flow of attention between tokens by summing across attention heads and multiplying the resulting attention matrices at each layer (Abnar and Zuidema, 2020). Like the previous method, results are not generated separately for each output class. We used an implementation provided by prior work (Chefer et al., 2021).

Common gradient-based methods Several methods that operate via input gradients are Vanilla gradients (Simonyan et al., 2013), SmoothGrad (Smilkov et al., 2017), VarGrad (Hooker et al., 2019), and IntGrad (Sundararajan et al., 2017). These methods were run using the Captum package (Kokhlikyan et al., 2020), and we used 10 samples per image for SmoothGrad, VarGrad and IntGrad. We tried applying these at the level of pixels and patch embeddings, and in both cases we created class-specific, patch-level attributions by summing across the unnecessary dimensions. We calculated the absolute value before summing for Vanilla and SmoothGrad, VarGrad automatically produces non-negative values, and we preserved the sign for IntGrad because it should be meaningful.

GradCAM Originally designed for intermediate convolutional layers (Selvaraju et al., 2017), GradCAM has since been generalized to the ViT context. The main operations

Table E.2: Performance metrics for target-class explanations with additional baselines. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	ImageNette			MURA		
	Ins. (↑)	Del. (↓)	Faith. (↑)	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.962 (0.004)	0.793 (0.013)	**0.694 (0.015)**	0.890 (0.010)	0.592 (0.013)	0.635 (0.016)
Attention rollout	0.938 (0.005)	0.880 (0.010)	**0.704 (0.015)**	0.845 (0.011)	0.692 (0.014)	0.618 (0.016)
GradCAM (LN)	0.914 (0.006)	0.937 (0.008)	0.680 (0.015)	0.899 (0.009)	0.681 (0.015)	0.631 (0.016)
GradCAM (Attn)	0.938 (0.006)	0.948 (0.006)	0.656 (0.014)	0.843 (0.012)	0.835 (0.011)	0.580 (0.016)
IntGrad (Pixel)	0.967 (0.004)	0.930 (0.008)	0.403 (0.024)	0.897 (0.010)	0.796 (0.015)	0.201 (0.022)
IntGrad (Embed.)	0.967 (0.004)	0.930 (0.008)	0.403 (0.024)	0.897 (0.010)	0.796 (0.015)	0.201 (0.022)
Vanilla (Pixel)	0.938 (0.005)	0.860 (0.011)	**0.700 (0.015)**	0.890 (0.010)	0.561 (0.014)	0.627 (0.016)
Vanilla (Embed.)	0.950 (0.004)	0.808 (0.013)	**0.703 (0.015)**	0.890 (0.010)	0.537 (0.014)	0.629 (0.016)
SmoothGrad (Pixel)	0.960 (0.005)	0.779 (0.013)	**0.706 (0.015)**	0.873 (0.010)	0.634 (0.014)	0.618 (0.016)
SmoothGrad (Embed.)	0.947 (0.005)	0.942 (0.006)	**0.703 (0.015)**	0.870 (0.010)	0.813 (0.011)	0.617 (0.016)
VarGrad (Pixel)	0.958 (0.005)	0.796 (0.013)	**0.682 (0.015)**	0.871 (0.010)	0.660 (0.013)	0.577 (0.015)
VarGrad (Embed.)	0.949 (0.005)	0.946 (0.005)	**0.700 (0.015)**	0.857 (0.011)	0.823 (0.011)	0.615 (0.016)
LRP	0.967 (0.004)	0.779 (0.014)	**0.705 (0.015)**	0.900 (0.009)	0.551 (0.013)	0.646 (0.016)
Leave-one-out	0.969 (0.002)	0.917 (0.010)	0.140 (0.040)	0.926 (0.008)	0.694 (0.017)	0.308 (0.032)
RISE	0.977 (0.001)	0.860 (0.014)	**0.704 (0.015)**	0.957 (0.004)	0.573 (0.018)	0.618 (0.016)
ViT Shapley	**0.985 (0.002)**	**0.691 (0.014)**	**0.711 (0.015)**	**0.971 (0.002)**	**0.307 (0.013)**	**0.707 (0.013)**
Random	0.951 (0.005)	0.951 (0.005)	-	0.849 (0.010)	0.847 (0.010)	-

remain the same, only the representation being analyzed is the layer-normed input to the final self-attention layer, and the aggregation is across the embedding dimension rather than convolutional channels (GradCAM LN) (Gildenblat and contributors, 2021). We also experimented with using a different internal layer for generating explanations (the attention weights computed in the final self-attention layer, denoted as GradCAM Attn. (Chefer et al., 2021)).

Layer-wise relevance propagation (LRP) Originally described as a set of constraints for a modified backpropagation routine (Bach et al., 2015), LRP has since been implemented for a variety of network layers and architectures, and it was recently adapted to ViTs (Chefer et al., 2021). We used an implementation provided by prior work (Chefer et al., 2021).

Leave-one-out The importance scores in this approach are the difference in prediction probability for the full-image and the iamge with a single patch removed. We removed patches by setting pixels to zero, similar to the original version for CNNs (Zeiler and Fergus, 2014).

RISE This approach involves sampling many occlusion masks and reporting the mean prediction when each patch is included. The original version for CNNs (Petsiuk et al., 2018) used a complex approach to generate masks, but we simply sampled subsets of patches. As in the original work, we sample from all subsets with equal probability, and we use 2,000 mask samples per sample to be explained. We occlude patches by setting pixel values to zero, similar to the original work.

Random Finally, we included a random baseline as a comparison for the insertion, deletion and ROAR metrics. These metrics only require a ranking of important patches, so we generated ten random orderings and averaged the results across these orderings.

Table E.2 shows the same metrics as Table 7.1 with additional results for alternative implementations of several baselines. For the methods based on input gradients, we experimented with generating explanations at both the pixel level and embedding level; the preferred approach depends on the method and metric, but both versions tend to underperform ViT Shapley, with the exception of faithfulness on ImageNette where the 95% confidence intervals overlap for many methods. We also experimented with two versions of GradCAM (described above) and find that the GradCAM LN implementation generally performs slightly better. In the main text, we present results only for GradCAM LN and the remaining gradient-based methods generated at the embedding level.

E.7 Metrics Details

This section provides additional details about the performance metrics used in the main text experiments (Section 7.7).

Insertion/deletion These metrics involve repeatedly making predictions while either inserting or deleting features in order of most to least important (Petsiuk et al., 2018). While the original work removed features by setting them to zero, we use the fine-tuned classifier that was trained to handle partial information. We calculated the area under the curve for individual predictions and then averaged the results across 1,000 test set examples; we used random examples for ImageNette, and only examples that were predicted to be abnormal for MURA. Table 7.1 presents results for the true class only, and Table 7.2 presents results averaged across all the remaining classes.

Sensitivity-n This metric samples feature subsets at random and calculates the correlation between the prediction with each subset and the sum of the corresponding features' attribution scores (Ancona et al., 2018). It typically considers subsets of a fixed size n, which means sampling from the following subset distribution $p_n(S)$:

$$p_n(S) = \mathbb{1}(|S| = n)\binom{d}{n}^{-1}.$$

Mathematically, the metric is defined for a model $f(\boldsymbol{x})$, an individual sample x and label y, feature attributions $\phi \in \mathbb{R}^d$ and subset size n as follows:

$$\text{Sens}(f, x, y, \phi, n) = \text{Corr}_{p_n(\boldsymbol{S})}\big(\boldsymbol{S}^\top \phi, f_y(x) - f_y(x_{[d]\setminus \boldsymbol{S}})\big).$$

Similar to insertion/deletion, we use the fine-tuned classifier to handle held-out patches and calculate the metric across 1,000 test set images. We use subset sizes ranging from 14 to 182 patches with step size 14, and we estimate the correlation for each example and subset size using 1,000 subset samples.

Faithfulness This metric is nearly identical to sensitivity-n, only it calculates the correlation across subsets of all sizes (Bhatt et al., 2021). Mathematically, it is defined as

$$\text{Faith}(f, x, y, \phi) = \text{Corr}_{p(\boldsymbol{S})}\big(\boldsymbol{S}^\top \phi, f_y(x) - f_y(x_{[d]\setminus \boldsymbol{S}})\big),$$

and we sample from a distribution with equal probability mass on all cardinalities, or

$$p(S) = \frac{1}{\binom{d}{|S|} \cdot (d+1)}.$$

We use the fine-tuned classifier to handle held-out patches, and we compute faithfulness across 1,000 test set images and with 1,000 subset samples per image.

ROAR Finally, ROAR evaluates the model's accuracy after removing features in order from most to least important (Hooker et al., 2019). We also experimented with *inserting* features in order of most to least important. Crucially, the ROAR authors propose handling held-out features by retraining the model with masked inputs. We performed masked retraining by performing test-time augmentations for all training, validation and test set images, generating explanations to identify the most important patches for the true class, and setting the corresponding pixels to zero.

Because masked retraining leaks information through the masking, we also replicated this metric using the fine-tuned classifier model, and with a separate evaluator model trained directly with random masking; the evaluator model trained with random masking has been used in prior work (Jethani et al., 2021a,b). We generated results for each number of inserted/deleted patches (1, 3, 7, 14, 28, 56, 84, 112, 140, 168, and 182) with the final accuracy computed across the entire test set.

Ground truth metrics Previous work has considered evaluations involving comparison with ground truth importance, where the ground truth is either identified by humans (Chefer et al., 2021) or introduced via synthetic dataset modifications (Zhou et al., 2022). An

important issue with such methods is that they test explanations against what a model should depend on rather than what it does depend on, so the results do not reflect the explanation's accuracy for the specific model (Petsiuk et al., 2018). We thus decided against including such metrics.

E.8 Additional Results

This section provides additional experimental results. We first show results involving similar baselines and metrics as in the main text, and we then show results comparing ViT Shapley to KernelSHAP.

E.8.1 Main Baselines and Metrics

Figure E.1 shows our evaluation of attention masking for handling held-out image patches using two separate metrics: (i) KL divergence relative to the full-image predictions (also shown in the main text), and (ii) top-1 accuracy relative to the true labels. The former can be understood as a divergence measure between the predictions with masked inputs and the predictions with patches marginalized out using their conditional distribution (see Appendix E.2). The latter is a more intuitive measure of how much the performance degrades given partial inputs. The results are similar between the two metrics, showing that the predictions diverge more quickly if the model is not fine-tuned with random masking.

Table E.3 shows insertion, deletion and faithfulness results for the MURA dataset with examples that were predicted to be normal, but while evaluating explanations for the abnormal class. ViT Shapley outperforms the baseline methods, reflecting that our explanations correctly identify patches that influence the prediction towards and against the abnormal class even for normal examples.

Table E.4 shows insertion, deletion and faithfulness results for the Pets dataset. We observe that ViT Shapley outperforms other methods for all metrics with the exception of faithfulness for target classes, where 95% confidence intervals overlap for many methods (similar to the other datasets).

Table E.5, Table E.6, and Table E.7 show insertion, deletion, and faithfulness metrics for ImageNette when using other ViT architectures (i.e., ViT-Tiny, -Small, and -Large, respectively) (Wightman, 2019; Dosovitskiy et al., 2020) for the classifier and explainer. They show results for target-class explanations and non-target-class explanations, respectively. The results are consistent with those obtained for ViT-Base, and ViT Shapley outperforms the baseline methods across all three metrics. This shows that our explainer model can be trained successfully with architectures of different sizes, including when using a relatively small number of parameters.

Table E.8 shows insertion, deletion and faithfulness results for a ViT classifier trained directly with random masking. Whereas our Section 7.7 experiments utilize a fine-tuned classifier to handle missing patches, a classifier trained with random masking allows us to bypass the fine-tuning stage and train the explainer directly. The results are similar to Table 7.1, and we find that ViT Shapley consistently achieves the best performance.

Figure E.2, Figure E.3, and Figure E.4 show the average curves used to generate the insertion/deletion AUC results. All sets of plots reflect that explanations from ViT Shapley identify relevant patches that quickly move the prediction towards or away from a given class. In the case of ImageNette and Pets, we observe that this holds for both target and non-target classes.

Figure E.5 shows the sensitivity-n metric evaluated for non-target classes on the ImageNette dataset. Similarly, these results show that ViT Shapley generates attribution scores that represent the impact of withholding features from a model, even for non-target classes. In this case, RISE and leave-one-out are more competitive with ViT Shapley, but their performance is less competitive when the correlation is calculated for subsets of all sizes (see faithfulness in Table 7.2).

Next, Figure E.6 shows ROAR results generated in both the insertion and deletion directions, using the four patch removal approaches: (i) the fine-tuned classifier, (ii) the separate evaluator model trained directly with random masking, (iii) masked retraining, and (iv) masked retraining without positional embeddings. The results show that in addition

to strong performance in the deletion direction, ViT Shapley consistently achieves the best results in the insertion direction, even in the case of masked retraining with positional embeddings.

Figure E.7 shows ROAR results for the same settings, but when using the ViT-Small architecture. We observe the same results obtained with ViT-Base. Except for the deletion direction with masked retraining and positional embeddings enabled, ViT Shapley achieves the best performance among all methods.

Finally, Table E.9 shows the time required to generate explanations using each approach. Because ViT Shapley requires a single forward pass through the explainer model, it is among the fastest approaches and is paralleled only by the attention-based methods. The gradient-based methods require forward and backward passes for all classes, and sometimes for many altered inputs (e.g., with noise injected for SmoothGrad). RISE is the slowest of all the approaches tested because it requires making several thousand predictions to explain each sample. Our evaluation was conducted on a GeForce RTX 2080 Ti GPU, with minibatches of 16 samples for attention last, attention rollout and ViT Shapley; batch size of 1 for Vanilla Gradients, GradCAM, LRP, leave-one-out and RISE; and internal minibatching for SmoothGrad, IntGrad and VarGrad (implemented via Captum (Kokhlikyan et al., 2020)).

ViT Shapley is the only method considered here to require training time, and as described in Appendix E.3, training the explainer models required roughly 0.8 days for ImageNette and 2.5 days for MURA. The training time is not insignificant, but investing time in training the explainer is worthwhile if (i) high-quality explanations are required, (ii) there are many examples to be explained (e.g., an entire dataset), or (iii) fast explanations are required during a model's deployment.

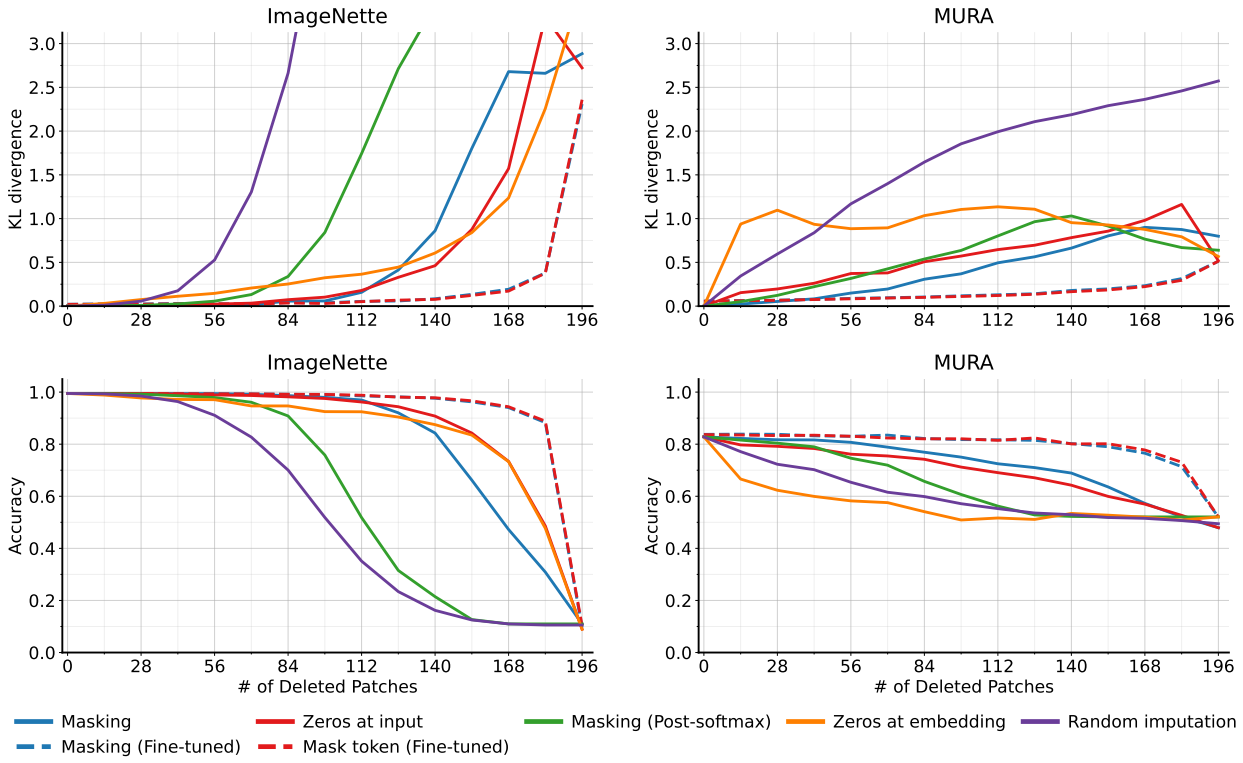


Figure E.1: ViT predictions given partial information. We delete patches at random using several removal mechanisms, and we then measure the quality of the resulting predictions via two metrics: the KL divergence relative to the original, full-image predictions (top), and the top-1 accuracy relative to the true labels (bottom).

Table E.3: MURA non-target metrics for images that were predicted to be normal. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	MURA		
	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.157 (0.011)	0.210 (0.009)	-0.455 (0.022)
Attention rollout	0.199 (0.011)	0.169 (0.010)	-0.480 (0.023)
GradCAM	0.197 (0.012)	0.152 (0.010)	-0.449 (0.022)
IntGrad	0.209 (0.014)	0.151 (0.010)	0.103 (0.023)
Vanilla	0.174 (0.011)	0.197 (0.009)	-0.477 (0.023)
SmoothGrad	0.169 (0.011)	0.176 (0.011)	-0.480 (0.023)
VarGrad	0.166 (0.011)	0.175 (0.011)	-0.478 (0.023)
LRP	0.169 (0.012)	0.200 (0.009)	-0.461 (0.023)
Leave-one-out	0.326 (0.016)	0.093 (0.007)	0.272 (0.029)
RISE	0.406 (0.018)	0.075 (0.006)	-0.479 (0.023)
ViT Shapley	**0.580 (0.016)**	**0.039 (0.003)**	**0.642 (0.014)**
Random	0.169 (0.011)	0.170 (0.011)	-

Table E.4: Performance metrics for ViT-Base on Pets. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	Target			Non-Target		
	Ins. (↑)	Del. (↓)	Faith. (↑)	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.876 (0.011)	0.494 (0.016)	0.545 (0.017)	-	-	-
Attention rollout	0.837 (0.012)	0.672 (0.017)	**0.559 (0.017)**	-	-	-
GradCAM (Attn)	0.837 (0.013)	0.737 (0.018)	0.527 (0.016)	0.005 (0.000)	0.008 (0.000)	-0.499 (0.013)
GradCAM (LN)	0.837 (0.011)	0.717 (0.019)	0.553 (0.017)	0.012 (0.001)	0.004 (0.000)	-0.511 (0.014)
IntGrad (Pixel)	0.899 (0.010)	0.802 (0.016)	0.437 (0.017)	0.006 (0.001)	0.003 (0.000)	0.126 (0.008)
IntGrad (Embed.)	0.899 (0.010)	0.802 (0.016)	0.437 (0.017)	0.006 (0.001)	0.003 (0.000)	0.126 (0.008)
Vanilla (Pixel)	0.862 (0.011)	0.596 (0.018)	**0.559 (0.017)**	0.004 (0.000)	0.011 (0.000)	-0.526 (0.014)
Vanilla (Embed.)	0.876 (0.011)	0.519 (0.017)	**0.561 (0.017)**	0.004 (0.000)	0.013 (0.000)	-0.529 (0.014)
SmoothGrad (Pixel)	0.889 (0.011)	0.488 (0.016)	**0.564 (0.017)**	0.003 (0.000)	0.014 (0.000)	-0.532 (0.014)
SmoothGrad (Embed.)	0.840 (0.013)	0.834 (0.013)	**0.558 (0.017)**	0.004 (0.000)	0.005 (0.000)	-0.526 (0.014)
VarGrad (Pixel)	0.891 (0.011)	0.513 (0.016)	**0.557 (0.017)**	0.003 (0.000)	0.013 (0.000)	-0.522 (0.014)
VarGrad (Embed.)	0.847 (0.012)	0.835 (0.013)	0.555 (0.017)	0.004 (0.000)	0.005 (0.000)	-0.523 (0.014)
LRP	0.890 (0.011)	0.466 (0.016)	**0.567 (0.017)**	0.004 (0.000)	0.013 (0.000)	-0.529 (0.014)
Leave-one-out	**0.936 (0.006)**	0.653 (0.022)	0.215 (0.036)	0.018 (0.001)	0.001 (0.000)	0.138 (0.023)
RISE	**0.946 (0.005)**	0.505 (0.022)	**0.559 (0.017)**	0.032 (0.002)	0.001 (0.000)	-0.523 (0.014)
ViT Shapley	**0.945 (0.006)**	**0.388 (0.017)**	**0.590 (0.016)**	**0.052 (0.002)**	**0.001 (0.000)**	**0.529 (0.012)**
Random	0.848 (0.012)	0.845 (0.012)	-	0.004 (0.000)	0.004 (0.000)	-

Table E.5: Performance metrics for ViT-Tiny on ImageNette. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	Target			Non-Target		
	Ins. (↑)	Del. (↓)	Faith. (↑)	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.917 (0.008)	0.702 (0.015)	0.649 (0.014)	-	-	-
Attention rollout	0.901 (0.009)	0.698 (0.015)	0.658 (0.015)	-	-	-
GradCAM (LN)	0.918 (0.007)	0.809 (0.014)	0.659 (0.014)	0.034 (0.002)	0.006 (0.000)	-0.616 (0.013)
GradCAM (Attn)	0.898 (0.009)	0.814 (0.013)	0.636 (0.014)	0.011 (0.001)	0.020 (0.001)	-0.607 (0.012)
IntGrad (Pixel)	0.928 (0.006)	0.852 (0.013)	0.293 (0.025)	0.018 (0.002)	0.008 (0.001)	0.094 (0.014)
IntGrad (Embed.)	0.928 (0.006)	0.852 (0.013)	0.293 (0.025)	0.018 (0.002)	0.008 (0.001)	0.094 (0.014)
Vanilla (Pixel)	0.884 (0.009)	0.801 (0.013)	0.656 (0.015)	0.015 (0.001)	0.020 (0.001)	-0.629 (0.013)
Vanilla (Embed.)	0.902 (0.008)	0.752 (0.015)	0.659 (0.015)	0.012 (0.001)	0.025 (0.001)	-0.632 (0.013)
SmoothGrad (Pixel)	0.908 (0.009)	0.733 (0.015)	0.659 (0.015)	0.011 (0.001)	0.029 (0.002)	-0.632 (0.013)
SmoothGrad (Embed.)	0.917 (0.008)	0.756 (0.014)	0.659 (0.015)	0.010 (0.001)	0.025 (0.002)	-0.632 (0.013)
VarGrad (Pixel)	0.914 (0.008)	0.754 (0.014)	0.648 (0.014)	0.011 (0.001)	0.027 (0.002)	-0.620 (0.012)
VarGrad (Embed.)	0.921 (0.008)	0.777 (0.014)	0.648 (0.014)	0.010 (0.001)	0.023 (0.001)	-0.618 (0.012)
LRP	0.938 (0.007)	0.648 (0.016)	**0.673 (0.014)**	0.014 (0.001)	0.024 (0.001)	-0.623 (0.013)
Leave-one-out	0.956 (0.004)	0.737 (0.019)	0.221 (0.039)	0.052 (0.004)	0.003 (0.000)	0.145 (0.025)
RISE	0.970 (0.003)	0.602 (0.019)	0.658 (0.015)	0.092 (0.005)	0.002 (0.000)	-0.628 (0.013)
ViT Shapley	**0.981 (0.002)**	**0.457 (0.015)**	**0.694 (0.014)**	**0.198 (0.006)**	**0.001 (0.000)**	**0.641 (0.011)**
Random	0.908 (0.008)	0.907 (0.008)	-	0.010 (0.001)	0.010 (0.001)	-

Table E.6: Performance metrics for ViT-Small on ImageNette. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	Target			Non-Target		
	Ins. (↑)	Del. (↓)	Faith. (↑)	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.949 (0.006)	0.706 (0.014)	**0.781 (0.011)**	-	-	-
Attention rollout	0.914 (0.007)	0.814 (0.013)	**0.787 (0.011)**	-	-	-
GradCAM (LN)	0.908 (0.006)	0.898 (0.011)	0.770 (0.010)	0.027 (0.002)	0.006 (0.000)	-0.740 (0.009)
GradCAM (Attn.)	0.904 (0.009)	0.911 (0.009)	0.724 (0.010)	0.008 (0.001)	0.016 (0.001)	-0.717 (0.009)
IntGrad (Pixel)	0.951 (0.006)	0.908 (0.010)	0.541 (0.022)	0.011 (0.001)	0.005 (0.001)	0.384 (0.018)
IntGrad (Embed.)	0.951 (0.006)	0.908 (0.010)	0.541 (0.022)	0.011 (0.001)	0.005 (0.001)	0.384 (0.018)
Vanilla (Pixel)	0.905 (0.007)	0.830 (0.012)	**0.784 (0.010)**	0.011 (0.001)	0.018 (0.001)	-0.754 (0.009)
Vanilla (Embed.)	0.921 (0.007)	0.793 (0.013)	**0.786 (0.011)**	0.009 (0.001)	0.022 (0.001)	-0.756 (0.009)
SmoothGrad (Pixel)	0.943 (0.006)	0.744 (0.014)	**0.786 (0.011)**	0.007 (0.001)	0.027 (0.001)	-0.756 (0.009)
SmoothGrad (Embed.)	0.946 (0.006)	0.753 (0.014)	**0.787 (0.011)**	0.006 (0.001)	0.026 (0.001)	-0.757 (0.009)
VarGrad (Pixel)	0.946 (0.006)	0.766 (0.013)	0.742 (0.010)	0.007 (0.001)	0.025 (0.001)	-0.709 (0.009)
VarGrad (Embed.)	0.947 (0.006)	0.774 (0.013)	0.758 (0.010)	0.007 (0.001)	0.024 (0.001)	-0.723 (0.009)
LRP	0.957 (0.005)	0.695 (0.015)	**0.793 (0.010)**	0.007 (0.001)	0.027 (0.001)	-0.753 (0.009)
Leave-one-out	0.967 (0.002)	0.855 (0.015)	0.044 (0.042)	0.024 (0.002)	0.003 (0.000)	0.079 (0.030)
RISE	0.976 (0.002)	0.752 (0.018)	**0.787 (0.010)**	0.049 (0.004)	0.002 (0.000)	-0.755 (0.009)
ViT Shapley	**0.982 (0.002)**	**0.591 (0.015)**	**0.801 (0.010)**	**0.130 (0.005)**	**0.001 (0.000)**	**0.747 (0.009)**
Random	0.933 (0.006)	0.934 (0.006)	-	0.007 (0.001)	0.007 (0.001)	-

Table E.7: Performance metrics for ViT-Large on ImageNette. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	Target			Non-Target		
	Ins. (↑)	Del. (↓)	Faith. (↑)	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.915 (0.005)	0.896 (0.008)	0.401 (0.022)	-	-	-
Attention rollout	0.929 (0.005)	0.928 (0.007)	**0.423 (0.023)**	-	-	-
GradCAM (LN)	0.955 (0.004)	0.924 (0.009)	**0.424 (0.023)**	0.012 (0.001)	0.005 (0.000)	-0.372 (0.020)
GradCAM (Attn)	0.935 (0.005)	0.930 (0.007)	0.395 (0.022)	0.007 (0.001)	0.006 (0.001)	-0.351 (0.018)
IntGrad (Pixel)	0.965 (0.003)	0.947 (0.006)	0.195 (0.020)	0.006 (0.001)	0.004 (0.000)	0.114 (0.013)
IntGrad (Embed.)	0.965 (0.003)	0.948 (0.006)	0.198 (0.020)	0.006 (0.001)	0.004 (0.000)	0.117 (0.013)
Vanilla (Pixel)	0.898 (0.007)	0.921 (0.007)	**0.411 (0.022)**	0.012 (0.001)	0.008 (0.001)	-0.370 (0.020)
Vanilla (Embed.)	0.911 (0.006)	0.905 (0.009)	**0.415 (0.023)**	0.010 (0.001)	0.010 (0.001)	-0.373 (0.020)
SmoothGrad (Pixel)	0.949 (0.004)	0.861 (0.011)	**0.424 (0.023)**	0.006 (0.001)	0.014 (0.001)	-0.382 (0.020)
SmoothGrad (Embed.)	0.956 (0.004)	0.857 (0.011)	**0.423 (0.023)**	0.005 (0.000)	0.015 (0.001)	-0.380 (0.020)
VarGrad (Pixel)	0.942 (0.005)	0.878 (0.010)	0.387 (0.021)	0.007 (0.001)	0.013 (0.001)	-0.350 (0.018)
VarGrad (Embed.)	0.944 (0.005)	0.875 (0.010)	0.383 (0.020)	0.007 (0.001)	0.013 (0.001)	-0.343 (0.017)
LRP	0.946 (0.004)	0.869 (0.010)	**0.424 (0.023)**	0.006 (0.001)	0.013 (0.001)	-0.381 (0.020)
Leave-one-out	0.970 (0.003)	0.938 (0.007)	0.123 (0.031)	0.009 (0.001)	0.003 (0.000)	0.118 (0.020)
RISE	0.978 (0.002)	0.882 (0.013)	**0.425 (0.023)**	0.019 (0.003)	0.002 (0.000)	-0.381 (0.020)
ViT Shapley	**0.986 (0.002)**	**0.742 (0.013)**	**0.439 (0.023)**	**0.077 (0.003)**	**0.001 (0.000)**	**0.391 (0.019)**
Random	0.957 (0.004)	0.956 (0.004)	-	0.005 (0.000)	0.005 (0.000)	-

Table E.8: Performance metrics for target classes explanations for a ViT classifier trained with random masking. Methods that fail to outperform the random baseline are shown in gray, and the best results are shown in bold (accounting for 95% confidence intervals).

	ImageNette		
	Ins. (↑)	Del. (↓)	Faith. (↑)
Attention last	0.963 (0.003)	0.839 (0.011)	**0.593 (0.017)**
Attention rollout	0.945 (0.004)	0.903 (0.008)	**0.597 (0.017)**
GradCAM (LN)	0.955 (0.003)	0.898 (0.011)	**0.585 (0.016)**
GradCAM (Attn)	0.945 (0.005)	0.942 (0.006)	0.564 (0.016)
IntGrad (Pixel)	0.977 (0.002)	0.933 (0.008)	0.474 (0.017)
IntGrad (Embed.)	0.977 (0.002)	0.933 (0.008)	0.474 (0.017)
Vanilla (Pixel)	0.934 (0.005)	0.901 (0.009)	**0.593 (0.017)**
Vanilla (Embed.)	0.948 (0.004)	0.864 (0.011)	**0.595 (0.017)**
SmoothGrad (Pixel)	0.966 (0.004)	0.809 (0.012)	**0.599 (0.017)**
SmoothGrad (Embed.)	0.954 (0.004)	0.950 (0.005)	**0.597 (0.017)**
VarGrad (Pixel)	0.964 (0.004)	0.826 (0.011)	**0.573 (0.016)**
VarGrad (Embed.)	0.953 (0.005)	0.954 (0.004)	**0.594 (0.017)**
LRP	0.972 (0.002)	0.806 (0.013)	**0.603 (0.017)**
Leave-one-out	0.977 (0.001)	0.881 (0.014)	0.134 (0.034)
RISE	0.982 (0.001)	0.793 (0.017)	**0.599 (0.017)**
ViT Shapley	**0.988 (0.001)**	**0.736 (0.013)**	**0.605 (0.017)**
Random	0.957 (0.004)	0.957 (0.004)	-

Table E.9: Time to generate explanations for a single sample (in milliseconds). For class-specific explanations, the running time involves generating explanations for all classes.

	ImageNette	MURA	Pets
Attention last	6.8	6.4	6.2
Attention rollout	10.6	10.0	11.1
GradCAM	275.9	26.2	986.4
IntGrad	1236.8	123.3	5004.9
Vanilla	230.1	23.0	810.7
SmoothGrad	1218.0	121.2	4854.8
VarGrad	1218.9	121.3	4882.5
LRP	1551.4	155.7	5583.5
Leave-one-out	810.0	815.9	922.5
RISE	8213.4	8171.1	8919.8
ViT Shapley	10.1	10.2	10.8

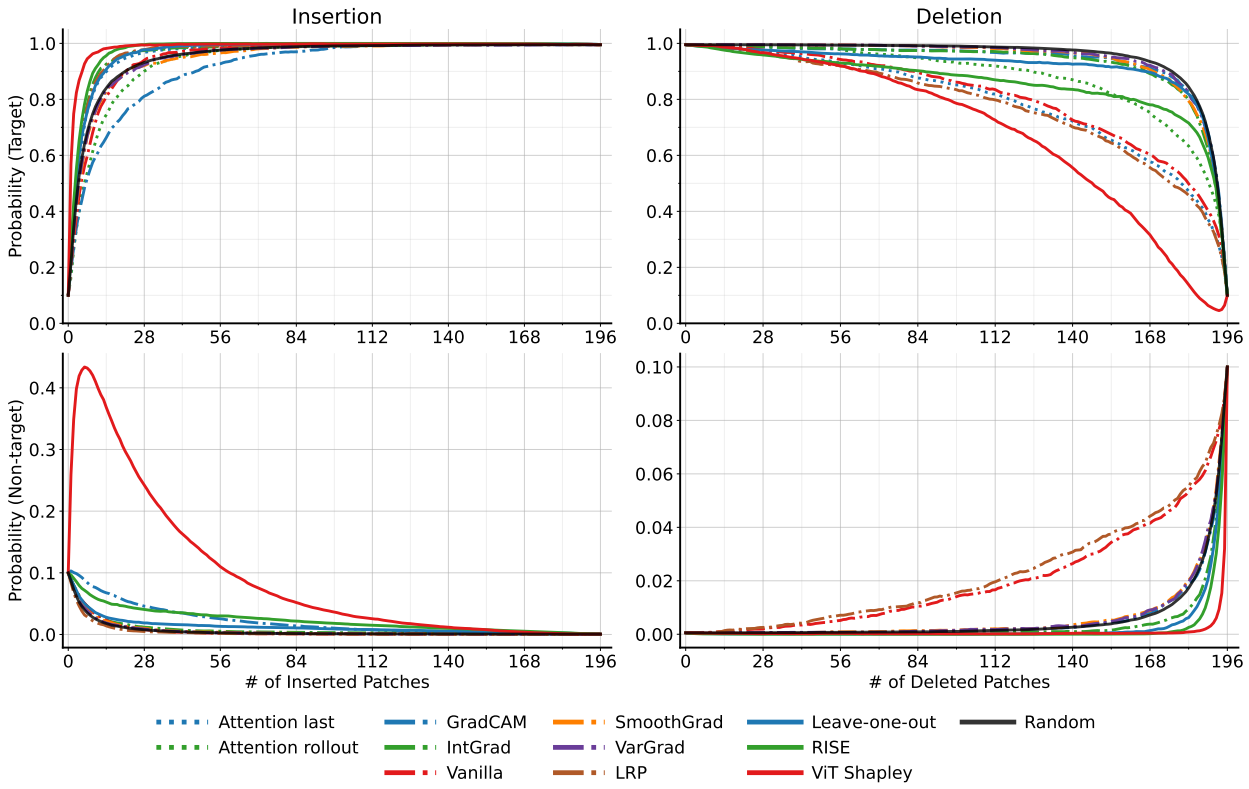


Figure E.2: ImageNette average insertion/deletion curves. Top: the mean prediction probability for the target class as features are inserted or deleted in order of most to least important. Bottom: the mean prediction probability, averaged across all non-target classes.

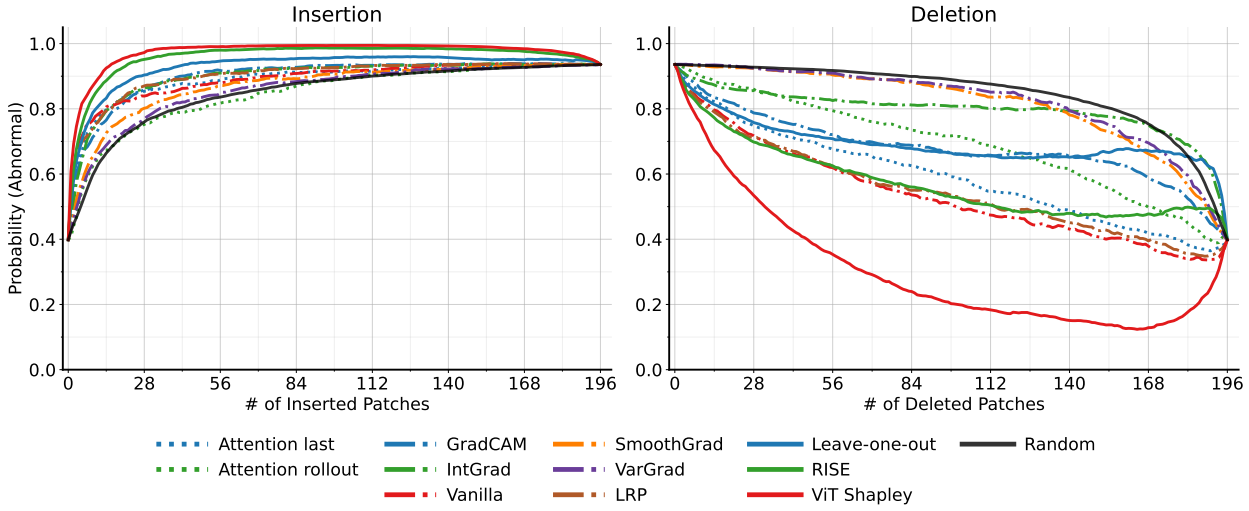


Figure E.3: MURA average insertion/deletion curves. The mean prediction probability for the target class is plotted as features are inserted or deleted in order of most to least important.

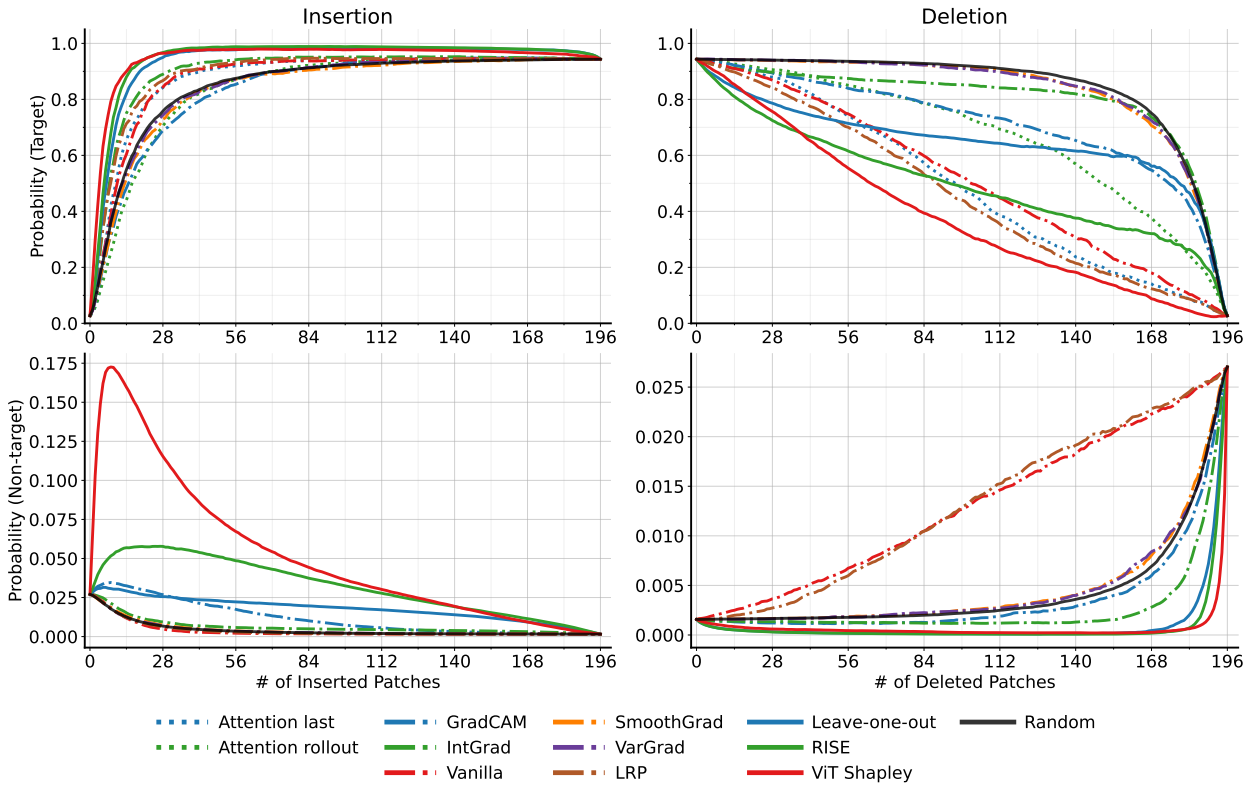


Figure E.4: Pets average insertion/deletion curves. Top: the mean prediction probability for the target class as features are inserted or deleted in order of most to least important. Bottom: the mean prediction probability, averaged across all non-target classes.

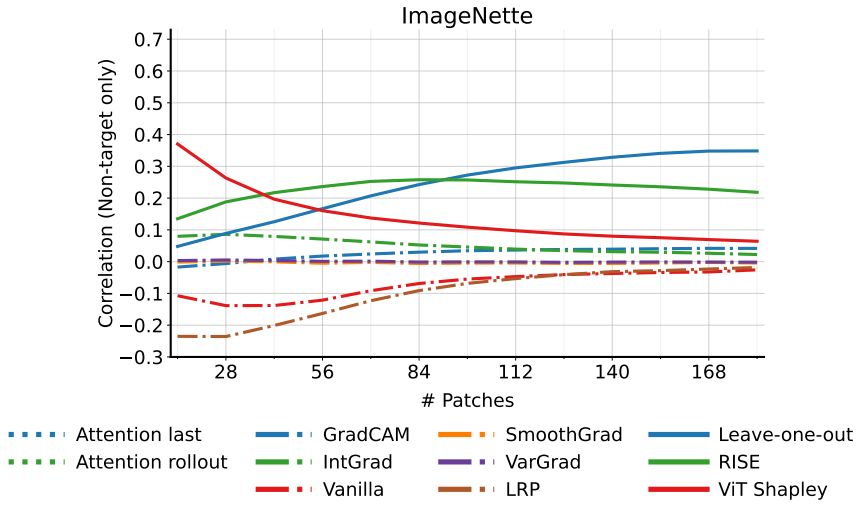


Figure E.5: Sensitivity-n evaluation for non-target classes. The results are generated separately for each subset size and averaged across all non-target classes.

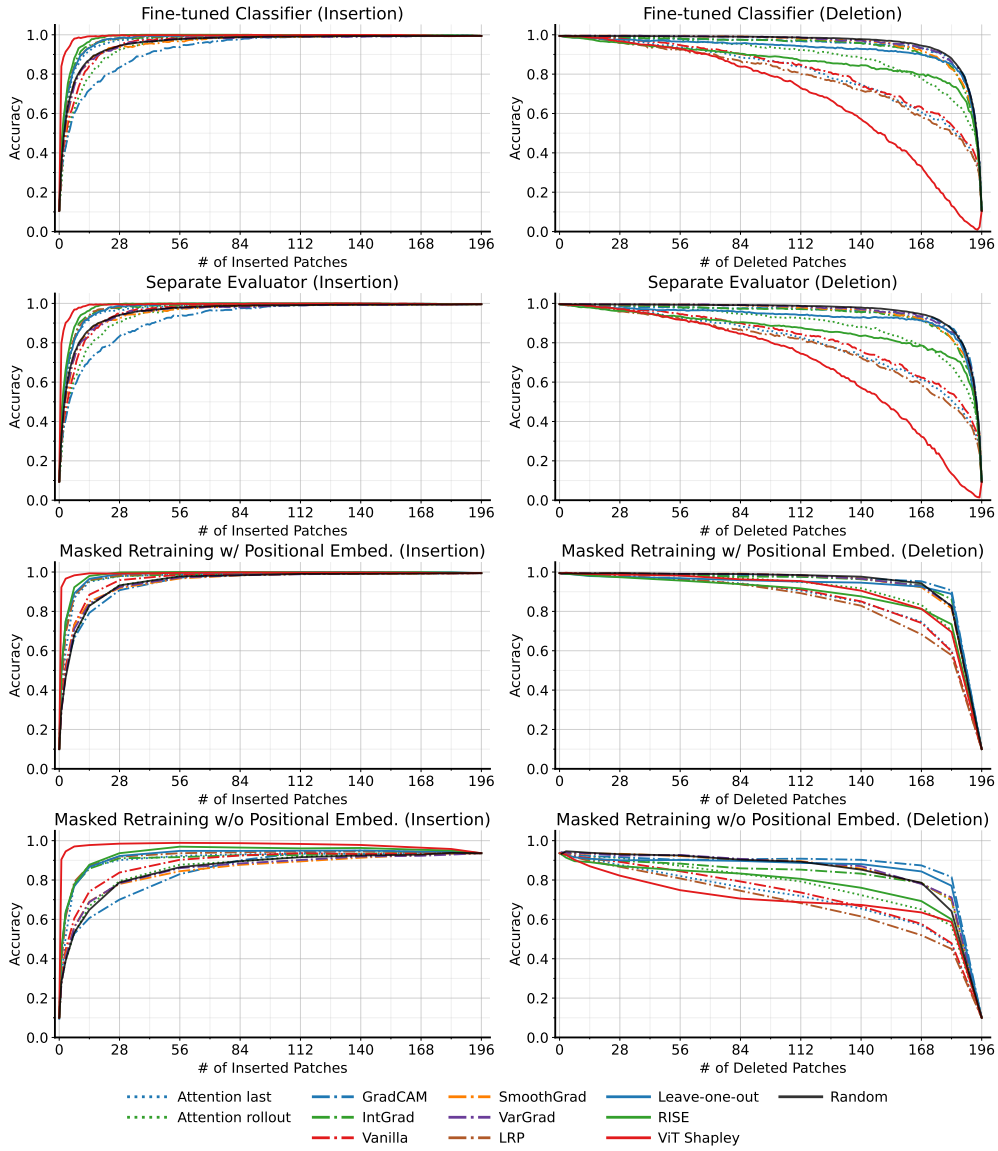


Figure E.6: ImageNette accuracy when removing features in order of their importance, run with four evaluation strategies (fine-tuned classifier, separate evaluator, masked retraining and masked retraining without positional embeddings) on the ViT-Base architecture. Left: inserting features from most to least important. Right: removing features from most to least important.

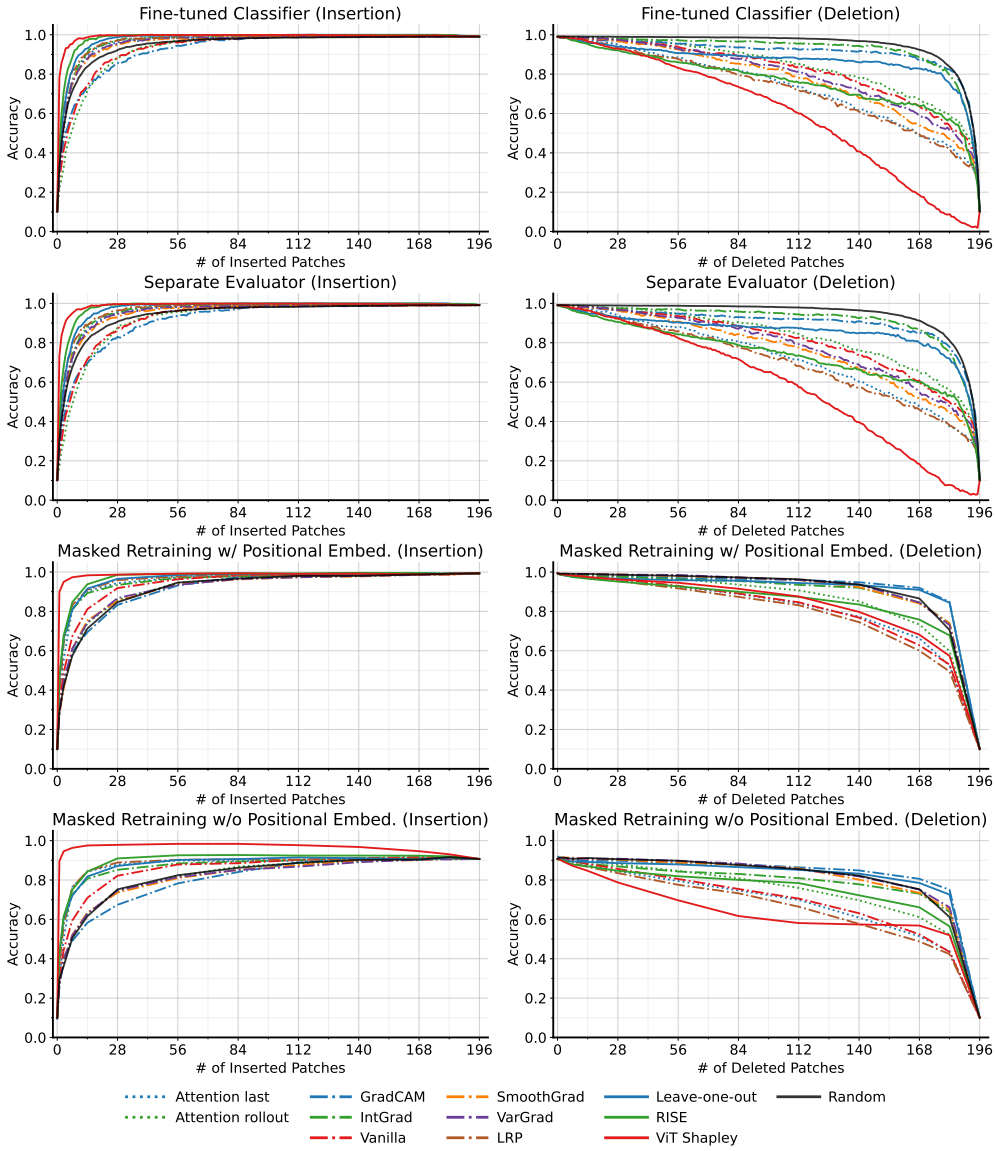


Figure E.7: ImageNette accuracy when removing features in order of their importance, run with four evaluation strategies (fine-tuned classifier, separate evaluator, masked retraining and masked retraining without positional embeddings) on the ViT-Small architecture. Left: inserting features from most to least important. Right: removing features from most to least important.

E.8.2 KernelSHAP Comparisons

Here, we provide two results comparing ViT Shapley with KernelSHAP.

First, Figure E.8 compares the approximation quality of Shapley value estimates produced by ViT Shapley and KernelSHAP. The estimates are evaluated in terms of ℓ_2 distance, Pearson correlation and Spearman (rank) correlation, and our ground truth is generated by running KernelSHAP for a large number of iterations. Specifically, we use the convergence detection approach described by Covert and Lee (2021) with a threshold of $t = 0.1$. The results are computed using just 100 randomly selected ImageNette images due to the significant computational cost.

Based on Figure E.8, we observe that the original version of KernelSHAP takes roughly 120,000 model evaluations to reach the accuracy that ViT Shapley reaches with a single model evaluation. KernelSHAP with paired sampling (Covert and Lee, 2021) converges faster, and it requires roughly 40,000 model evaluations on average. ViT Shapley's estimates are not perfect, but they reach nearly 0.8 correlation with the ground truth for the target class, and nearly 0.7 correlation on average across non-target classes.

Next, Table E.10 compares the ViT Shapley estimates and the fully converged estimates from KernelSHAP via the insertion and deletion metrics. The fully converged KernelSHAP estimates performed better than ViT Shapley on both metrics, and the gap is largest for deletion with the target class. The results were also computed using only 100 ImageNette examples due to the computational cost. These results reflect that there is room for further improvement if ViT Shapley's estimates can be made more accurate. KernelSHAP itself is not a viable option in practice, as we found that its estimates took between 30 minutes and 2 hours to converge when using paired sampling (Covert and Lee, 2021) (equivalent to roughly 300k and 1,200k model evaluations), but it represents an upper bound on how well ViT Shapley could perform with near-perfect estimation quality.

Table E.10: Comparing the quality of Shapley value estimates obtained using ViT Shapley and KernelSHAP via insertion/deletion scores.

	Target		Non-target	
	Ins. (↑)	Del. (↓)	Ins. (↑)	Del. (↓)
ViT Shapley	0.985 (0.002)	0.691 (0.014)	0.093 (0.004)	0.001 (0.000)
KernelSHAP	0.990 (0.002)	0.589 (0.053)	0.158 (0.021)	0.001 (0.000)

E.9 Qualitative Examples

To conserve space, we refer readers to the original work for additional qualitative examples (Covert et al., 2022). These results include a qualitative comparison with KernelSHAP, baseline comparisons across all three datasets (ImageNette, MURA, Pets), and examples of ViT Shapley's class-specific explanations.

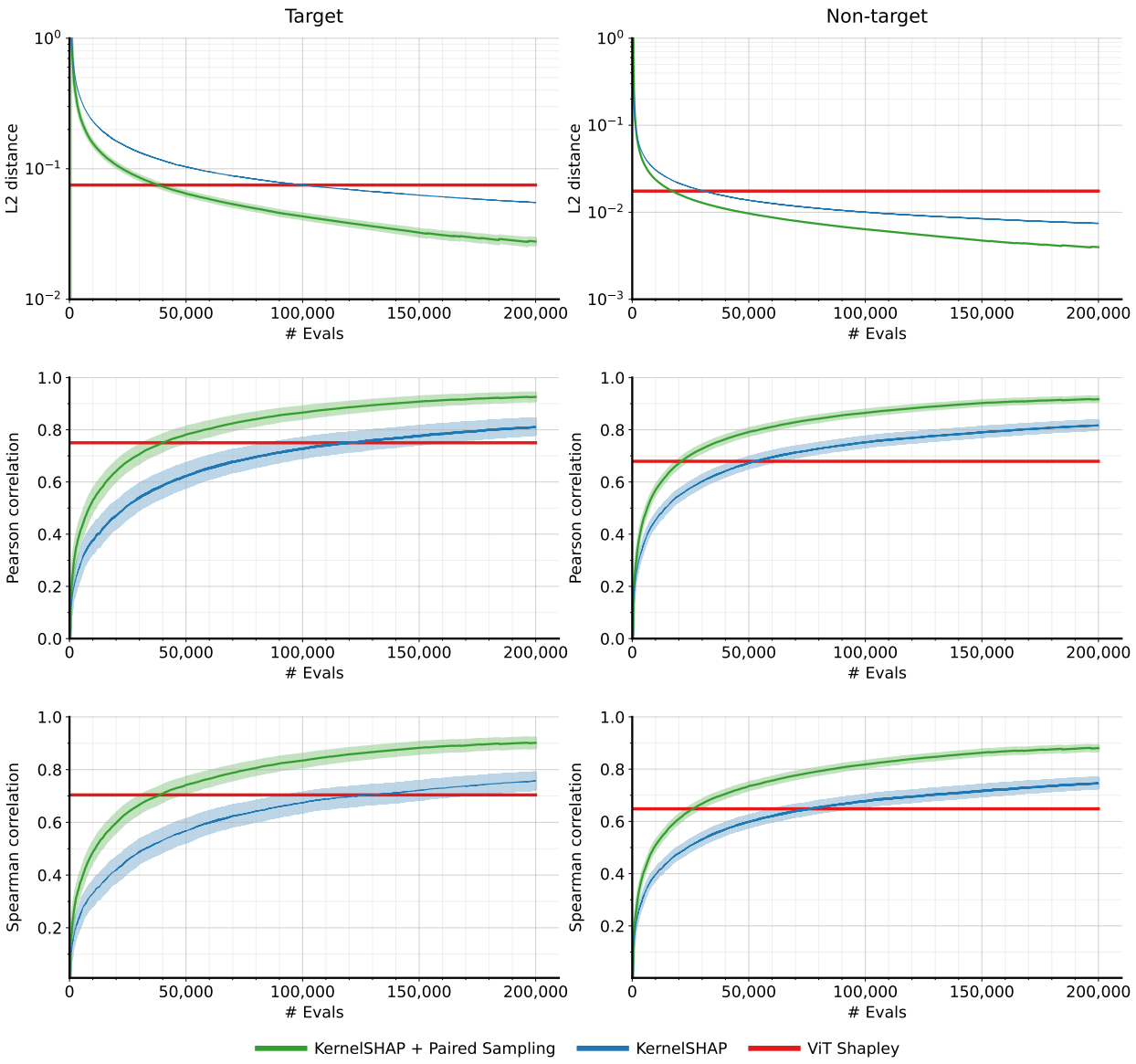


Figure E.8: Comparing the quality of Shapley value estimates obtained by ViT Shapley and KernelSHAP. The shaded areas represent 95% confidence intervals.

Appendix F

GENE SELECTION FOR SPATIAL TRANSCRIPTOMICS

This appendix section provides several additional results for evaluating PERSIST. First, we present results for the expression profile reconstruction metrics using the supervised methods, scGeneFit, SMaSH and MutInfo, which leverage cell type labels when selecting genes. Figure F.1 shows their explained variance and expressed gene prediction accuracy for the SSv4 and 10X datasets. MutInfo is competitive for small gene panels, but none of the three methods matches PERSIST's performance.

Next, we examine whether introducing binarization into the baseline methods enables better performance on our evaluation metrics, which simulate the use of gene panels in a FISH study. Figure F.2 shows that binarization makes scGeneFit, SMaSH and GeneBasis more competitive with PERSIST, and that they in some cases even match PERSIST's performance. In contrast, binarization makes Seurat and Cell Ranger perform worse, which is unsurprising because variance and dispersion values are significantly less informative for binary data. Overall, even when all the gene selection methods used binarized data, PERSIST is either the best or tied for best on each metric, suggesting better transferability to FISH studies.

To give a more granular view of PERSIST's expressed gene prediction accuracy, we provide additional plots with gene-level accuracy metrics. As in the main text, we calculate accuracy according to how often the prediction agrees with each gene's detection in the data, but we now report results for each individual gene in the SSv4 dataset. Figure F.3 shows results for the PERSIST panels containing 32 and 128 genes, with the per-gene accuracy plotted against the portion of cells in which the gene is expressed, and against the average expression level (after log-CPM normalization). The results show that the most difficult

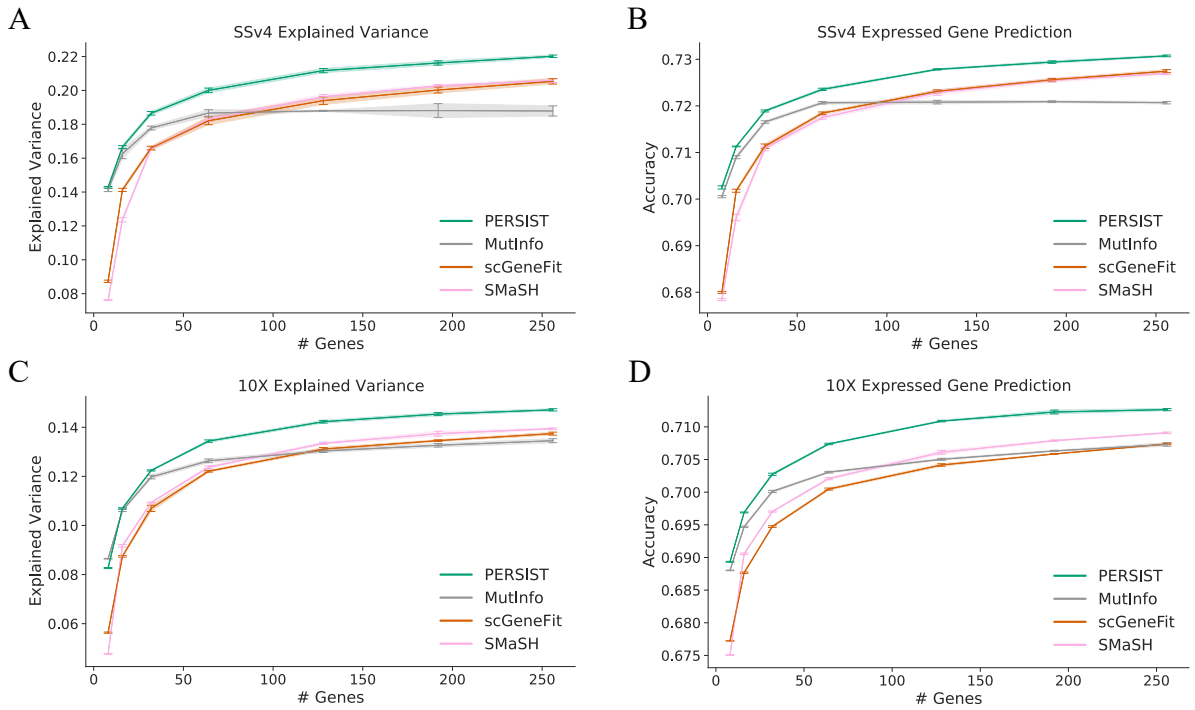


Figure F.1: Expression profile reconstruction with supervised selection methods. We compare PERSIST with the two supervised methods that leverage cell type labels, scGeneFit and MutInfo, on the tasks of predicting log-normalized expression counts (**A**, **C**) and whether each gene is expressed (**B**, **D**). **A**, Explained variance for gene panels with the SSv4 dataset. **B**, Expressed gene prediction accuracy for the SSv4 dataset. **C**, Explained variance for gene panels with the 10X dataset. **D**, Expressed gene prediction accuracy for the 10X dataset. All error bars represent 95% confidence intervals determined by training with five bootstrapped datasets.

genes to predict are those which are neither ubiquitously expressed or not expressed, and which have moderate mean expression. There is a visible improvement when we use a panel of 128 rather than 32 genes, but even then, the prediction problem remains difficult due to stochasticity in gene expression and detection.

In the main text, the expressed gene prediction results focus on genes that are expressed in 20−80% of cells, ignoring the remaining genes whose expression is easiest to predict. The results with different ranges of 10−90% and 0−100% are shown in Figure F.4. The trend between methods does not depend on the cutoff, but including the easier genes increases the

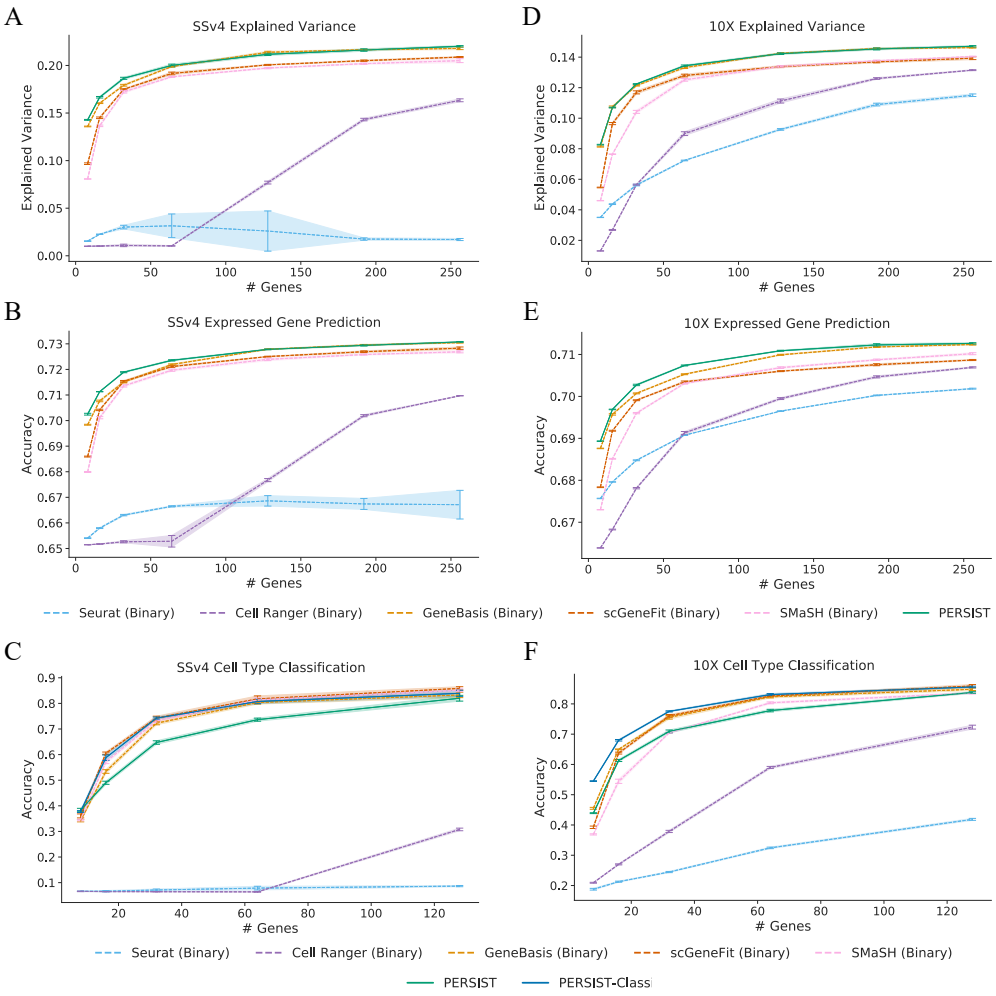


Figure F.2: Binarizing gene expression for existing methods. Binarizing gene expression levels before selecting the gene panel enable certain baselines to perform better in our metrics. GeneBasis, SMaSH and scGeneFit become more competitive with PERSIST and PERSIST-Classification, while Seurat and Cell Ranger perform worse. **A**, Explained variance for gene panels with the SSv4 dataset. **B**, Expressed gene prediction accuracy for the SSv4 dataset. **C**, Cell type classification accuracy for the SSv4 dataset. **D**, Explained variance for the 10X dataset. **E**, Expressed gene prediction for the 10X dataset. **F**, Cell type classification accuracy for the 10X dataset. All error bars represent 95% confidence intervals determined by training with five bootstrapped datasets.

overall accuracy while shrinking the gap between methods.

Regarding cell type classification, the main text reports accuracy results averaged across

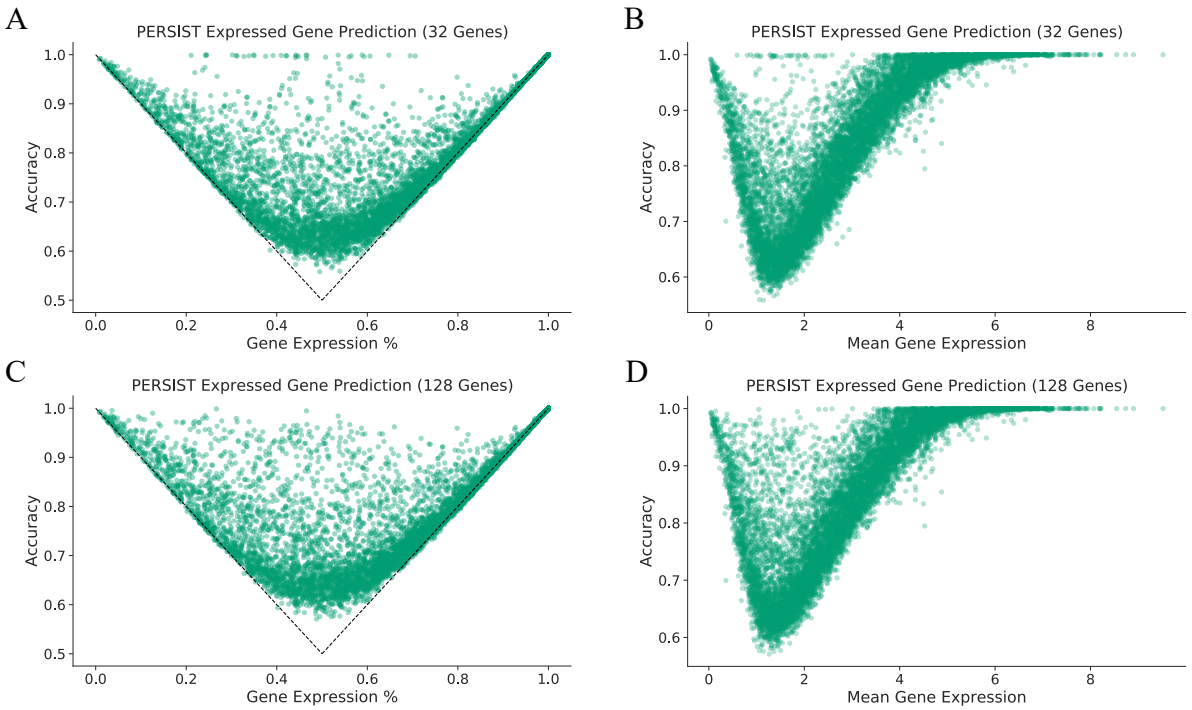


Figure F.3: PERSIST expressed gene prediction. The accuracy when predicting each gene's expression is plotted against the gene's characteristics. Genes that are either frequently expressed or infrequently expressed are easiest to predict (**A**, **C**), and those with moderate mean expression are most difficult to predict (**B**, **D**). **A**, Prediction accuracy versus the percentage of cells in which the gene is expressed, for the PERSIST panel containing 32 genes. Dotted lines indicate the accuracy achieved by trivially predicting that each gene is either always expressed or never expressed. **B**, Prediction accuracy versus mean expression level, for the 32-gene PERSIST panel. **C**, Prediction accuracy versus the percentage of cells in which the gene is expressed for the 128-gene PERSIST panel. **D**, Prediction accuracy versus mean expression level for the 128-gene PERSIST panel.

all cell types, so we now provide additional results that give a more granular view. In Figure F.5, we present two results for the PERSIST panel containing 32 genes from the SSv4 dataset: a confusion matrix showing how cells of each type are classified, and the recall (true positive rate) shown for each cell type. The results show that while some cell types are mostly correctly classified, some have relatively low recall, reflecting the imperfect overall accuracy (63%). In Figure F.6, we show the same results but for the PERSIST-Classification panel

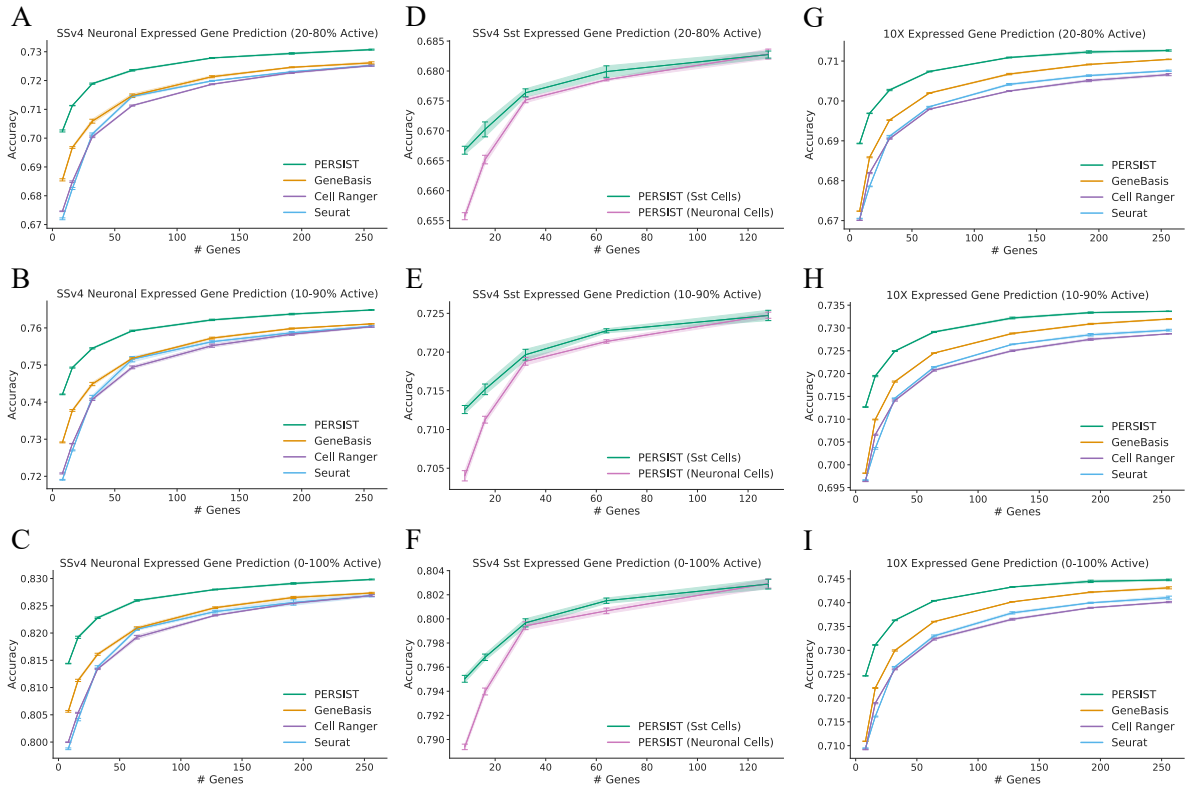


Figure F.4: Expressed gene prediction with different expression cutoffs. Genes that are expressed in either all cells or no cells are easiest to predict, and focusing on these genes can inflate the expressed gene prediction accuracy. We examine how the accuracy changes as we apply different cutoffs to focus on more difficult-to-predict genes. **A**, SSv4 accuracy for genes expressed in 20-80% of cells. **B**, SSv4 accuracy for genes expressed in 10-90% of cells. **C**, SSv4 accuracy for genes expressed in 0-100% of cells (all genes). **D**, SSv4 Sst accuracy for genes expressed in 20-80% of cells. **E**, SSv4 Sst accuracy for genes expressed in 10-90% of cells. **F**, SSv4 Sst accuracy for genes expressed in 0-100% of cells. **G**, 10X accuracy for genes expressed in 20-80% of cells. **H**, 10X accuracy for genes expressed in 10-90% of cells. **I**, 10X accuracy for genes expressed in 0-100% of cells (all genes). All error bars represent 95% confidence intervals determined by bootstrapped training.

containing 32 genes. The results are visibly improved, reflecting the higher overall accuracy (74%), but certain cell types remain difficult to classify. This provides further motivation for designing FISH studies that focus on coarse-grained cell types, as we described in the main text.

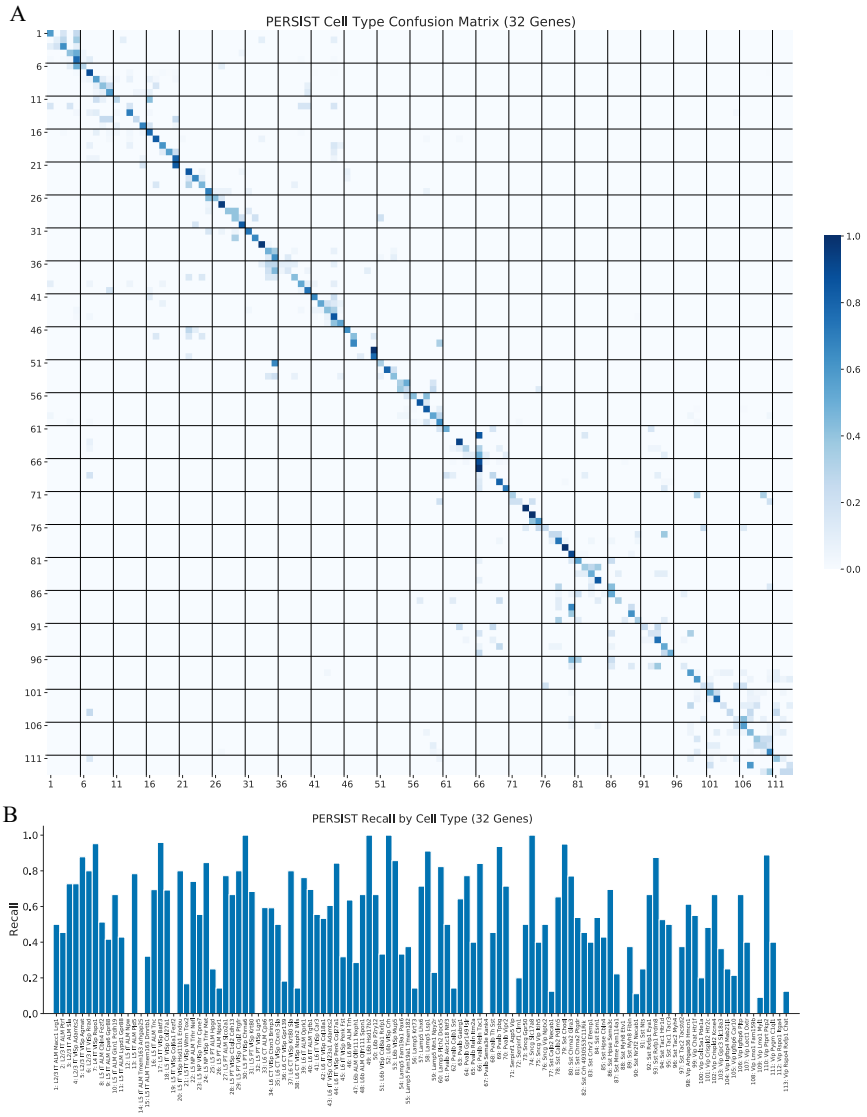


Figure F.5: PERSIST cell type classification metrics. Additional metrics quantify the classification accuracy on a per-type level for the 32-gene PERSIST panel. **A**, Confusion matrix where each row corresponds to a cell type and values indicate the percentage of cells classified into each type. Names for each numbered cell type are provided in the bar chart below. **B**, Recall (true positive rate) for each cell type.

Next, as an intuitive visualization of how PERSIST can distinguish cell types using only binarized expression levels, we plot the frequency of each gene being expressed within each cell type. Figure F.7 displays this expression matrix for the full set of transcriptomic

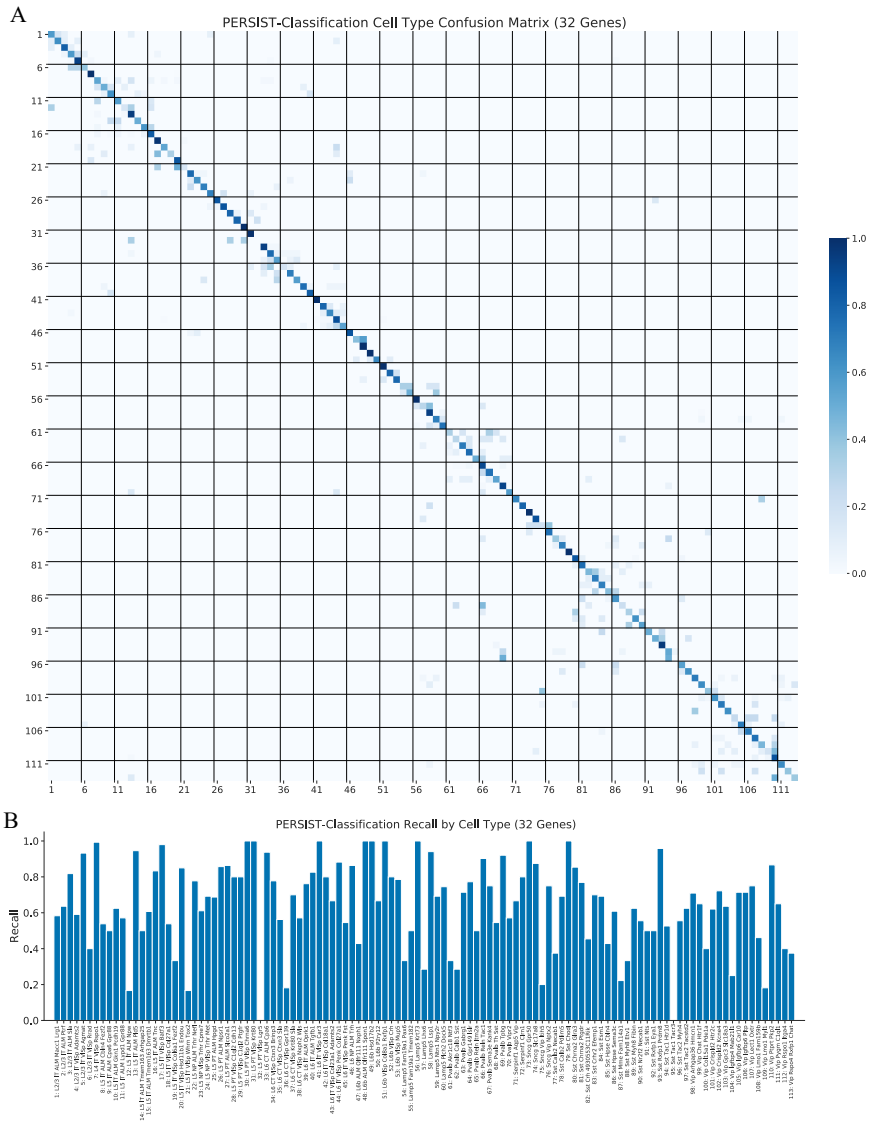


Figure F.6: PERSIST-Classification cell type classification metrics. Additional metrics quantify the classification accuracy on a per-type level for the 32-gene PERSIST-Classification panel. **A**, Confusion matrix where each row corresponds to a cell type and values indicate the percentage of cells classified into each type. Names for each numbered cell type are provided in the bar chart below. **B**, Recall (true positive rate) for each cell type.

cell types in the SSv4 dataset, as well as for 50 and 25 subclasses, using the PERSIST and PERSIST-Classification panels containing 16 genes. The results show that each cell type has a unique expression pattern, enabling us to distinguish cell types using a surprisingly small

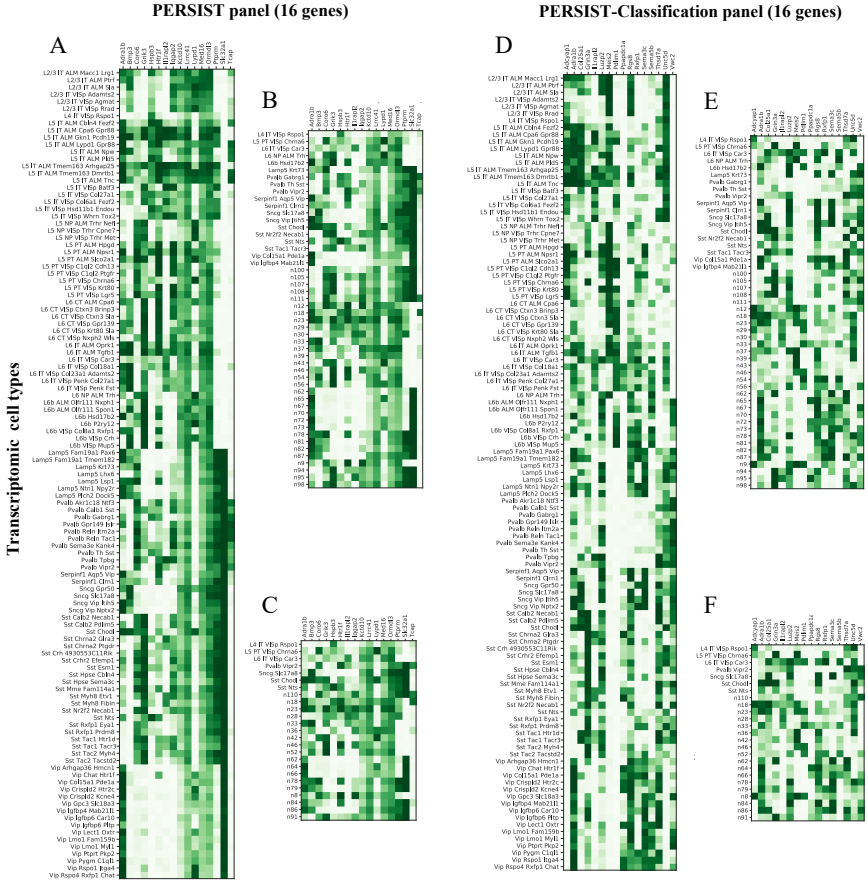


Figure F.7: PERSIST gene expression frequency by cell type. Using the SSv4 dataset, we plot the portion of the time that each gene in a panel is expressed within each cell type (white indicates never expressed, dark green indicates always expressed). Examining these matrices for the PERSIST and PERSIST-Classification panels containing 16 genes, using both the full set of transcriptomic cell types as well as small numbers of subclasses, we can identify distinct expression patterns within each cell type. **A**, Expression matrix for PERSIST gene panel with all cell types. **B**, Expression matrix for PERSIST gene panel with 50 subclasses. **C**, Expression matrix for PERSIST gene panel with 25 subclasses. **D**, Expression matrix for PERSIST-Classification gene panel with all cell types. **E**, Expression matrix for PERSIST-Classification gene panel with 50 subclasses. **F**, Expression matrix for PERSIST-Classification gene panel with 25 subclasses.

number of genes. Although all configurations of gene expression are technically possible, we find that many genes are expressed nearly all of the time or none of the time within each cell

type. For example, we find that with all 113 transcriptomic cell types, 67% of the entries are expressed either less than 20% or more than 80% of the time (Figure F.7A); this figure is even higher at 74% for PERSIST-Classification (Figure F.7D).

For the MERFISH gene imputation experiments, we provide several additional results in Figure F.8. First, Figure F.8A-B show the imputation accuracy for the supervised methods, scGeneFit, SMaSH and MutInfo. We find that when using the SSv4 V1 scRNA-seq data, PERSIST and MutInfo reach comparable accuracy for small gene panels, but that PERSIST achieves higher accuracy with larger panels; when using the SSv4 ALM scRNA-seq data, the various methods offer comparable accuracy for panels of all sizes. Next, we examine the importance of using a carefully chosen threshold when binarizing the MERFISH gene expression counts. If we naively use a threshold value of zero, the imputation accuracy is significantly lower than if we choose a threshold to match the quantile that zero represents in the scRNA-seq data (Figure F.8C). Finally, we use the fully observed MERFISH dataset to quantify the accuracy lost due to training on out-of-domain scRNA-seq data. We find that training with the in-domain MERFISH data yields an accuracy improvement of $4-6\%$ (Figure F.8D), suggesting that our binarization step does not completely remove the effects of the domain shift relative to the SSv4 V1 scRNA-seq data.

Regarding the diversity in gene panels between different methods, in addition to the results for panels of 32 genes (Figure 8.6), we report the gene panel overlap for panels of size 16 and 128 in Figure F.9. For panels of these sizes, we still find that PERSIST selects genes that are largely different than those identified by other methods. We also find several sets of similar methods that appeared in the main text: scGeneFit and GeneBasis, Seurat and Cell Ranger, and MutInfo, SMaSH and PERSIST-Classification.

Similarly, we report gene panel overlap for the experiments with Patch-seq data (Figure F.10). The results show that PERSIST-Ephys selects up to roughly 30% of the same genes as other methods, with Cell Ranger being the most similar for all panel sizes. PERSIST and Cell Ranger are the next most effective methods in this experiment after PERSIST-Ephys, but these three methods select largely distinct gene panels.

Finally, we examine to what extent the variability in PERSIST's selections across trials impacts the performance in our evaluation metrics (Figure F.11). Using five independent trials with the SSv4 dataset, we examine the mean performance across trials, the gap between the minimum and maximum performance, and the performance from the single trial chosen according to its validation loss (calculated using the PERSIST reconstruction model's hurdle loss). We find that the variability in performance is relatively small, particularly for the explained variance results (Figure F.11A). Choosing a single trial based on its validation loss tends to result in performance above the mean for smaller panels ($\leq$16 genes), but with larger panels it can be slightly less effective than choosing a single trial at random. These results suggest that the significant redundancy in genome-wide expression profiles enables the selection of diverse panels with nearly equivalent information content.

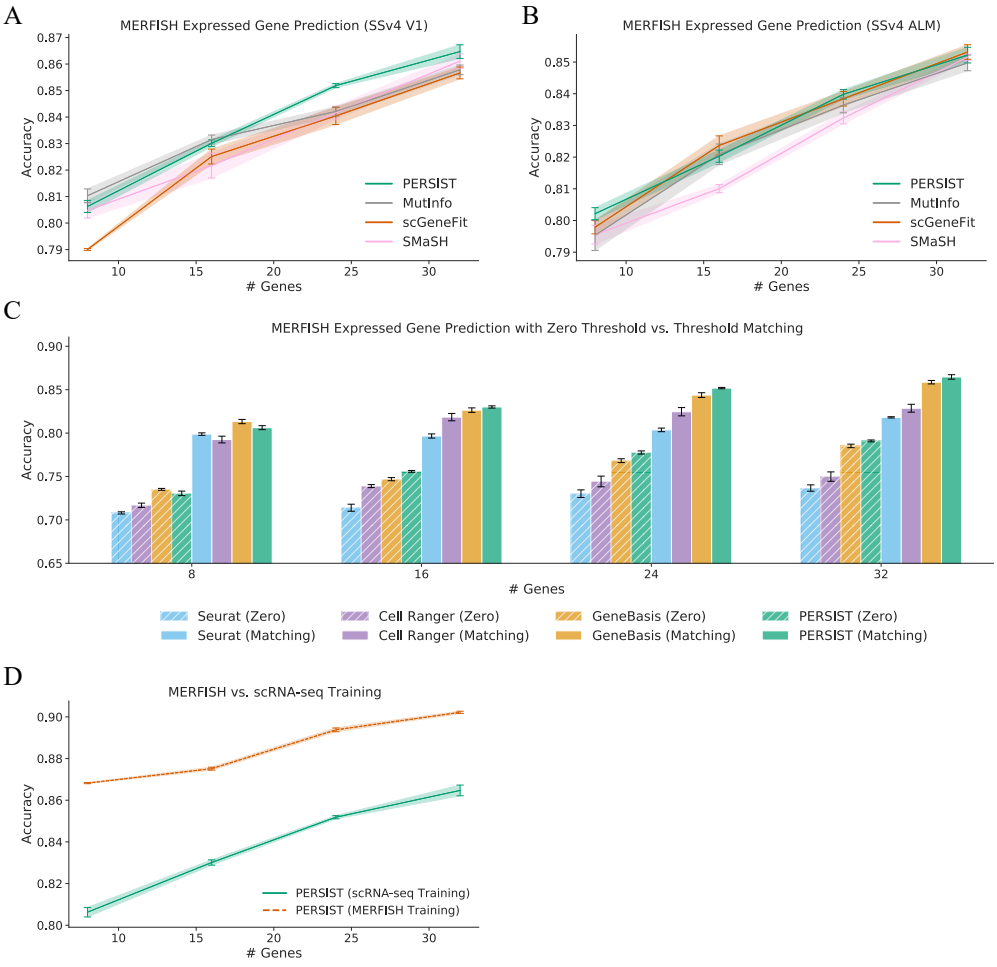


Figure F.8: Additional MERFISH imputation results. Here, we report results for the supervised gene selection methods, investigate the importance of the thresholding procedure for MERFISH expression counts, and examine the accuracy lost due to domain shift between the scRNA-seq and MERFISH datasets. **A**, Imputation accuracy for panels selected by each method when using the SSv4 V1 scRNA-seq data. PERSIST narrowly outperforms the supervised baselines, except for MutInfo with small panel sizes. **B**, Imputation accuracy for panels selected by each method when using the SSv4 ALM scRNA-seq data. The various methods achieve similar performance. **C**, Imputation accuracy when using a zero threshold for MERFISH expression counts, versus when using a threshold that matches the scRNA-seq quantile. Matching the threshold significantly improves the accuracy for all methods across all panel sizes. **D**, Comparing imputation accuracy when selecting genes using V1 cells but training on V1 scRNA-seq versus MOp MERFISH data. Training on in-domain MERFISH data, which is free of domain shift but inaccessible in practice, reveals an accuracy loss of 4-6% due to training on scRNA-seq. All error bars represent 95% confidence intervals determined by training with five bootstrapped datasets.

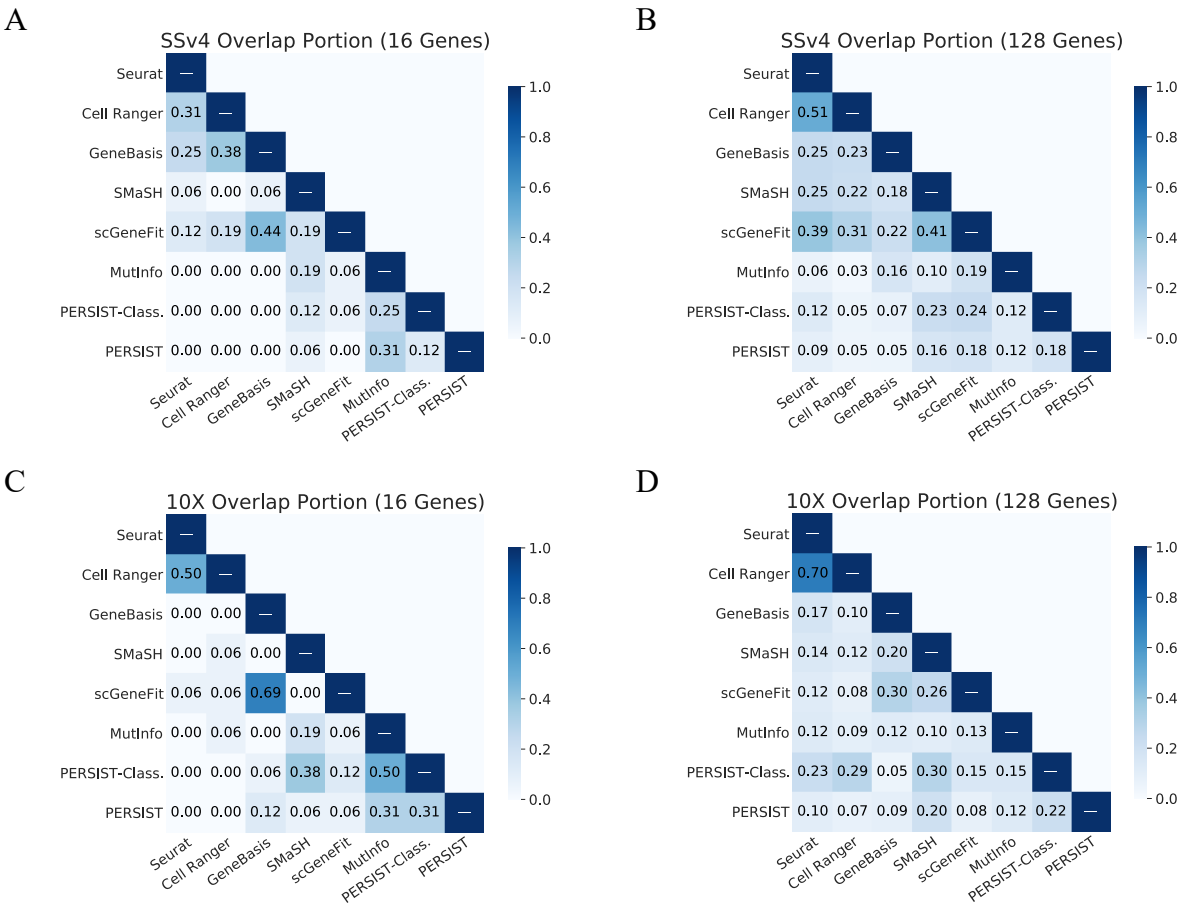


Figure F.9: Diversity in gene panels of different sizes. Using the SSv4 and 10X dataset, we examine the overlap between panels containing 16 and 128 genes. The genes identified by PERSIST for these panel sizes remain distinct from those selected by existing methods. **A**, Portion of overlapping genes between panels of 16 genes for the SSv4 dataset. **B**, Portion of overlapping genes between panels of 128 genes for the SSv4 dataset. **C**, Portion of overlapping genes between panels of 16 genes for the 10X dataset. **D**, Portion of overlapping genes between panels of 128 genes for the 10X dataset.

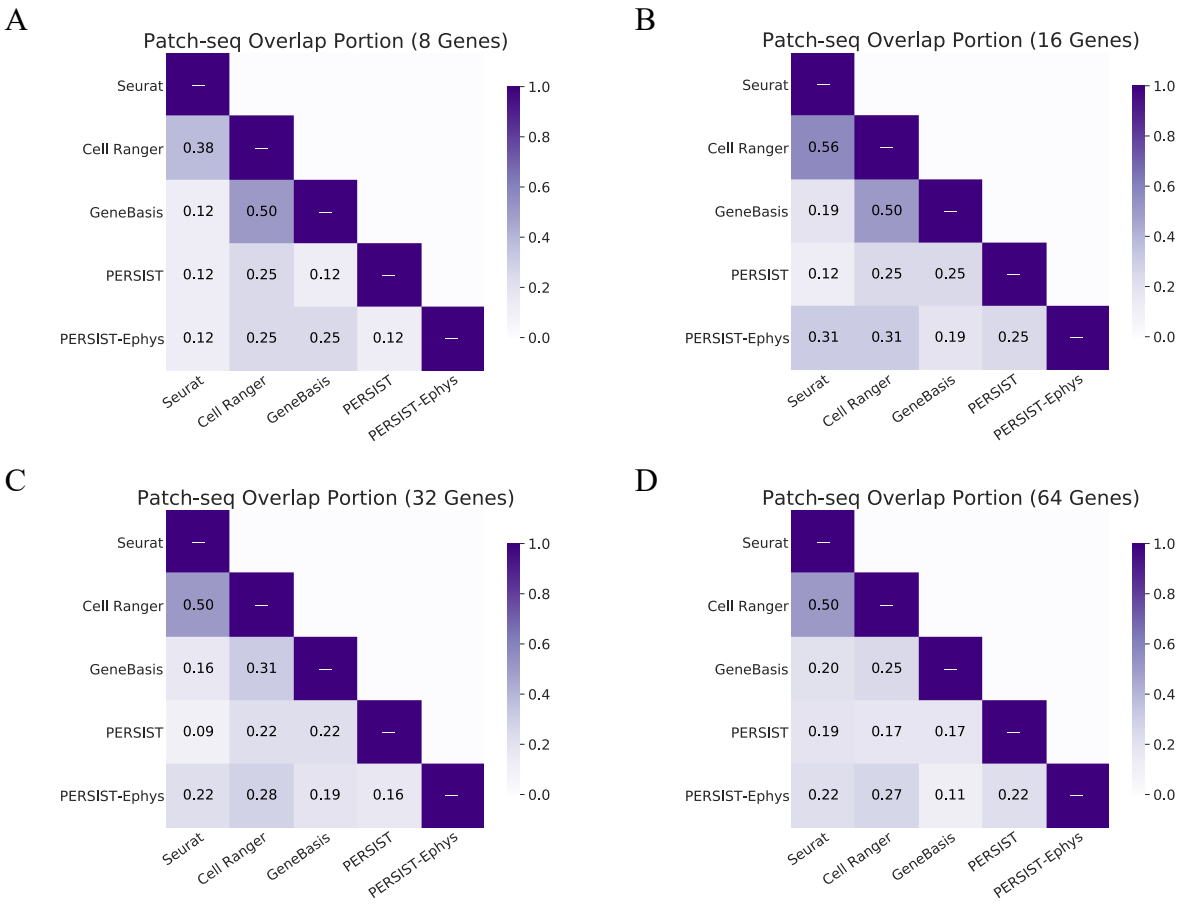


Figure F.10: Diversity in Patch-seq gene panels. We find that for panels of all sizes, PERSIST-Ephys selects distinct genes from the remaining methods. **A**, Portion of overlapping genes for panels of 8 genes. **B**, Portion of overlapping genes for panels of 16 genes. **C**, Portion of overlapping genes for panels of 32 genes. **D**, Portion of overlapping genes for panels of 64 genes.

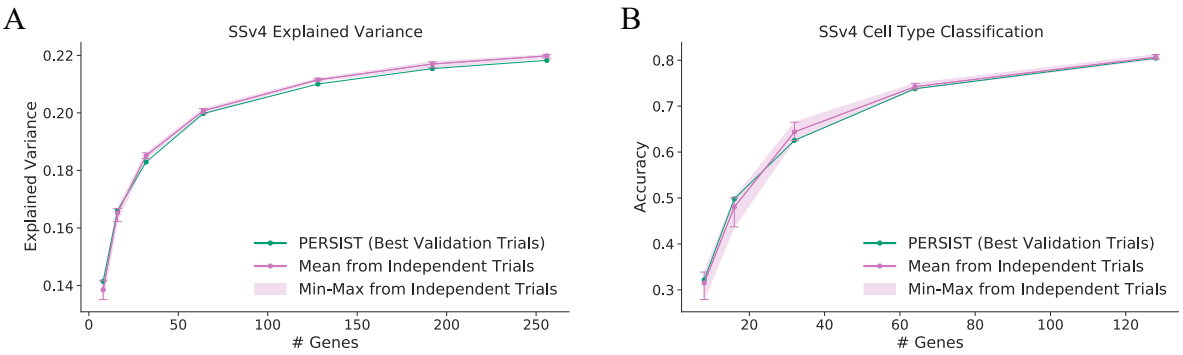


Figure F.11: PERSIST performance variability across runs. PERSIST can select different panels across multiple runs due to stochasticity in its training, but the variability in performance is relatively small. We compare the mean performance across five trials to the minimum and maximum performance, as well as the performance from the single trial chosen according to its validation loss. **A**, Explained variance across five trials for the SSv4 dataset. **B**, Cell type classification accuracy across five trials for the SSv4 dataset.

Appendix G

MAXIMIZING MUTUAL INFORMATION FOR DYNAMIC FEATURE SELECTION

G.1 Proofs

In this section, we re-state and prove our main theoretical results. We begin with our proposition regarding the optimal predictor for an arbitrary policy π.

Proposition G.1. *When $\boldsymbol{y}$ is discrete and ℓ is cross entropy loss, eq.* 9.4 *is minimized for any policy π by the Bayes classifier, or $f^*(x_S) = p(\boldsymbol{y} \mid x_S)$.*

Proof. Given the predictor inputs x_S, our goal is to determine the prediction that minimizes the expected loss. Because features are selected sequentially by π with no knowledge of the non-selected values, there is no other information to condition on; for the predictor, we do not even need to distinguish the order in which features were selected. We can therefore derive the optimal prediction $\hat{y} \in \Delta^{K-1}$ for a discrete response $\boldsymbol{y} \in [K]$ as follows:

$$\begin{aligned} f^*(x_S) &= \arg\min_{\hat{y}} \; \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S} \big[\ell(\hat{y}, \boldsymbol{y})\big] \\ &= \arg\min_{\hat{y}} \; \sum_{i \in \mathcal{Y}} p(\boldsymbol{y} = i \mid x_S) \log \hat{y}_i \\ &= \arg\min_{\hat{y}} \; D_{\mathrm{KL}}\big(p(\boldsymbol{y} \mid x_S) \mid\mid \hat{y}\big) + H(\boldsymbol{y} \mid x_S) \\ &= p(\boldsymbol{y} \mid x_S). \end{aligned}$$

In the case of a continuous response $\boldsymbol{y} \in \mathbb{R}$ with squared error loss, we have a similar result

involving the response's conditional expectation:

$$\begin{aligned} f^*(x_S) &= \arg\min_{\hat{y}} \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S} \left[(\hat{y} - \boldsymbol{y})^2\right] \\ &= \arg\min_{\hat{y}} \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S} \left[(\hat{y} - \mathbb{E}[\boldsymbol{y} \mid x_S])^2\right] + \mathrm{Var}(\boldsymbol{y} \mid x_S) \\ &= \mathbb{E}[\boldsymbol{y} \mid x_S]. \end{aligned}$$

□

Proposition G.2. *When $\boldsymbol{y}$ is discrete, ℓ is cross entropy loss and the predictor is the Bayes classifier f^*, eq.* 9.4 *is minimized by the greedy CMI policy, or $\pi^*(x_S) = \arg\max_i \; I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$.*

Proof. Following eq. 9.4, the policy network's selection $i = \pi(x_S)$ incurs the following expected loss with the distribution $p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)$:

$$\begin{aligned} \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[\ell(f^*(x_S \cup \boldsymbol{x}_i), \boldsymbol{y})\right] &= \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[\ell(p(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S), \boldsymbol{y})\right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S} \left[\mathop{\mathbb{E}}_{\boldsymbol{y}|\boldsymbol{x}_i,x_S} [\ell(p(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S), \boldsymbol{y})] \right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S} \left[H(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)\right] \\ &= H(\boldsymbol{y} \mid x_S) - I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S). \end{aligned}$$

Note that $H(\boldsymbol{y} \mid x_S)$ is a constant that does not depend on i. When identifying the index that minimizes the expected loss, we thus have the following result:

$$\arg\min_i \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[\ell(f^*(x_S \cup \boldsymbol{x}_i), \boldsymbol{y})\right] = \arg\max_i \; I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S).$$

In the case of a continuous response with squared error loss and an optimal predictor given

by $f^*(x_S) = \mathbb{E}[\boldsymbol{y} \mid x_S]$, we have a similar result:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}\left[(f^*(x_S \cup \boldsymbol{x}_i) - \boldsymbol{y})^2\right] &= \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}\left[(\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}_i, x_S] - \boldsymbol{y})^2\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\Big[\mathop{\mathbb{E}}_{\boldsymbol{y}|\boldsymbol{x}_i,x_S}[(\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}_i, x_S] - \boldsymbol{y})^2]\Big] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)].
\end{aligned}$$

When we aim to minimize the expected loss, our selection is thus the index that yields the lowest expected conditional variance:

$$\arg\min_i \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)].$$

□

We also prove the limiting result presented in eq. 9.3, which states that $I_i^n \to I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$.

Proof. Conditional mutual information $I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$ is defined as follows (Cover and Thomas, 2012):

$$\begin{aligned}
I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S) &= D_{\mathrm{KL}}\big(p(\boldsymbol{x}_i, \boldsymbol{y} \mid x_S) \mid\mid p(\boldsymbol{x}_i \mid x_S)p(\boldsymbol{y} \mid x_S)\big) \\
&= \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}\Big[\log \frac{p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)}{p(\boldsymbol{x}_i \mid x_S)p(\boldsymbol{y} \mid x_S)}\Big].
\end{aligned}$$

Rearranging terms, we can write this as an expected KL divergence with respect to $\boldsymbol{x}_i$:

$$\begin{aligned}
I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S) &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S,\boldsymbol{x}_i}\Big[\log \frac{p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)}{p(\boldsymbol{x}_i \mid x_S)p(\boldsymbol{y} \mid x_S)}\Big] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S,\boldsymbol{x}_i}\Big[\log \frac{p(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)}{p(\boldsymbol{y} \mid x_S)}\Big] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\Big[D_{\mathrm{KL}}\big(p(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S) \mid\mid p(\boldsymbol{y} \mid x_S)\big)\Big]
\end{aligned}$$

Now, when we sample multiple values $x_i^1, \ldots, x_i^n \sim p(\boldsymbol{x}_i \mid x_S)$ and make predictions using

the Bayes classifier, we have the following mean prediction as n becomes large:

$$\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^{n} p(\boldsymbol{y} \mid x_S, x_i^j) = \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\big[p(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)\big] = p(\boldsymbol{y} \mid x_S).$$

Calculating the mean KL divergence across the predictions, we arrive at the following result:

$$\lim_{n\to\infty} I_i^n = \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\Big[D_{\mathrm{KL}}\big(p(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S) \;||\; p(\boldsymbol{y} \mid x_S)\big)\Big] = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S).$$

□

Theorem G.1. *When $\boldsymbol{y}$ is discrete and ℓ is cross entropy loss, the global optimum of eq.* 9.5 *is a predictor that satisfies $f(x_S; \theta^*) = p(\boldsymbol{y} \mid x_S)$ and a policy $\pi(x_S; \phi^*)$ that puts all probability mass on $i^* = \arg\max_i I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$.*

Proof. We first consider the predictor network $f(\boldsymbol{x}_S; \theta)$. When the predictor is given the feature values x_S, it means that one index $i \in S$ was chosen by the policy according to $\pi(x_{s\setminus i}; \phi)$ and the remaining indices $S \setminus i$ were sampled from $p(\boldsymbol{S})$. Because $\boldsymbol{S}$ is sampled independently from $(\boldsymbol{x}, \boldsymbol{y})$, and because $\pi(x_{S\setminus i}; \phi)$ is not given access to $(\boldsymbol{x}_{[d]\setminus S}, \boldsymbol{x}_i, \boldsymbol{y})$, the predictor's expected loss must be considered with respect to the distribution $\boldsymbol{y} \mid x_S$. The globally optimal predictor $f(x_S; \theta^*)$ is thus defined as follows, regardless of the selection policy $\pi(x_S; \phi)$ and which index i was selected last:

$$f(x_S; \theta^*) = \arg\min_{\hat{y}} \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}\big[\ell(\hat{y}, \boldsymbol{y})\big] = p(\boldsymbol{y} \mid x_S).$$

The above result follows from our proof for Proposition 9.1. Now, given the optimal predictor $f(x_S; \theta^*)$, we can define the globally optimal policy by minimizing the expected loss for a fixed input x_S. Denoting the probability mass placed on each index $i \in [d]$ as $\pi_i(x_S; \phi)$,

where $\pi(x_S;\phi) \in \Delta^{d-1}$, the expected loss is the following:

$$\begin{aligned}\mathop{\mathbb{E}}_{i\sim\pi(x_S;\phi)} \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \big[\ell(f(x_S \cup \boldsymbol{x}_i;\theta^*),\boldsymbol{y})\big] &= \sum_{i\in[d]} \pi_i(x_S;\phi) \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \big[\ell\big(f(x_S \cup \boldsymbol{x}_i;\theta^*),\boldsymbol{y}\big)\big] \\ &= \sum_{i\in[d]} \pi_i(x_S;\phi) \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S} [H(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)].\end{aligned}$$

The above result follows from our proof for Proposition 9.2. If there exists a single index $i^* \in [d]$ that yields the lowest expected conditional entropy, or

$$\mathop{\mathbb{E}}_{\boldsymbol{x}_{i^*}|x_S} [H(\boldsymbol{y} \mid \boldsymbol{x}_{i^*}, x_S)] < \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S} [H(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)] \quad \forall i \neq i^*,$$

then the optimal predictor must put all its probability mass on i^*, or $\pi_{i^*}(x_S;\phi^*) = 1$. Note that the corresponding feature $\boldsymbol{x}_{i^*}$ has maximum conditional mutual information with $\boldsymbol{y}$, because we have

$$I(\boldsymbol{y};\boldsymbol{x}_{i^*} \mid x_S) = \underbrace{H(\boldsymbol{y} \mid x_S)}_{\text{Constant}} - \mathop{\mathbb{E}}_{\boldsymbol{x}_{i^*}|x_S} [H(\boldsymbol{y} \mid \boldsymbol{x}_{i^*}, x_S)].$$

To summarize, we derived the global optimum to our objective $\mathcal{L}(\theta,\phi)$ by first considering the optimal predictor $f(\boldsymbol{x}_S;\theta^*)$, and then considering the optimal policy $\pi(\boldsymbol{x}_S;\phi^*)$ when we assume that we use the optimal predictor. □

Theorem G.2. *When $\boldsymbol{y}$ is continuous and ℓ is squared error loss, the global optimum of eq.* 9.5 *is a predictor that satisfies $f(x_S;\theta^*) = \mathbb{E}[\boldsymbol{y} \mid x_S]$ and a policy $\pi(x_S;\phi^*)$ that puts all probability mass on $i^* = \arg\min_i \mathbb{E}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)]$.*

Proof. Our proof follows the same logic as our proof for Theorem 9.1. For the optimal predictor given an arbitrary policy, we have:

$$f(x_S;\theta^*) = \arg\min_{\hat{y}} \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S} \big[(\hat{y} - \boldsymbol{y})^2\big] = \mathbb{E}[\boldsymbol{y} \mid x_S].$$

Then, for the policy's expected loss, we have:

$$\mathop{\mathbb{E}}_{i\sim\pi(x_S;\phi)} \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[\left(f(x_S \cup \boldsymbol{x}_i;\theta^*) - \boldsymbol{y}\right)^2\right] = \sum_{i\in[d]} \pi_i(x_S;\phi) \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S} [\mathrm{Var}(\boldsymbol{y} \mid \boldsymbol{x}_i, x_S)].$$

If there exists an index $i^* \in [d]$ that yields the lowest expected conditional variance, then the optimal policy must put all its probability mass on i^*, or $\pi_{i^*}(x_S;\phi^*) = 1$. □

G.2 Datasets

The datasets used in our experiments are summarized in Table G.1. Three of the tabular datasets and the two image classification datasets are publicly available, and the three emergency medicine tasks were privately curated from the Harborview Medical Center Trauma Registry.

Table G.1: Summary of datasets used in our experiments.

Dataset	# Features	# Feature Groups	# Classes	# Samples
Fluid	224	162	2	2,770
Respiratory	112	35	2	65,515
Bleeding	121	44	2	6,496
Spam	58	–	2	4,601
MiniBooNE	51	–	2	130,064
Diabetes	45	–	3	92,062
MNIST	784	–	10	60,000
CIFAR-10	1,024	64	10	60,000

MiniBooNE and spam classification The spam dataset includes features extracted from e-mail messages to predict whether or not a message is spam. Three features describes the usage of capital letters in the e-mail, and the remaining 54 features describe the frequency

with which certain key words or characters are used. The MiniBooNE particle identification dataset involves distinguishing electron neutrinos from muon neutrinos based on various continuous features (Roe et al., 2005). Both datasets were obtained from the UCI repository (Dua and Graff, 2017).

Diabetes classification The diabetes dataset was obtained from from the National Health and Nutrition Examination Survey (NHANES) (NHA, 2018), an ongoing survey designed to assess the well-being of adults and children in the United States. We used a version of the data pre-processed by Kachuee et al. (2018, 2019) that includes data collected from 1999 through 2016. The input features include demographic information (age, gender, ethnicity, etc.), lab results (total cholesterol, triglyceride, etc.), examination data (weight, height, etc.), and questionnaire answers (smoking, alcohol, sleep habits, etc.). An expert was also asked to suggest costs for each feature based on the financial burden, patient privacy, and patient inconvenience, but we assume uniform feature costs in our experiments. Finally, the fasting glucose values were used to define three classes based on standard threshold values: normal, pre-diabetes, and diabetes.

Image classification datasets The MNIST and CIFAR-10 datasets were downloaded using PyTorch (Paszke et al., 2017). We used the standard train-test splits, and we split the train set to obtain a validation set with the same size as the test set (10,000 examples).

Emergency medicine datasets The emergency medicine datasets used in this study were gathered over a 13-year period (2007-2020) and encompass 14,463 emergency department admissions. We excluded patients under the age of 18, and we curated 3 clinical cohorts commonly seen in pre-hospitalization settings. These include (i) pre-hospital fluid resuscitation, (ii) emergency department respiratory support, and (iii) bleeding after injury. These datasets are not publicly available due to patient privacy concerns.

Pre-hospital fluid resuscitation We selected 224 variables that were available in the pre-hospital setting, including dispatch information (injury date, time, cause, and location), demographic information (age, sex), and pre-hospital vital signs (blood pressure, heart rate, respiratory rate). The outcome was each patient's response to fluid resuscitation, following the Advanced Trauma Life Support (ATLS) definition (Subcommittee et al., 2013).

Emergency department respiratory support In this cohort, our goal is to predict which patients require respiratory support upon arrival in the emergency department. Similar to the previous dataset, we selected 112 pre-hospital clinical features including dispatch information (injury date, time, cause, and location), demographic information (age, sex), and pre-hospital vital signs (blood pressure, heart rate, respiratory rate). The outcome is defined based on whether a patient received respiratory support, including both invasive (intubation) and non-invasive (BiPap) approaches.

Bleeding In this cohort, we only included patients whose fibrinogen levels were measured, as this provides an indicator for bleeding or fibrinolysis (Mosesson, 2005). As with the previous datasets, demographic information, dispatch information, and pre-hospital observations were used as input features. The outcome, based on experts' opinion, was defined by whether an individual's fibrinogen level is below 200 mg/dL, which represents higher risk of bleeding after injury.

G.3 Baselines

This section provides more details on the baseline methods used in our experiments (Section 9.7).

G.3.1 Global Feature Importance Methods

Two of our static feature selection baselines, permutation tests and SAGE, are *global feature importance methods* that rank features based on their role in improving model accuracy

(Covert et al., 2021). In our experiments, we ran each method using a single classifier trained on the entire dataset, and we then selected the top k features depending on the budget.

When running the permutation test, we calculated the validation AUROC while replacing values in the corresponding feature column with random draws from the training set. When running SAGE, we used the authors' implementation with automatic convergence detection (Covert et al., 2020). To handle held-out features, we averaged across 128 sampled values for the six tabular datasets, and for MNIST we used a zeros baseline to achieve faster convergence.

G.3.2 Local Feature Importance Methods

Two of our static feature selection baselines, DeepLift and Integrated Gradients, are *local feature importance methods* that rank features based on their importance to a single prediction. In our experiments, we generated feature importance scores for the true class using all examples in the validation set. We then selected the top k features based on their mean absolute importance. We used a mean baseline for Integrated Gradients (Sundararajan et al., 2017), and both methods were run using the Captum package (Kokhlikyan et al., 2020).

G.3.3 Differentiable Feature Selection

Our last static feature selection baseline is the Concrete autoencoder (CAE) from Balın et al. (2019). The method was originally proposed to perform unsupervised feature selection by reconstructing the full input vector, but we changed the prediction target to use it in a supervised fashion. The authors propose training with an exponentially decayed temperature over a hand-tuned number of epochs, but we used an approach similar to our own method: we trained with a sequence of temperature values, performing early stopping using the validation loss for each one, and we returned the features chosen after training with the final temperature.

We tried a similar method proposed by Yamada et al. (2020), but this method requires tuning a penalty parameter to achieve the desired number of features, and we found that it gave similar performance in our experiments on MNIST. Among methods that learn to select features within a neural network, there are several others that do so using group sparse penalties (Feng and Simon, 2017; Tank et al., 2021; Lemhadri et al., 2021); we tested the LassoNet approach from Lemhadri et al. (2021) and found that it was not competitive on MNIST. For simplicity, we present results only for the supervised CAE.

G.3.4 CMI Estimation

Our experiments use two versions of the CMI estimation approach described in Section 9.4. Both are inspired by the EDDI method introduced by Ma et al. (2019), but a key difference is that we do not jointly model $(\boldsymbol{x}, \boldsymbol{y})$ within the same conditional generative model: we instead separately model the response with a classifier $f(\boldsymbol{x}_S) \approx p(\boldsymbol{y} \mid \boldsymbol{x}_S)$ and the features with a generative model of $p(\boldsymbol{x}_i \mid \boldsymbol{x}_S)$. This partially mitigates one challenge with this approach, which is working with mixed continuous/categorical data (i.e., we do not need to jointly model categorical response variables).

For the first version of this approach, we train a PVAE as a generative model (Ma et al., 2019). The encoder and decoder both have two hidden layers, the latent dimension is set to 16, and we use 128 samples from the latent posterior to approximate $p(\boldsymbol{x}_i \mid x_S) = \int p(\boldsymbol{x}_i \mid \boldsymbol{z}) p(\boldsymbol{z} \mid x_S)$. We use Gaussian distributions for both the latent and decoder spaces, and we generate samples using the decoder mean, similar to the original approach (Ma et al., 2019). In the second version, we bypass the need for a generative model with a simple approximation: we sample features from their marginal distribution, which is equivalent to assuming feature independence.

G.3.5 Opportunistic Learning

Kachuee et al. (2018) proposed Opportunistic Learning (OL), an approach to solve DFS using RL. The model consists of two networks analogous to our policy and predictor: a Q-

network that estimates the value associated with each action, where actions correspond to features, and a P-network responsible for making predictions. When using OL, we use the same architectures as our approach, and OL shares network parameters between the P- and Q-networks.

The authors introduce a utility function for their reward, shown in eq. G.1, which calculates the difference in prediction uncertainty as approximated by MC dropout (Gal and Ghahramani, 2016). The reward also accounts for feature costs, but we set all feature costs to $c_i = 1$:

$$r_i = \frac{||Cert(x_S) - Cert(x_S \cup x_i)||}{c_i} \tag{G.1}$$

To provide a fair comparison with the remaining methods, we made several modifications to the authors' implementation. These include (i) preventing the prediction action until the pre-specified budget is met, (ii) setting all feature costs to be identical, and (iii) supporting pre-defined feature groups as described later in this appendix. When training, we update the P-, Q-, and target Q-networks every $1 + \frac{d}{100}$ experiences, where d is the number of features in a dataset. In addition, the replay buffer is set to store the $1000d$ most recent experiences, and the random exploration probability is decayed so that it eventually reaches a value of 0.1.

G.4 Training Approach and Hyperparameters

This section provides more details on our training approach and hyperparameter choices.

G.4.1 Training Pseudocode

Algorithm 3 summarizes our training approach. Briefly, we select features by drawing a Concrete sample using policy network's logits, we calculate the loss based on the subsequent prediction, and we then update the mask for the next step using a discrete sample from the policy's distribution. We implemented this approach using PyTorch (Paszke et al., 2017) and PyTorch Lightning.[1]

Algorithm 3: Training pseudocode

Input: Data distribution $p(\boldsymbol{x}, \boldsymbol{y})$, budget $k > 0$, learning rate $\gamma > 0$, temperature $\tau > 0$

Output: Predictor model $f(\boldsymbol{x}; \theta)$, policy model $\pi(\boldsymbol{x}; \phi)$

initialize $f(\boldsymbol{x}; \theta), \pi(\boldsymbol{x}; \phi)$

while *not converged* **do**

  sample $x, y \sim p(\boldsymbol{x}, \boldsymbol{y})$

  initialize $\mathcal{L} = 0$, $m = [0, \ldots, 0]$

  for $j = 1$ **to** k **do**

    calculate logits $\alpha = \pi(x \odot m; \phi)$, sample $G_i \sim$ Gumbel for $i \in [d]$

    set $\tilde{m} = \max\big(m, \text{softmax}(G + \alpha, \tau)\big)$ `// update with Concrete`

    set $m = \max\big(m, \text{softmax}(G + \alpha, 0)\big)$ `// update with one-hot`

    update $\mathcal{L} \leftarrow \mathcal{L} + \ell\big(f(x \odot \tilde{m}; \theta), y\big)$

  end

  update $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}$, $\phi \leftarrow \phi - \gamma \nabla_\phi \mathcal{L}$

end

return $f(\boldsymbol{x}; \theta), \pi(\boldsymbol{x}; \phi)$

[1] `https://www.pytorchlightning.ai`

One notable difference between Algorithm 3 and our objective $\mathcal{L}(\theta, \phi)$ in the main text is the use of the policy $\pi(\boldsymbol{x}; \phi)$ for generating feature subsets. This differs from eq. 9.5, which generates feature subsets using a subset distribution $p(\boldsymbol{S})$. The key shared factor between both approaches is that there are separate optimization problems over each feature set that are effectively treated independently. For each feature set x_S, the problem is the one-step-ahead loss, and it incorporates both the policy and predictor as follows:

$$\mathop{\mathbb{E}}_{i \sim \pi(\boldsymbol{x}_{\boldsymbol{S}}; \phi)} \big[\ell\big(f(\boldsymbol{x}_{\boldsymbol{S}} \cup \boldsymbol{x}_i; \theta), \boldsymbol{y} \big) \big]. \tag{G.2}$$

The problems for each subset do not interact: during optimization, the selection given x_S is based only on the immediate change in the loss, and gradients are not propagated through multiple selections as they would be for an RL-based solution. In solving these multiple problems, the difference is simply that eq. 9.5 weights them according to $p(\boldsymbol{S})$, whereas Algorithm 3 weights them according to the current policy $\pi(\boldsymbol{x}, \phi)$.

We find that incorporating the current policy when generating feature sets is important to achieve good performance. As an ablation, we tested how much the method's performance changes when we instead generate training examples $(\boldsymbol{x}_{\boldsymbol{S}}, \boldsymbol{y})$ at random: using the MNIST dataset with varying numbers of features, we find that using random subsets leads to a significant drop in performance (Table G.2).

G.4.2 Model Selection

One detail not shown in Algorithm 3 that we alluded to in the main text is our approach for decaying the Concrete distribution's temperature parameter. We train with a sequence of relatively few temperature values, using the validation loss to perform early stopping with each value. To perform model selection, we separately calculate the validation loss using a temperature value of zero, which more accurately represents the model's usage at inference time; we eventually return the version of the model that performed best on this zero-temperature loss, chosen across all training temperatures.

Table G.2: Training algorithm ablation using MNIST.

# Features	5	10	15	20	25	30	40	50
Ours	0.695	0.875	0.926	0.950	0.960	0.966	0.973	0.975
Ablation	0.578	0.757	0.807	0.819	0.838	0.850	0.869	0.883

G.4.3 Hyperparameters

Our experiments with the six tabular datasets used fully connected architectures with dropout in all layers (Srivastava et al., 2014). The dropout probability is set to 0.3, the networks have two hidden layers of width 128, and we performed early stopping using the validation loss. For our method, the predictor and policy were separate networks with identical architectures. When training models with the features selected by static methods, we reported results using the best model from multiple training runs based on the validation loss. We did not perform any additional hyperparameter tuning due to the large number of models being trained.

For MNIST, we used fully connected architectures with two layers of width 512 and the dropout probability set to 0.3. Again, our method used separate networks with identical architectures. For CIFAR-10, we used a shared ResNet backbone (He et al., 2016b) consisting of several residually connected convolutional layers. The classification head consists of global average pooling and a linear layer, and the selection head consisted of a transposed convolution layer followed by a 1×1 convolution, which outputs a grid of logits with size 8×8. Our CIFAR-10 networks are trained using random crops and random horizontal flips as augmentations.

G.4.4 Feature Grouping

All of the methods used in our experiments were designed to select individual features, but this is undesirable when using categorical features with one-hot encodings. Each of our three

emergency medicine tasks involve such features, so we extended each method to support feature grouping.

SAGE and permutation tests are trivial to extend to feature groups: we simply removed groups of features rather than individual features when calculating importance scores. For DeepLift and Integrated Gradients, we used the summed importance within each group, which preserves each method's additivity property. For the method based on Concrete Autoencoders, we implemented a generalized version of the selection layer that operates on feature groups. We also extended OL to operate on feature groups by having actions map to groups rather than individual features.

Finally, for our method, we parameterized the policy network $\pi(\boldsymbol{x}; \phi)$ so that the number of outputs is the number of groups g rather than the total number of features d (where $g < d$). When applying masking, we first generate a binary mask $m \in [0, 1]^g$, and we then project the mask into $[0, 1]^d$ using a binary group matrix $G \in \{0, 1\}^{d \times g}$, where $G_{ij} = 1$ if feature i is in group j and $G_{ij} = 0$ otherwise. Thus, our masked input vector is given by $x \odot (Gm)$.

G.5 Additional Results

This section provides several additional experimental results. First, Figure G.1 and Figure G.2 show the same results as Figure 9.2 but larger for improved visibility. Next, Figure G.3 though Figure G.8 display the feature selection frequency for each of the tabular datasets when using the greedy method. The heatmaps in each plot show the portion of the time that a feature (or feature group) is selected under a specific feature budget. These plots reveal that our method is indeed selecting different features for different samples.

Finally, Figure G.9 displays examples of CIFAR-10 predictions given different numbers of revealed patches. The predictions generally become relatively accurate after revealing only a small number of patches, reflecting a similar result as Figure 9.3. Qualitatively, we can see that the policy network learns to select vertical stripes, but the order in which it fills out each stripe depends on where it predicts important information may be located.

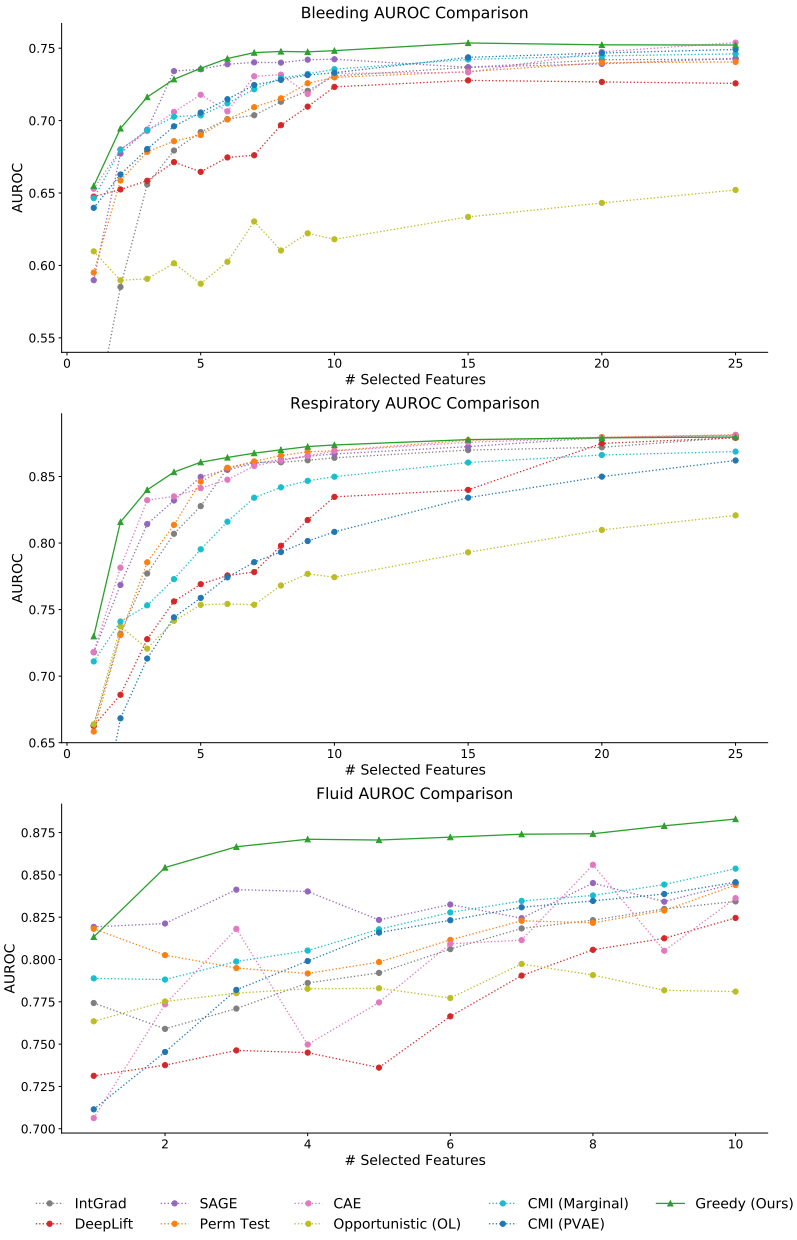


Figure G.1: AUROC comparison on the three emergency medicine diagnosis tasks.

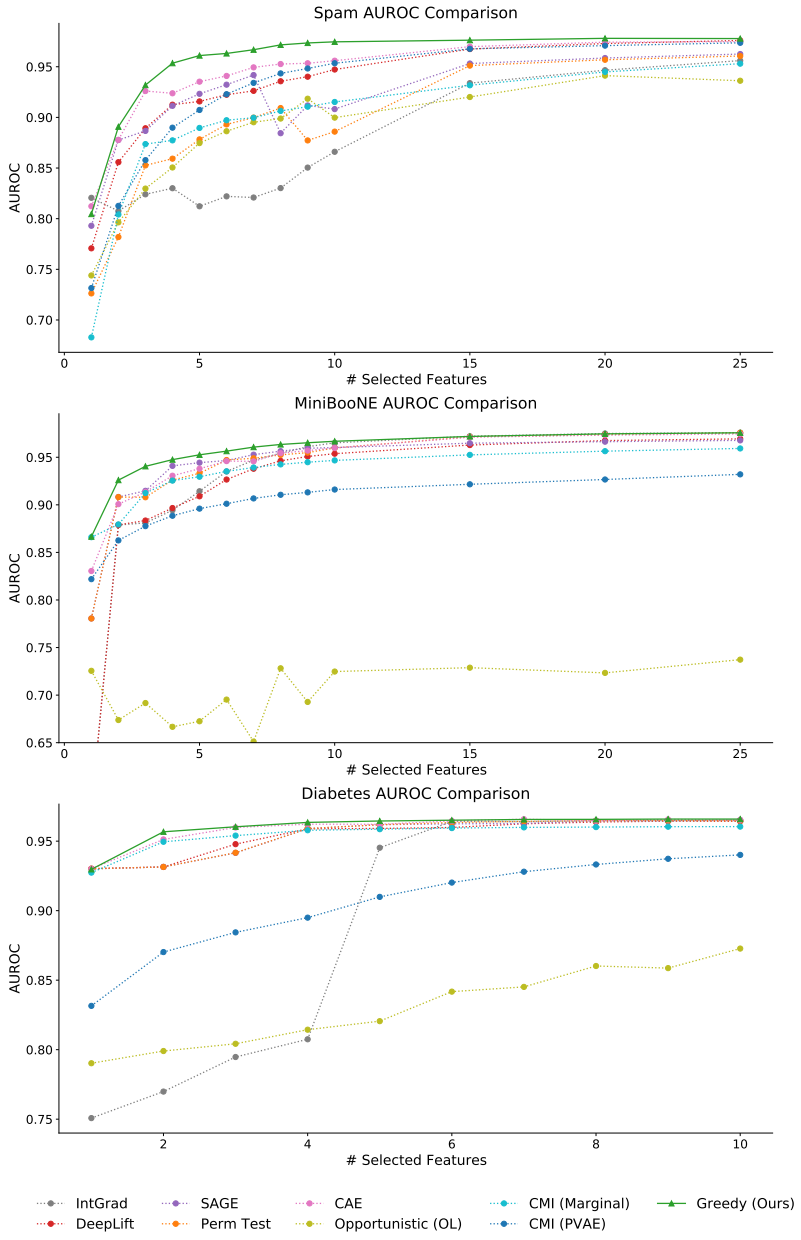


Figure G.2: AUROC comparison on the three public tabular datasets.

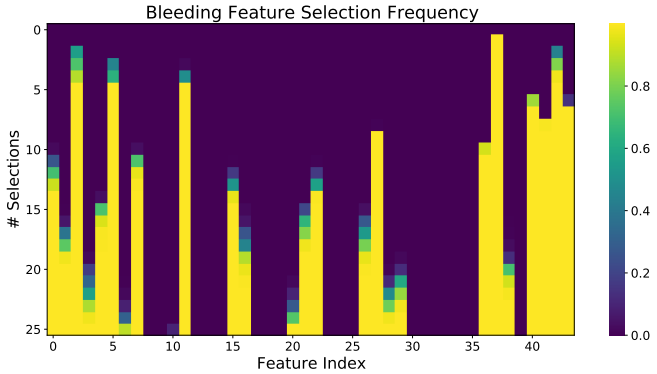


Figure G.3: Feature selection frequency for our greedy approach on the bleeding dataset.

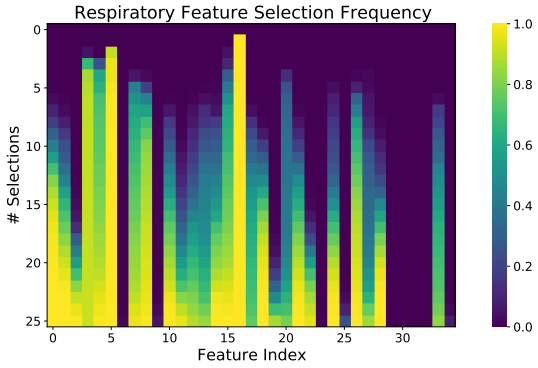


Figure G.4: Feature selection frequency for our greedy approach on the respiratory dataset.

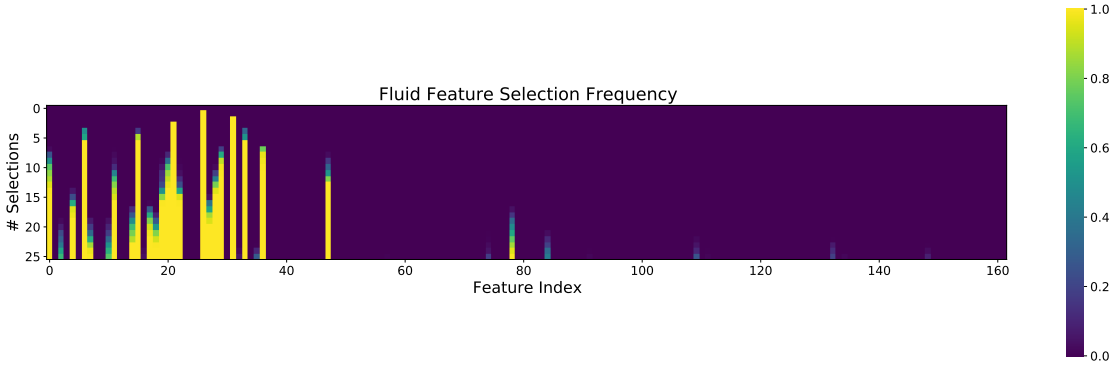


Figure G.5: Feature selection frequency for our greedy approach on the fluid dataset.

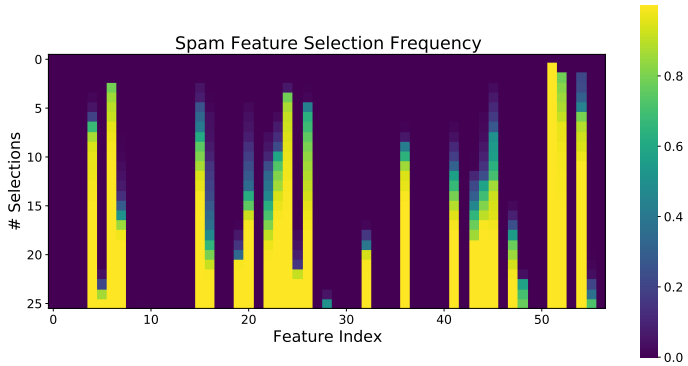


Figure G.6: Feature selection frequency for our greedy approach on the spam dataset.

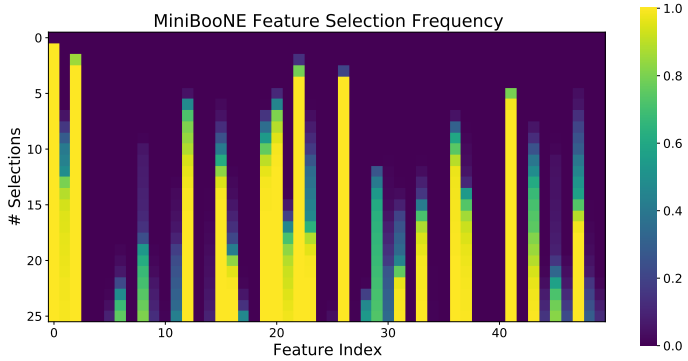


Figure G.7: Feature selection frequency for our greedy approach on the MiniBooNE dataset.

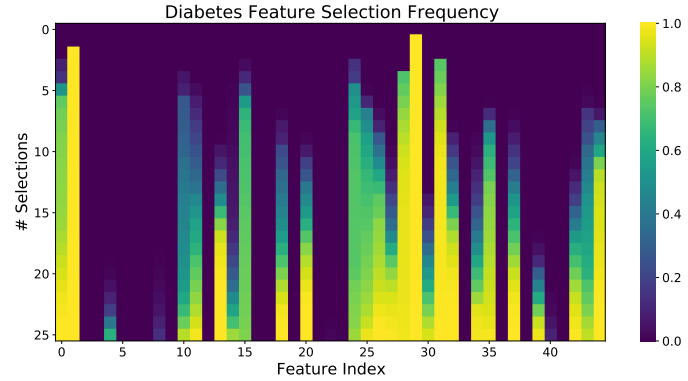


Figure G.8: Feature selection frequency for our greedy approach on the diabetes dataset.

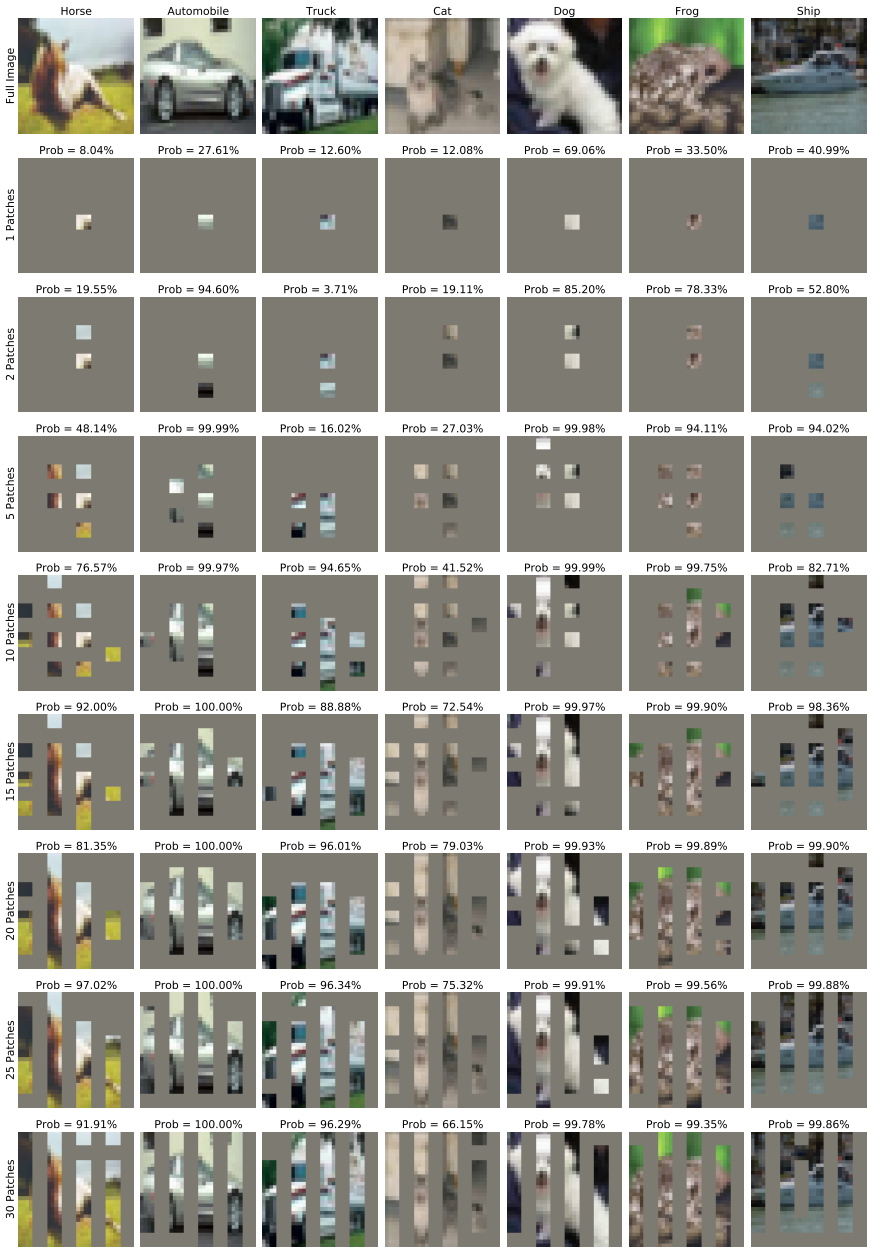


Figure G.9: CIFAR-10 predictions with different numbers of patches revealed by our approach.

Appendix H

ESTIMATING MUTUAL INFORMATION FOR DYNAMIC FEATURE SELECTION

H.1 Proofs

In this section, we re-state and prove our results from the main text.

H.1.1 Estimating Conditional Mutual Information

Lemma H.1. *When we use the Bayes classifier $p(\boldsymbol{y} \mid \boldsymbol{x}_S)$ as a predictor and ℓ is cross entropy loss, the incremental loss improvement is an unbiased estimator of the CMI for each $(x_S, \boldsymbol{x}_i)$ pair:*

$$\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})] = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S).$$

Proof. Consider the expected cross entropy loss given the prediction $p(\boldsymbol{y} \mid x_S)$:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}[\ell(p(\boldsymbol{y} \mid x_S), \boldsymbol{y})] &= -\sum_{y=1}^{K} p(\boldsymbol{y} = y \mid x_S) \log p(\boldsymbol{y} = k \mid x_S) \\ &= H(\boldsymbol{y} \mid x_S).\end{aligned}$$

Next, consider the loss given the prediction $p(\boldsymbol{y} \mid x_S, x_i)$, taken in expectation across $\boldsymbol{x}_i$ and $\boldsymbol{y}$'s conditional distribution $p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)$:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\ell(p(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i), \boldsymbol{y})] &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S} \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S,\boldsymbol{x}_i=x_i}[\ell(p(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i = x_i), \boldsymbol{y})] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}[H(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i = x_i)] \\ &= H(\boldsymbol{y} \mid x_i, \boldsymbol{x}_i).\end{aligned}$$

We therefore have the following expectation for the incremental loss improvement:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S,\boldsymbol{x}_i,\boldsymbol{y})] &= \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\ell(p(\boldsymbol{y}\mid x_S),\boldsymbol{y}) - \ell(p(\boldsymbol{y}\mid x_S,\boldsymbol{x}_i),\boldsymbol{y})] \\ &= H(\boldsymbol{y}\mid x_S) - H(\boldsymbol{y}\mid x_S,\boldsymbol{x}_i) \\ &= I(\boldsymbol{y};\boldsymbol{x}_i\mid x_S).\end{aligned}$$

Thus, the loss improvement $\Delta(x_S,\boldsymbol{x}_i,\boldsymbol{y})$ is an unbiased estimator of the CMI $I(\boldsymbol{y};\boldsymbol{x}_i\mid x_S)$. □

Theorem H.1. *When ℓ is cross entropy loss, the objectives eq.* 10.2 *and eq.* 10.3 *are jointly optimized by a predictor $f(x_S;\theta^*) = p(\boldsymbol{y}\mid x_S)$ and value network where $v_i(x_S;\phi^*) = I(\boldsymbol{y};\boldsymbol{x}_i\mid x_S)$ for $i\in[d]$.*

Proof. Similar to (Covert et al., 2023b), our proof considers both models' optimal predictions for each input. Beginning with the predictor, consider the output given the input x_S. The selections were made given only x_S, so observing this input conveys no information about $\boldsymbol{y}$ or the remaining features $\boldsymbol{x}_{[d]\setminus S}$. The expected loss is therefore

$$\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}[\ell(f(x_S;\theta),\boldsymbol{y})].$$

Assuming that ℓ is cross entropy loss, we can decompose the expected loss as follows:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}[\ell(f(x_S;\theta),\boldsymbol{y})] &= \sum_{y=1}^{K} p(\boldsymbol{y}=y\mid x_S)\log f_y(x_S;\theta) \\ &= \sum_{y=1}^{K} p(\boldsymbol{y}=y\mid x_S)\log p(\boldsymbol{y}=y\mid x_S)\frac{f_y(x_S;\theta)}{p(\boldsymbol{y}=y\mid x_S)} \\ &= H(\boldsymbol{y}\mid x_S) + D_{\mathrm{KL}}(p(\boldsymbol{y}\mid x_S)\mid\mid f(x_S;\theta)).\end{aligned}$$

Due to the non-negative KL divergence term, we see that the optimal prediction is $p(\boldsymbol{y}\mid x_S)$. We can make this argument for any input x_S, so we say that the optimal predictor is

$f(x_S; \theta^*) = p(\boldsymbol{y} \mid x_S)$ for all x_S. Notably, this argument does not depend on the selection policy: it only requires that the policy has no additional information about the response variable or unobserved features.

Next, we consider the value network while assuming that we use the optimal predictor $f(\boldsymbol{x}_S; \theta^*)$. Given an input x_S, we once again have no further information about $\boldsymbol{y}$ or $\boldsymbol{x}_{[d]\setminus S}$, so the expected loss is taken across the distribution $p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)$ as follows:

$$\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[(v(x_S; \phi) - \Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y}))^2 \right].$$

The expected loss can then be decomposed,

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[(v(x_S; \phi) - \Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y}))^2 \right] = \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} \left[(v(x_S; \phi) - \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} [\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})])^2 \right] \\ + \operatorname{Var} \left(\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})] \mid x_S \right),\end{aligned}$$

which reveals that the optimal prediction is $\mathbb{E}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})]$. Following Lemma 10.1, we know that this is equal to $I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$. And because we can make this argument for any x_S, we say that the optimal value network is given by $v(x_S; \phi^*) = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S)$. □

H.1.2 Prior Information

Before proving Theorem 10.2, we first present a preliminary result analogous to Lemma 10.1.

Lemma H.2. *When we use the Bayes classifier $p(\boldsymbol{y} \mid \boldsymbol{x}_S, \boldsymbol{z})$ as a predictor and ℓ is cross entropy loss, the incremental loss improvement is an unbiased estimator of the CMI for each $(x_S, z, \boldsymbol{x}_i)$ tuple:*

$$\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S} [\Delta(x_S, \boldsymbol{x}_i, z, \boldsymbol{y})] = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S, z).$$

Proof. The proof follows the same logic as Lemma 10.1, where we consider the expected loss before and after incorporating the additional feature $\boldsymbol{x}_i$. The only difference is that each

expectation must also condition on $\boldsymbol{z} = z$, so the terms to analyze are

$$\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S,z}[\ell(p(\boldsymbol{y} \mid x_S, z), \boldsymbol{y})]$$
$$\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S,z}[\ell(p(\boldsymbol{y} \mid x_S, z, \boldsymbol{x}_i), \boldsymbol{y})].$$

□

We now prove the main result for incorporating prior information.

Theorem H.2. *When ℓ is cross entropy loss, the objectives in eq.* 10.5 *are jointly optimized by a predictor $f(x_S, z; \theta^*) = p(\boldsymbol{y} \mid x_S, z)$ and value network where $v_i(x_S, z; \phi^*) = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S, z)$ for all $i \in [d]$.*

Proof. The proof follows the same logic as Theorem 10.1. For the predictor with input x_S, we can decompose the expected loss as follows:

$$\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S,z}[\ell(f(x_S, z; \theta), \boldsymbol{y})] = H(\boldsymbol{y} \mid x_S, z) + D_{\mathrm{KL}}(p(\boldsymbol{y} \mid x_S, z) \mid\mid f(x_S, z; \theta)).$$

This shows that the optimal predictor is $f(x_S, z; \theta^*) = p(\boldsymbol{y} \mid x_S, z)$. Next, assuming we use the optimal predictor, the value network's expected loss can be decomposed as follows:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S,z}\left[(v(x_S, z; \phi) - \Delta(x_S, \boldsymbol{x}_i, z, \boldsymbol{y}))^2\right] = &\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S,z}\left[(v(x_S, z; \phi) - \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S,z}[\Delta(x_S, \boldsymbol{x}_i, z, \boldsymbol{y})])^2\right] \\ &+ \mathrm{Var}\left(\Delta(x_S, \boldsymbol{x}_i, z, \boldsymbol{y})] \mid x_S, z\right).\end{aligned}$$

Based on this, Lemma H.2 implies that the optimal value network is $v(x_S, z; \phi^*) = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S, z)$. □

H.1.3 Regression Version

Before proving our main result for regression models, we first present a preliminary result analogous to Lemma 10.1.

Lemma H.3. *When we use the conditional expectation $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}_S]$ as a predictor and ℓ is mean squared error, the incremental loss improvement is an unbiased estimator of the expected reduction in conditional variance for each $(x_S, \boldsymbol{x}_i)$ pair:*

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})] &= \mathrm{Var}(\boldsymbol{y} \mid x_S) - \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i)] \\ &= \mathrm{Var}(\mathbb{E}[\boldsymbol{y} \mid x_S, \boldsymbol{x}_i] \mid x_S).\end{aligned}$$

Proof. Consider the expected loss given the prediction $\mathbb{E}[\boldsymbol{y} \mid x_S]$:

$$\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}\left[(\mathbb{E}[\boldsymbol{y} \mid x_S] - \boldsymbol{y})^2\right] = \mathrm{Var}(\boldsymbol{y} \mid x_S).$$

Next, consider the loss given the prediction $\mathbb{E}[\boldsymbol{y} \mid x_S, x_i]$, taken in expectation across $\boldsymbol{x}_i$ and $\boldsymbol{y}$'s conditional distribution $p(\boldsymbol{y}, \boldsymbol{x}_i \mid x_S)$:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}\left[(\mathbb{E}[\boldsymbol{y} \mid x_S, \boldsymbol{x}_i] - \boldsymbol{y})^2\right] &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S,\boldsymbol{x}_i=x_i}\left[(\mathbb{E}[\boldsymbol{y} \mid x_S, \boldsymbol{x}_i] - \boldsymbol{y})^2\right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i)].\end{aligned}$$

We therefore have the following expectation for the incremental loss improvement:

$$\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})] = \mathrm{Var}(\boldsymbol{y} \mid x_S) - \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}[\mathrm{Var}(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i)].$$

Using the law of total variance, we can simplify this difference as follows:

$$\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})] = \mathrm{Var}\left(\mathbb{E}[\boldsymbol{y} \mid x_S, \boldsymbol{x}_i] \mid x_S\right).$$

This provides a measure similar to the CMI: it quantifies to what extent different plausible values of $\boldsymbol{x}_i$ affect our best estimate for the response variable. □

We now present our main result for regression models.

Theorem H.3. *When ℓ is mean squared error, the objectives eq.* 10.2 *and eq.* 10.3 *are jointly optimized by a predictor $f(x_S;\theta^*) = \mathbb{E}[\boldsymbol{y} \mid x_S]$ and value network where for $i \in [d]$ we have*

$$v(x_S;\phi^*) = \mathrm{Var}(\mathbb{E}[\boldsymbol{y} \mid x_S, \boldsymbol{x}_i] \mid x_S).$$

Proof. We follow the same proof technique as in Theorem 10.1. The expected loss for the predictor with input x_S can be decomposed as follows,

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}[\ell(f(x_S;\theta),\boldsymbol{y})] &= \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}\left[(f(x_S;\theta)-\boldsymbol{y})^2\right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}\left[(f(x_S;\theta)-\mathbb{E}[\boldsymbol{y} \mid x_S])^2\right] + \mathrm{Var}(\boldsymbol{y} \mid x_S),\end{aligned}$$

which shows that the optimal predictor network is $f(x_S;\theta^*) = \mathbb{E}[\boldsymbol{y} \mid x_S]$. Assuming we use the optimal predictor, the expected loss for the value network can then be decomposed as

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}\left[(v(x_S;\phi)-\Delta(x_S,\boldsymbol{x}_i,\boldsymbol{y}))^2\right] = \mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}&\left[(v(x_S;\phi)-\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S,\boldsymbol{x}_i,\boldsymbol{y})])^2\right] \\ &+ \mathrm{Var}\left(\Delta(x_S,\boldsymbol{x}_i,\boldsymbol{y})] \mid x_S\right),\end{aligned}$$

which shows that the optimal value network prediction is $\mathbb{E}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}[\Delta(x_S,\boldsymbol{x}_i,\boldsymbol{y})]$. Lemma H.3 lets us conclude that the optimal value network is therefore $v(x_S;\phi^*) = \mathrm{Var}(\mathbb{E}[\boldsymbol{y} \mid x_S,\boldsymbol{x}_i] \mid x_S)$. □

H.1.4 Allowing a Variable Feature Budget

We first re-state our informal claim, and then introduce notation required to show a formal version.

Proposition H.1. *(Informal) For any feature budget k, the best policy to achieve this budget on average achieves lower loss than the best policy with a per-prediction budget constraint. Similarly, for any confidence level m, the best policy to achieve this confidence on average achieves lower cost than the best policy with a per-prediction confidence constraint.*

In order to account for a policy's stopping criterion, we generalize our earlier notation so that policies are functions of the form $\pi(x_S) \in \{0\} \cup [d]$, and we say a policy terminates (or stops selecting new features) when $\pi(x_S) = 0$. Given an input x, we let $S(\pi, x) \subseteq [d]$ denote the set of indices selected upon termination. The cost for this prediction is $c(\pi, x) = \sum_{i \in S(\pi,x)} c_i$, and there is also a notion of expected loss $\ell(\pi, x)$ that we define as follows:

$$\ell(\pi, x) = \mathop{\mathbb{E}}_{\boldsymbol{y} | x_{S(\pi,x)}} [\ell(f(x_{S(\pi,x)}), \boldsymbol{y})]. \tag{H.1}$$

For example, if ℓ is cross entropy loss and we use the Bayes classifier $f(x_S) = p(\boldsymbol{y} \mid x_S)$, we have $\ell(\pi, x) = H(\boldsymbol{y} \mid x_S)$; due to this interpretation of the expected loss, we refer to constraints on $\ell(\pi, x)$ as *confidence constraints*. For example, Chattopadhyay et al. (2023) suggests continuing to select features until $H(\boldsymbol{y} \mid x_S) \leq m$ for a confidence level m. In comparing policies, we must consider the trade-off between accuracy and feature cost, and we have two competing objectives – the average loss and the average cost:

$$\ell(\pi) = \mathbb{E}[\ell(\pi, \boldsymbol{x})] \qquad\qquad c(\pi) = \mathbb{E}[c(\pi, \boldsymbol{x})]. \tag{H.2}$$

Now, there are three types of policies we wish to compare: (i) those that adopt a budget constraint for each prediction, (ii) those that adopt a confidence constraint for each prediction, and (iii) those with no constraints. These classes of selection policies are defined as follows:

1. (Budget-constrained) These policies adopt a budget k that must be respected for each input x. That is, we have $c(\pi, x) \leq k$ for all x. This can be ensured by terminating the policy when the budget is exactly satisfied (Ma et al., 2019; Rangrej and Clark, 2021; Covert et al., 2023b) or when there are no more candidates that will not exceed the budget. Policies of this form are said to belong to the set Π_k.

2. (Confidence constrained) These policies adopt a minimum confidence m that must be respected for each input m. That is, we must have $\ell(\pi, x) \leq m$ for all x. Technically, we

may not be able to guarantee this for all predictions due to inherent uncertainty, so we can instead keep making predictions as long as the expected loss exceeds m (Chattopadhyay et al., 2023). Policies of this form are said to belong to the set Π_m.

3. (Unconstrained) These policies have no per-prediction constraints on the feature cost or expected loss. These policies are said to belong to the set Π, where we have $\Pi_k \subseteq \Pi$ and $\Pi_m \subseteq \Pi$.

With these definitions in place, we now present a more formal version of our claim.

Proposition H.2. *(Formal) For any average feature cost k, the best unconstrained policy achieves lower expected loss than the best budget-constrained policy:*

$$\min_{\pi \in \Pi : c(\pi) \leq k} \ell(\pi) \leq \min_{\pi \in \Pi_k} \ell(\pi). \tag{H.3}$$

Similarly for any average confidence level m, the best unconstrained policy achieves lower expected cost than the best confidence-constrained policy:

$$\min_{\pi \in \Pi : \ell(\pi) \leq m} c(\pi) \leq \min_{\pi \in \Pi_m} c(\pi). \tag{H.4}$$

In other words, for any desired average feature cost or confidence level, it cannot help to adopt that level as a per-prediction constraint. The best policy to achieve these levels *on average* can violate the constraint for some predictions, and as a result provide either a lower average cost or expected loss.

Proof. The proof of this claim relies on the fact that $\Pi_k \subseteq \Pi$ and $\Pi_m \subseteq \Pi$. It is easy to see that $\Pi_k \subseteq \{\pi \in \Pi : c(\pi) \leq k\}$. This implies the inequality in eq. H.3 because the right-hand side takes the minimum over a smaller set of policies. Similarly, it is easy to see that $\Pi_m \subseteq \{\pi \in \Pi : \ell(\pi) \leq m\}$, which implies the inequality in eq. H.4. □

H.2 Predictor Suboptimality

Consider a feature subset x_S where the ideal prediction from the Bayes classifier is $p(\boldsymbol{y} \mid x_S)$, but the learned classifier instead outputs $q(\boldsymbol{y} \mid x_S)$. The incorrect prediction can result in a skewed loss, which then provides incorrect labels to the value network $v(x_S; \phi)$. Specifically, the expected loss assuming many data points $(\boldsymbol{x}, \boldsymbol{y})$ such that $\boldsymbol{x}_S = x_S$ becomes

$$\mathop{\mathbb{E}}_{\boldsymbol{y}|x_S}\left[\ell(q(\boldsymbol{y} \mid x_S), \boldsymbol{y})\right] = H(\boldsymbol{y} \mid x_S) + D_{\mathrm{KL}}\left(p(\boldsymbol{y} \mid x_S) \mid\mid q(\boldsymbol{y} \mid x_S)\right). \tag{H.5}$$

The loss is therefore higher on average than it should be given the Bayes classifier, with the extra loss being equal to the KL divergence between the ideal and actual predictions. However, this does not imply that $v(x_S; \phi)$ systematically overestimates the CMI, because its labels are based on the expected loss *reduction*.

Consider that the above situation with incorrect predictions occurs not only for x_S, but also for all values of $\boldsymbol{x}_i$: that is, the classifier outputs $q(\boldsymbol{y} \mid x_S, x_i)$ rather than $p(\boldsymbol{y} \mid x_S, x_i)$ for each value x_i. Now, the expected loss reduction is the following:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{y},\boldsymbol{x}_i|x_S}\left[\Delta(x_S, \boldsymbol{x}_i, \boldsymbol{y})\right] = I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S) &+ D_{\mathrm{KL}}(p(\boldsymbol{y} \mid x_S) \mid\mid q(\boldsymbol{y} \mid x_S)) \\ &- \mathop{\mathbb{E}}_{\boldsymbol{x}_i|x_S}\left[D_{\mathrm{KL}}(p(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i) \mid\mid q(\boldsymbol{y} \mid x_S, \boldsymbol{x}_i))\right].\end{aligned} \tag{H.6}$$

This implies that given infinite data and a value network that perfectly optimizes its objective, the learned CMI estimates are biased according to a *difference* in KL divergence terms. Notably, the difference can be either positive or negative, so the CMI estimates can be incorrect in either direction. And intuitively, the bias shrinks to zero as the classifier approaches $p(\boldsymbol{y} \mid x_S)$ for all predictions.

H.3 Training Algorithm

The following algorithm summarizes our learning approach, where we jointly train the predictor and value networks according to the objectives in eq. 10.2 and eq. 10.3. We implemented it in PyTorch (Paszke et al., 2017) using PyTorch Lightning.[1] Note that the algorithm requires a dataset of fully observed $\boldsymbol{x}$ samples with corresponding labels $\boldsymbol{y}$.

Algorithm 4 is simplified to omit several details that we implement in practice. These details are discussed below.

Masked pre-training When pre-training the predictor $f(\boldsymbol{x}_S; \theta)$, we sample feature subsets as follows: we first sample a cardinality $\{0, \ldots, d\}$ uniformly at random, and we then sample the members of the subset at random. This distribution ensures even coverage of different subset sizes $|s|$, whereas treating each feature's membership as an independent Bernoulli variable biases the subsets towards a specific size.

Minibatching As is conventional in deep learning, we calculate gradients in parallel for multiple inputs. In Algorithm 4, this means that we take gradient steps calculated over (i) multiple data samples $(\boldsymbol{x}, \boldsymbol{y})$ and (ii) multiple feature budgets.

Learning rate schedule Rather than train with a fixed learning rate $\gamma > 0$, we reduce its value over the course of training. To avoid setting a precise number of epochs for each dataset, we decay the learning rate when the loss reaches a plateau, and we perform early stopping when the learning rate is sufficiently low. The initial learning rate depends on the architecture, but we use values similar to those used for conventional training (e.g., ViTs require a lower learning rate than CNNs or MLPs).

Annealing exploration probability Setting a large value for ϵ helps encourage exploration, but at inference time we set $\epsilon = 0$. To avoid the mismatch between these settings, we

[1] `https://www.pytorchlightning.ai`

sometimes anneal ϵ towards zero over the course of training. Specifically, we train the model with a sequence of ϵ values, warm-starting each model with the output from the previous value.

Parameter sharing As mentioned in Section 10.5, we sometimes share parameters between the predictor and value network. We implement this via a shared backbone, e.g., a sequence of self-attention layers in a ViT (Dosovitskiy et al., 2020). The backbone is initialized via the predictor pre-training with random masks, and it is then used for both $f(\boldsymbol{x}_S; \theta)$ and $v(\boldsymbol{x}_S; \phi)$ with separate output heads for each one.

Scaling value network outputs To learn the optimal value network outputs, it is technically sufficient to let the network make unconstrained, real-valued predictions. However, given that the true CMI values are non-negative, or $I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S) \geq 0$ for all $(x_S, \boldsymbol{x}_i)$, it is sensible to constrain the predictions: for example, we can apply a softplus output activation. Similarly, we know that the true CMI values are upper bounded by the current prediction entropy $H(\boldsymbol{y} \mid x_S)$ (Cover and Thomas, 2012). These simultaneous bounds can be summarized as follows:

$$0 \leq I(\boldsymbol{y}; \boldsymbol{x}_i \mid x_S) \leq H(\boldsymbol{y} \mid x_S).$$

To enforce both inequalities, we apply a sigmoid operation to the unconstrained value network prediction $v(x_S; \phi)$, and we multiply this by the empirical prediction entropy from $f(x_S; \theta)$. An ablation showing the effect of this approach is in Figure H.3.

Prior information We found that an issue with using prior information (as discussed in Section 10.5.2) is overfitting to $\boldsymbol{z}$. This is perhaps unsurprising, particularly when $\boldsymbol{z}$ is high-dimensional, because the same input is used repeatedly with different feature subsets $\boldsymbol{x}_S$ and the same label $\boldsymbol{y}$. To mitigate this, we applied the following simple fix: for the separate network that processes the prior variable $\boldsymbol{z}$, we detached gradients when using the learned representation to make classifier predictions, so that gradients are propagated only for the

value network's CMI predictions. An ablation demonstrating this approach is in Figure H.8.

Inference time. At inference time, we follow a similar procedure as in Algorithm 4 but with $\epsilon = 0$, so that we always make the most valuable selection. In terms of stopping criteria for making a prediction, we explore multiple approaches, as discussed in Section 10.5.3: (i) a budget-constrained approach with parameter k, (ii) a confidence constrained approach with parameter m, and (iii) a penalized approach with parameter λ. Our results are generated by evaluating a single learned policy with several values for each of these parameters. The range of reasonable values for the confidence parameter m and penalty parameter λ depend on the dataset, so these are tuned by hand.

H.4 Datasets

This section provides details about the datasets used in our experiments. The size of each dataset is summarized in Table H.1.

MNIST This is the standard digit classification dataset (LeCun et al., 1998). We downloaded it with PyTorch and used the standard train and test splits, with 10,000 training samples held out as a validation set.

Intubation This is a privately curated dataset from a university medical center, gathered over a 13-year period (2007-2020). Our goal is to predict which patients require respiratory support upon arrival in the emergency department. We selected 112 pre-hospital clinical features including dispatch information (injury date, time, cause, and location), demographic information (age, sex), and pre-hospital vital signs (blood pressure, heart rate, respiratory rate). The outcome is defined based on whether a patient received respiratory support, including both invasive (intubation) and non-invasive (BiPap) approaches. We excluded patients under the age of 18, and because many features represent one-hot encodings for categorical variables, we grouped them into 35 feature groups. Feature acquisition costs

were obtained by having a board-certified emergency physician estimate the relative cost of obtaining each feature. The dataset is not publicly available due to patient privacy concerns.

ROSMAP The Religious Order Study (ROS) and Memory Aging Project (MAP) Bennett et al. (2012a,b) are complementary epidemiological studies that enroll participants to study dementia. ROS is a logitudinal study that enrolls clergy without known dementia from across the United States, including Catholic nuns, priests, and brothers aged 65 years and older. Participants agree to annual medical and psychological evaluation and pledge their brain for donation. MAP is a longitudinal study that enrolls participants encompassing a wider community from 40 continuous care retirement facilities around the Chicago metropolitan area. Participants are without known dementia and agree to annual clinical evaluation and donation of brain, spinal cord and muscle after death. While entering the study, participants share demographic information (e.g. age, sex) and also provide their blood samples for genotyping. At each annual visit, their medical information is updated and they take a series of cognitive tests, which generate multiple measurements over time. This results in 46 different variables, grouped into 43 feature groups to account for one-hot encodings. The task is to predict dementia onset within the next three years given the current medical information and no prior history of dementia. In total, the data contains 3,194 individuals with between 1 and 23 annual visits. Following the preprocessing steps used in (Beebe-Wang et al., 2021), we applied a four-year sliding window over each sample, thereby generating multiple samples per participant. Each sample is split into an input window consisting of the current year visit t and a prediction window of the next three years $(t+1, t+2, t+3)$. To avoid overlap between the training, validation, or testing sets, we ensured that all samples from a single individual fell into only one of the data splits. Feature acquisition costs expressed in terms of time taken were borrowed from Beebe-Wang et al. (2021) for the cognitive tests and rough estimates were assigned to the remaining features using prior knowledge. We discarded the genotypic feature (APOE e4 allele) from the feature set since it is highly predictive of dementia and it is difficult to assign an appropriate cost. The dataset can be accessed at `https://dss.`

niagads.org/cohorts/religious-orders-study-memory-and-aging-project-rosmap/.

ImageNette and ImageNet-100 These are both subsets of the standard ImageNet dataset (Deng et al., 2009). ImageNette contains 10 classes and was downloaded using the Fast.ai deep learning library (Howard and Gugger, 2020), ImageNet-100 contains 100 classes and was downloaded from Kaggle (Ambityga), and in both cases we split the images to obtain train, validation and test splits. Images were resized to 224×224 resolution for both architectures we explored, ResNets (He et al., 2016b) and ViTs (Dosovitskiy et al., 2020).

MHIST The MHIST (**m**inimalist **hist**opathology) (Wei et al., 2021) dataset is an image classification dataset comprising 3,152 hematoxylin and eosin (H&E)-stained Formalin Fixed Paraffin-Embedded (FFPE) fixed-size images of colorectal polyps from patients at Dartmouth-Hitchcock Medical Center (DHMC). The task is to perform binary classification between hyperplastic polyps (HPs) and sessile serrated adenomas (SSAs), which is a challenging prediction task with significant variation in inter-pathologist agreement (Abdeljawad et al., 2015; Farris et al., 2008; Glatz et al., 2007; Khalid et al., 2009; Wong et al., 2009). HPs are typically benign, while SSAs are precancerous lesions that can turn into cancer if not treated promptly. The fixed-size images were obtained by scanning 328 whole-slide images and then extracting regions of size 224×224 representing diagnostically-relevant regions of interest for HPs or SSAs. For the ground truth, each image was assigned a gold-standard label determined by the majority vote of seven board-certified gastrointestinal pathologists at the DHMC. The dataset can be accessed by filling out the form at https://bmirds.github.io/MHIST/.

H.5 Models

Here, we briefly describe the types of models used for each dataset. The exploration probability ϵ for all models is set to 0.05 at the start with an annealing rate of 0.2.

Tabular datasets For all the tabular datasets, we use multilayer perceptrons (MLPs) with two hidden layers and ReLU non-linearity. We use 128 neurons in the hidden layers for the ROSMAP and Intubation datasets, and 512 neurons for MNIST. The initial learning rate is set to 10^{-3} at the start and we also use dropout with probability 0.3 in all layers to reduce overfitting (Srivastava et al., 2014). The value and predictor networks use separate but identical network architectures. The networks are trained on a NVIDIA RTX 2080 Ti GPU with 12GB of memory.

Image datasets: ResNet We use a shared ResNet-50 backbone (He et al., 2016b) for the predictor and value networks. The final representation from the backbone has shape 7×7, and the output heads for each network are specified as follows. The predictor head contains a Conv $\rightarrow$ Batch Norm $\rightarrow$ ReLU sequence followed by global average pooling and a fully connected layer. The value network head consists of an upsampling block with a transposed convolutional layer, followed by a 1×1 convolution and a sigmoid to scale the predictions (see Appendix H.3). The learning rate starts at 10^{-5}, and the networks are trained on a NVIDIA RTX 2080 Ti GPU with 12GB of memory.

Image datasets: ViT We use a shared ViT backbone (`vit_small_patch16_224`) (Dosovitskiy et al., 2020) for the predictor and value networks. We use ViT and ResNet backbones having a similar number of parameters for a fair comparison: ResNet-50 has approximately 23M parameters, and ViT-Small has 22M parameters. The predictor head contains a linear layer applied to the class token, and the value network head contains a linear layer applied to all tokens except for the class token, followed by a sigmoid function. When incorporating prior information, a separate ViT backbone is used for both the predictor and value networks to generate an embedding, which is then concatenated with the masked image embedding to get either the predicted CMIs or the class prediction. The learning rate starts at 10^{-5}, and the networks are trained on a NVIDIA Quadro RTX 6000 GPU with 24GB of memory.

H.6 Baselines

Here, we provide more details here on our baseline methods.

Concrete autoencoder This is a static feature selection method that optimizes a differentiable selection module within a neural network (Balın et al., 2019). The layer can be added at the input of any architecture, so we use this method for both tabular and image datasets. The original work suggested training with an exponentially decayed temperature and a hand-tuned number of epochs, but we use a different approach to minimize the tuning required for each dataset: we train with a sequence of temperature values, and we perform early stopping for each one based on the validation loss. We return the features that are selected after training with the lowest temperature, and we evaluate them by training a model from scratch with only those features provided.

EDDI This is a dynamic feature selection method that relies on a generative model to sample the unobserved features (Ma et al., 2019). We implement a PVAE to sample the unknown features, and these samples are used to estimate the CMI for candidate features at each selection step. We separately implement a classifier that makes predictions with arbitrary feature sets, similar to the one obtained after masked pre-training in Algorithm 4. We use this method only for our tabular datasets, as the PVAE is not expected to work well for images, and the computational cost at inference time is relatively high.

Probabilistic hard attention This method extends EDDI to work for images by imputing unobserved features within a lower dimensional, learned feature space (Rangrej and Clark, 2021). To ensure that the method operates on the same image regions as DIME, we implemented a feature extractor that separately computes embeddings for non-overlapping 14×14 patches, similar to a ViT (Dosovitskiy et al., 2020) or bag-of-features model (Brendel and Bethge, 2019). Specifically, our extractor consists of a single 16×16 convolutional layer, followed by a series of 1×1 convolutions. The features from each patch are aggregated by

a recurrent module, and we retain the same structure used in the original implementation.

Argmax Direct This is a dynamic feature selection method that directly estimates the feature index with maximum CMI. It is based on two concurrent works whose main difference is how gradients are calculated for the selector network (Chattopadhyay et al., 2023; Covert et al., 2023b); for simplicity, we only explore the technique based on the Concrete distribution from (Covert et al., 2023b). As a discriminative method, this baseline allows us to use arbitrary architectures and is straightforward to apply with either tabular or image datasets.

H.7 Additional Results

Here, we present the results of several additional experiments and ablations on the different datasets.

Figure H.1 shows the prediction calibration of the predictor network by plotting DIME's performance at different levels of confidence for a specific budget of $k = 15$, along with the density of the samples at those confidence levels across multiple datasets. This shows that the predictor network is well-calibrated and does not systematically overestimate or underestimate its predicted probabilities. Proper calibration is important to achieve accurate loss values, because these are then used to train the value network (see eq. 10.3).

Figure H.2 shows the calibration of the predicted CMIs by the value network for both a tabular and image dataset by plotting the difference in entropy or losses against the predicted CMI. A linear trend showcases that the CMIs predicted by the value network align well with the difference in either the entropy or the loss. Since we do not have ground truth CMI values to evaluate the accuracy of our value network, this serves a viable alternative for real-world datasets, and we can verify that the CMI predictions correctly represent the expected reduction in either loss or entropy.

Figure H.3 shows the effect of constraining the predicted CMIs using the current prediction entropy, as described in Appendix H.3. Without the constraint, there are some samples that have unrealistically high CMIs which are greater than the prediction entropy. After

applying the sigmoid activation on the value network predictions, this issue is corrected.

Figure H.4 shows multiple trials for the penalized policy on the tabular datasets. This provides a simple way to represent variability between trials when we cannot precisely control the budget between independent policies. Similarly, Figure H.5 and Figure H.6 show multiple trials while considering non-uniform feature costs. The relative results stay the same across all trials.

Figure H.7 shows the confidence distribution of full input predictions in the tabular datasets. Across all datasets, we observe that the model has high confidence in many of the samples, but there are some that remain uncertain even after observing all the features. This provides motivation for using the penalized approach, because a confidence-constrained approach could suffer here by expending the entire feature acquisition budget only to remain at high uncertainty.

Figure H.8 shows the effectiveness of detaching gradients for the predictor network of the sketch $\boldsymbol{z}$ in the histopathology dataset, as described in Appendix H.3. The penalized policy performs much better when we propagate gradients only for CMI predictions, which we attribute to reduced overfitting to the prior information. Figure H.9 compares the confidence-constrained stopping criterion with the other two approaches (penalized and budget-constrained) for two of the image datasets.

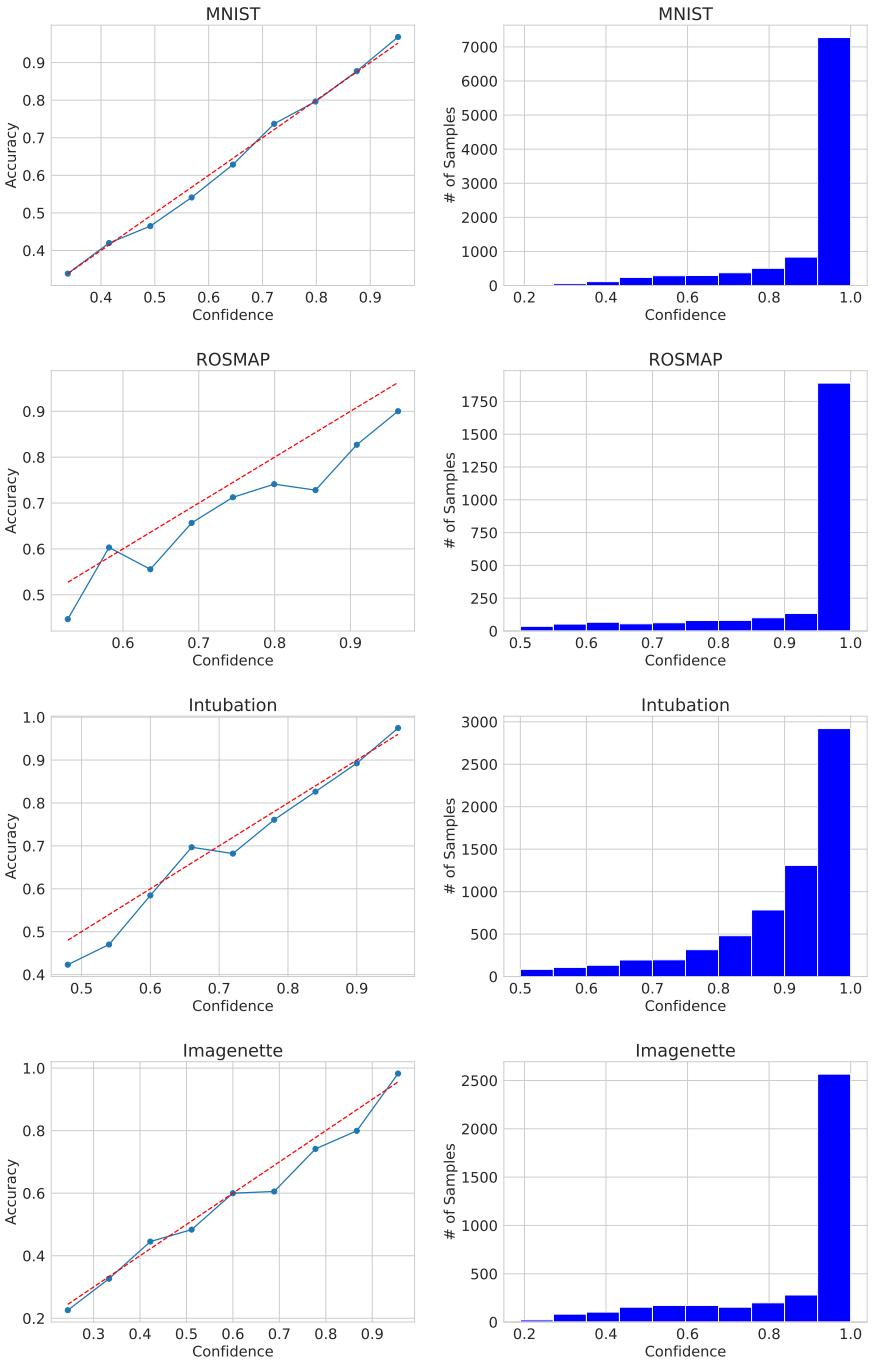


Figure H.1: Evaluation of prediction calibration for a fixed budget of $k = 15$. The left column shows the prediction calibration of the predictor network by plotting the accuracy for different confidence levels. The right column shows the distribution of confidences across all samples.

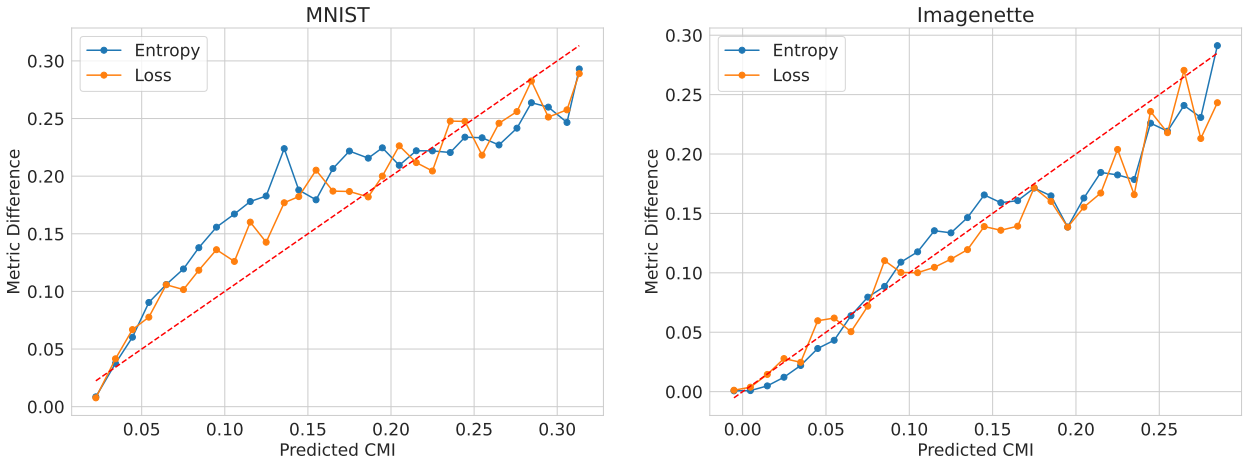


Figure H.2: Evaluation of CMI calibration. The x-axis shows different values for the predicted CMI throughout the selection process, and the y-axis shows the reduction in either loss or entropy after the corresponding feature is selected.

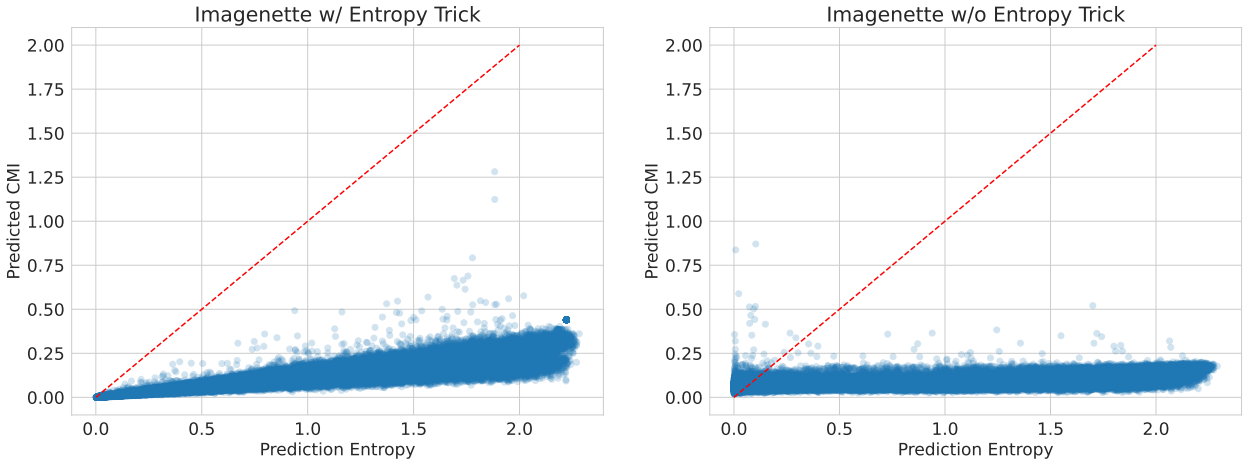


Figure H.3: Predicted CMIs with and without the entropy trick to scale value network outputs.

Algorithm 4: Training algorithm

Input: Data distribution $p(\boldsymbol{x}, \boldsymbol{y})$, learning rate γ, budget k, exploration $\epsilon \in (0, 1)$, costs $c \in \mathbb{R}_+^d$

Output: Predictor $f(\boldsymbol{x}_S; \theta)$, value network $v(\boldsymbol{x}_S; \phi)$

`// Prepare models`

initialize $v(\boldsymbol{x}_S; \phi)$, pre-train $f(\boldsymbol{x}_S; \theta)$ with random masks

while *not converged* **do**

`// Initialize variables`

initialize $S = \varnothing$, $\mathcal{L}_\theta = 0$, $\mathcal{L}_\phi = 0$

sample $x, y \sim p(\boldsymbol{x}, \boldsymbol{y})$

`// Initial prediction`

calculate $\hat{y}_{\text{prev}} = f(x_\varnothing; \theta)$

update $\mathcal{L}_\theta \leftarrow \mathcal{L}_\theta + \ell(\hat{y}_{\text{prev}}, y)$

while $\sum_{i \in s} c_i \leq k$ **do**

`// Determine next selection`

calculate $I = v(x_S; \phi)$

set $j = \arg\max_{i \notin s} I_i / c_i$ with probability $1 - \epsilon$, else sample j from $[d] \setminus S$

`// Update predictor loss`

update $S \leftarrow S \cup j$

calculate $\hat{y} = f(x_S; \theta)$

update $\mathcal{L}_\theta \leftarrow \mathcal{L}_\theta + \ell(\hat{y}, y)$

`// Update value loss`

calculate $\Delta = \ell(\hat{y}_{\text{prev}}, y) - \ell(\hat{y}, y)$

update $\mathcal{L}_\phi \leftarrow \mathcal{L}_\phi + (I_j - \Delta)^2$

set $\hat{y}_{\text{prev}} = \hat{y}$

end

`// Gradient step`

update $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}_\theta, \quad \phi \leftarrow \phi - \gamma \nabla_\phi \mathcal{L}_\phi$

end

Table H.1: Summary of datasets used in our experiments.

Dataset	# Features	# Feature Groups	# Classes	# Samples
MNIST	784	–	10	60,000
Intubation	112	35	2	65,515
ROSMAP	46	43	2	13,438
ImageNette	50,176	196	10	13,395
ImageNet-100	50,176	196	100	135,000
Histopathology	50,176	196	2	3152

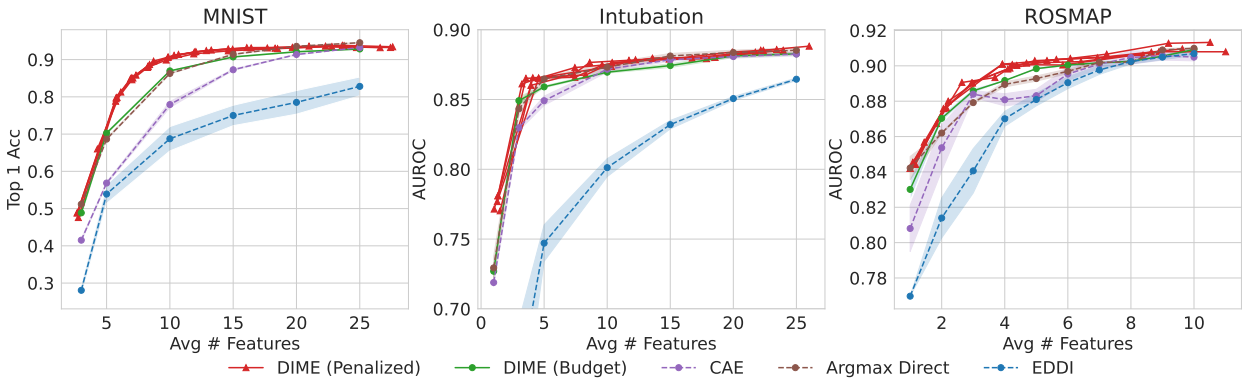


Figure H.4: Multiple trials using the penalized policy for tabular datasets. DIME remains the best method across five independent trials.

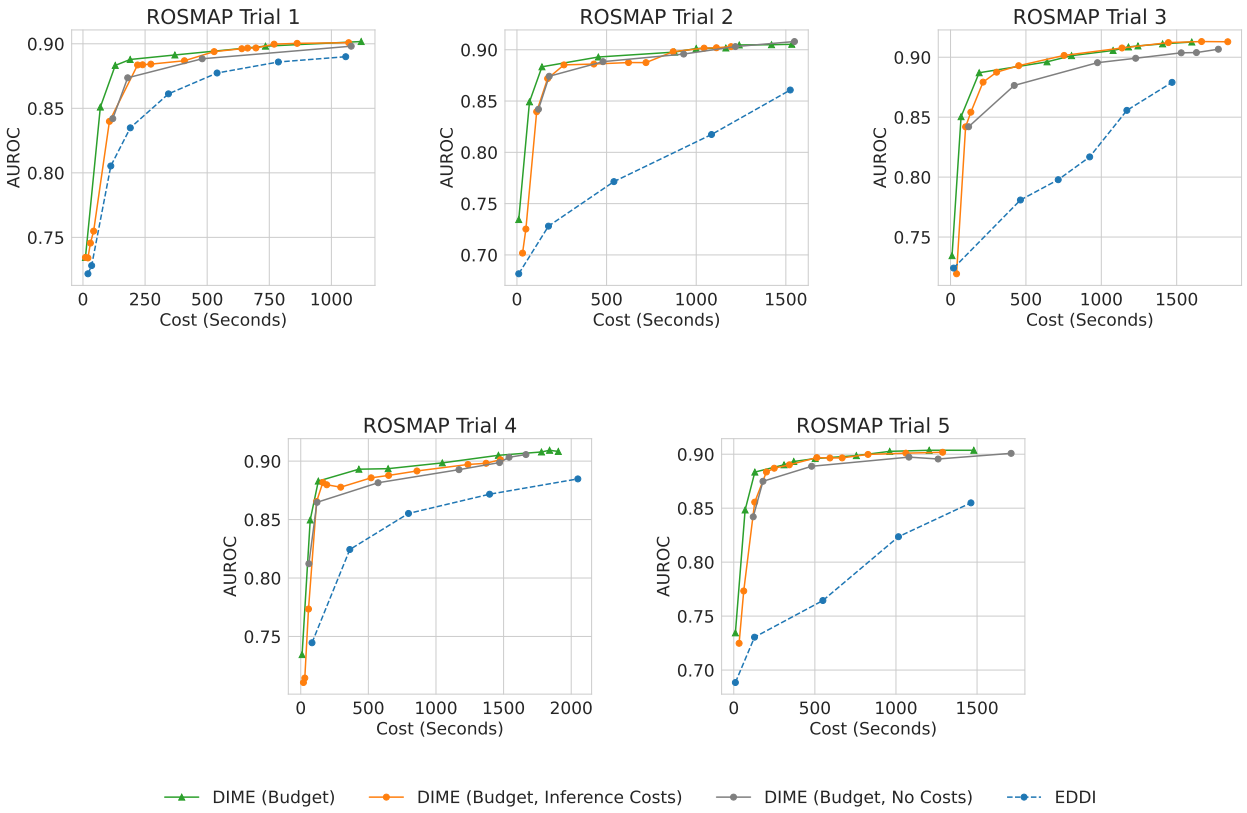


Figure H.5: Multiple trials when using non-uniform feature costs for ROSMAP.

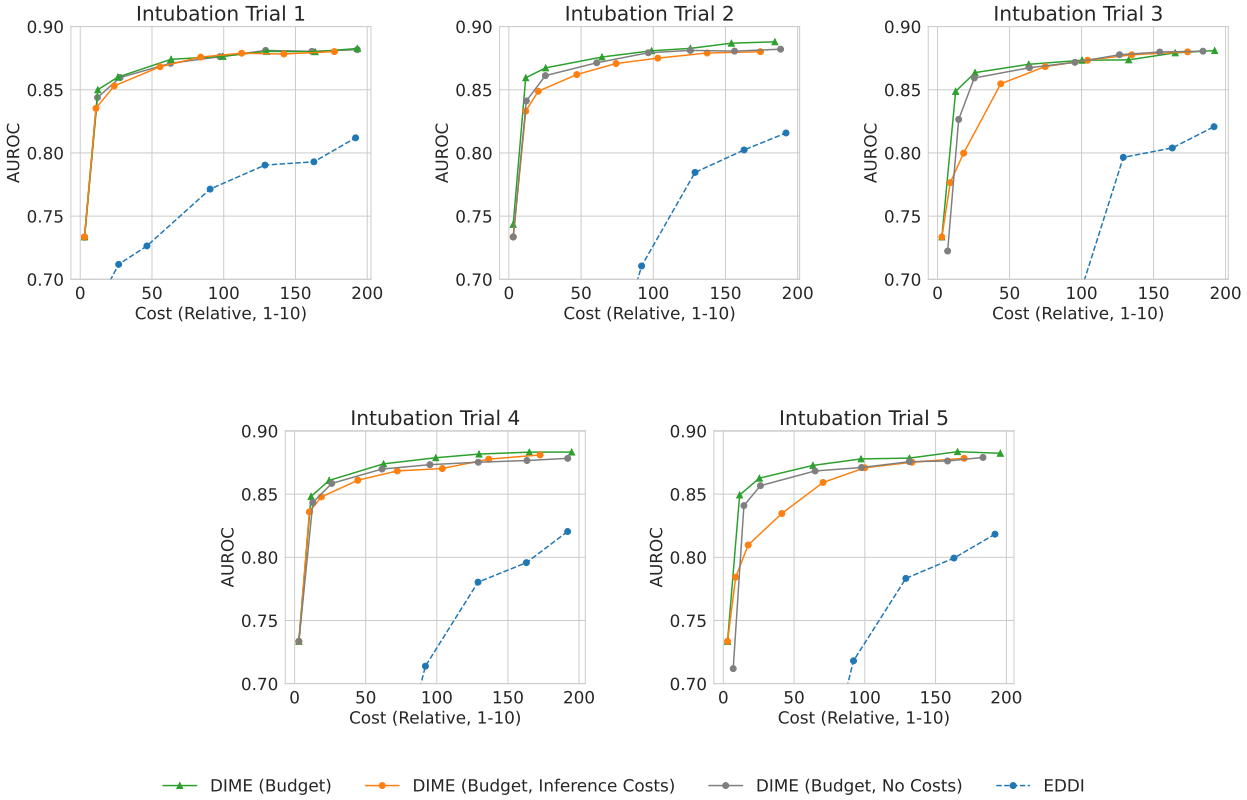


Figure H.6: Multiple trials when using non-uniform feature costs for Intubation.

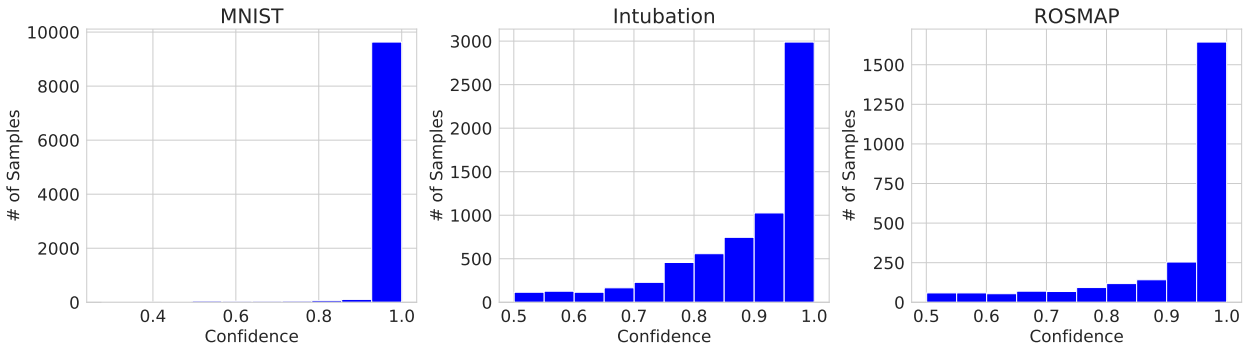


Figure H.7: Confidence distribution on full-input predictions across the tabular datasets.

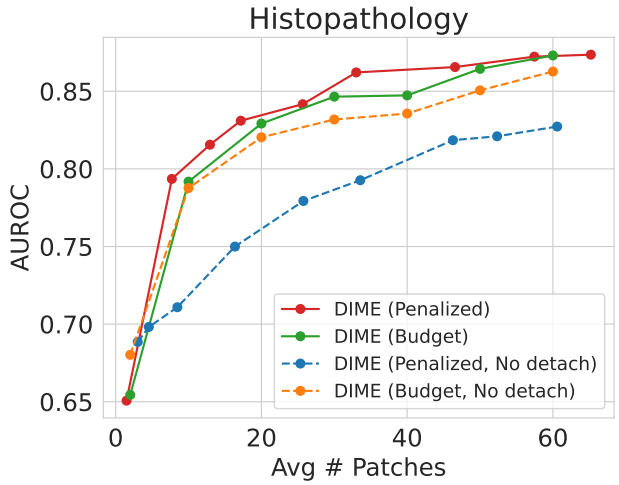


Figure H.8: Ablation of stop-gradients trick when using prior information for the histopathology dataset (Appendix H.3).

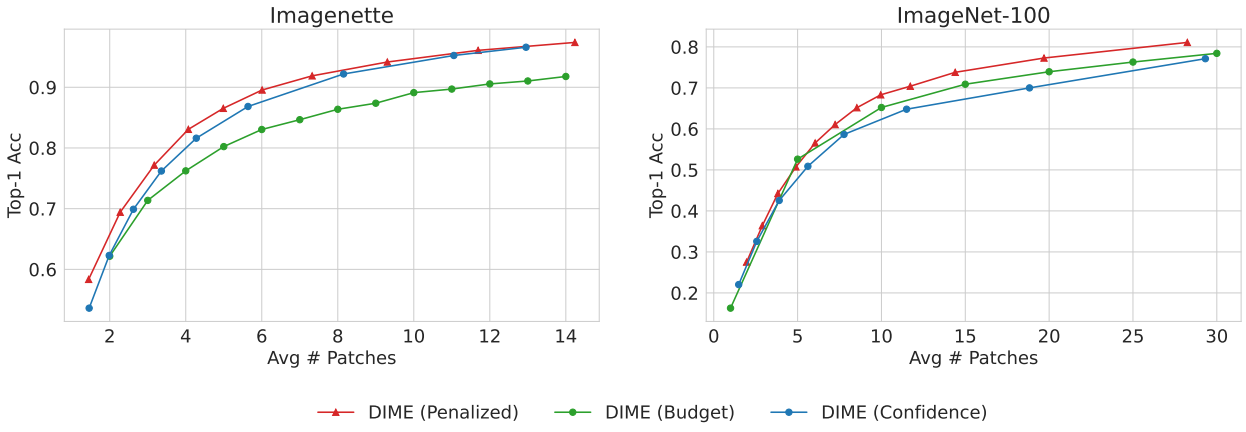


Figure H.9: Comparison of the budget-constrained and penalized stopping criteria with the confidence-constrained approach.

VITA

Ian Covert was born in San Francisco to Mary Connick and Derek Covert. He is currently a PhD candidate at the University of Washington, where he works under the supervision of Prof. Su-In Lee. His main research interest is creating highly performant machine learning systems that are well-understood by humans, both in terms of how they are learned and how they make decisions once trained.

During his PhD, Ian spent nine months as a student researcher at Google, where he worked on the healthcare team under Jiening Zhan and Ming Jack Po. He also spent a summer at Citadel Securities working as a quantitative researcher with Dianqi Li and Martin Lorilla. Prior to his PhD, he studied math, computer science and statistics at Columbia University, where he graduated summa cum laude and was elected to Phi Beta Kappa. His first research experience was at Columbia in Liam Paninski's lab, under the supervision of Uygar Sümbül. Prior to his undergraduate studies, Ian was a "lifer" at the French American International School in San Francisco, which he attended with his sister, Darcy Covert.

Outside of work, Ian is an avid tennis player. He also enjoys traveling, reading nonfiction books, cycling, playing squash and badminton, and spending time with friends and family.