Applications of Optimization and Machine Learning to Healthcare

Roman Levin

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

Doctor of Philosophy

University of Washington

2022

Reading Committee:
Aleksandr Aravkin, Chair
Minsun Kim
J. Nathan Kutz

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

Applications of Optimization and Machine Learning to Healthcare

Roman Levin

Chair of the Supervisory Committee:
Associate Professor Aleksandr Aravkin
Department of Applied Mathematics

As the field of healthcare becomes increasingly data-driven, optimization and machine learning methods provide the scientific community and practitioners with powerful tools to extract insights from the data with potential to improve clinical practice and inform public health efforts. However, medical applications pose a set of unique challenges. Medical data is often limited since accumulating large amounts of patient data with labels is difficult, especially for rare conditions or hospital-specific tasks. Additionally, the high stakes healthcare environment requires development of reliable explainability methods to ensure the safe application of machine learning models. On the other hand, abundant unlabelled data on the population level provides unique opportunities in public health.

In our work we leverage these opportunities and develop methods to address the above challenges. We first consider applications in radiation oncology and public health. We propose a novel optimization framework for multi-modality radiation therapy and discuss our work on leveraging unlabelled cell phone mobility data and manifold learning to gain insights into population behavior during the COVID-19 pandemic.

We then proceed to the methodological part of our work. We propose a novel parameter-saliency explainability method for deep neural networks which can be used to analyze model mistakes. After that, we present our work on transfer learning with deep tabular models where in a realistic medical setting we show that representation learning with neural networks

provides a definitive advantage over the traditionally dominant gradient boosted decision tree tabular methods when downstream data is limited. Finally, we present an application of deep tabular models to improve patient-specific quality assurance in radiation oncology.

TABLE OF CONTENTS

Page

List of Figures iv

List of Tables xii

Chapter 1: Introduction 1
1.1 Organization 2

Chapter 2: Multi-Modality Radiotherapy Optimization 4
2.1 Introduction 4
2.2 Multi-Modality Treatment Planning Optimization Model 8
2.3 Optimization Algorithm 12
2.4 Numerical Simulations 23
2.5 Conclusion 31

Chapter 3: Cell phone mobility data and manifold learning: Insights into population behavior during the COVID-19 pandemic 33
3.1 Introduction 33
3.2 Data 35
3.3 Methods 38
3.4 Results 40
3.5 Discussion and Conclusion 51

Chapter 4: Where do Models go Wrong? Parameter-Space Saliency Maps for Explainability 55
4.1 Introduction 55
4.2 Related Work 57
4.3 Method 58
4.4 Experiments 62

4.5 Discussion 71

Chapter 5: Transfer Learning with Deep Tabular Models 73
5.1 Introduction 73
5.2 Related Work 75
5.3 Transfer Learning Setup in Tabular Domain 78
5.4 Results for Transfer Learning with Deep Tabular Models 81
5.5 Self-Supervised Pre-training 83
5.6 Aligning Upstream and Downstream Feature Sets with Pseudo-Features . . . 85
5.7 Discussion 86

Chapter 6: Improving Patient-Specific Quality Assurance: Radiotherapy Plan Failure Prediction with Deep Tabular Models 89
6.1 Introduction 89
6.2 Materials and Methods 92
6.3 Results 94
6.4 Discussion 96
6.5 Conclusion 97

Bibliography 98

Appendix A: Multi-Modality Radiotherapy Optimization 122
A.1 Proximal Operator Calculation 122
A.2 Projection Calculation 123

Appendix B: Cell phone mobility data and manifold learning: Insights into population behavior during the COVID-19 pandemic 126
B.1 Linear dimensionality reduction and clustering 126
B.2 Nonlinear Dimensionality Reduction Methods 127
B.3 Robustness of GMM fitting 128
B.4 GMM Model and Uncertainty Quantification 128
B.5 GMM Model Selection 129
B.6 Altering the Number of Clusters and Continuous Colormap 129
B.7 Clustering in metropolitan areas: Georgia and California 130
B.8 Response Speed Distributions 130

Appendix C: Where do Models go Wrong? Parameter-Space Saliency Maps for Explainability 141
C.1 Additional Experiments 141
C.2 Implementation details 151
C.3 Limitations and Impact 152

Appendix D: Transfer Learning with Deep Tabular Models 161
D.1 Limitations and Impact 161
D.2 Experimental Details. 161
D.3 Hyperparameter Tuning 166
D.4 Additional Results 173

Appendix E: Improving Patient-Specific Quality Assurance: Radiotherapy Plan Failure Prediction with Deep Tabular Models 182
E.1 Hyperparameter Search Spaces 182

LIST OF FIGURES

Figure Number Page

2.1 **Introduction to radiation therapy.** Left: Elekta medical linear accelerator. Right: Percent depth dose comparison between X-rays and protons. . . 5

2.2 **Craniospinal irradiation patient plans.** Top: Isodose distributions with a single posterior-anterior proton beam. Bottom: Isodose distributions with a single posterior-anterior photon beam. Both plans are normalized to 3600 cGy at 100 %. 6

2.3 **Phantom geometry** 25

2.4 **Value function optimization.** Left: The surface represents $V(N_1, N_2)$ computed by brute force. The dots represent iterates starting from four different initial guesses. Right: Level sets of $V(N_1, N_2)$ and the iterates. 31

3.1 **Data-driven analysis overview.** A.) Example CBGs on the map of Washington. B.) Mobility time-series of the example CBGs. C.) Mobility time series are aggregated in a matrix form. D.) 3D visualization of the 14D Laplacian Eigenmaps embedding of the data with clusters highlighted in color. Large dots represent the example CBGs. E.) Clusters plotted on the Washington map. F.) Average mobility time-series for every cluster. 36

3.2 **Results are consistent across four states.** Column i. presents the 3D Laplacian Eigenmap visualization of the data manifold. Column ii. shows geographic maps. Column iii. presents average mobility time series for each cluster. Clusters are highlighted in color. It is noted that clusterig was done in 14D (optimal) embedding space. 43

3.3 **Clustering in metropolitan areas.** Section A: clustering in Washington, Section B: clustering in Texas. 45

3.4 **Distribution of stay-at-home behavior and demographic covariates by cluster.** The boxplots present the interquartile range (boxes) and median values (center horizontal lines) of the covariate values for CBGs in each of the five clusters. Whiskers span the 95% range. The "mean stay at home" fraction of a CBG is the mean of the daily percent of mobile devices that stayed completely at home during the time period analyzed. 46

3.5 **Proportions of census block groups in each cluster by population characteristics associated with geographic mobility.** The fraction of a CBG's population associated with the characteristic is plotted on the x-axis. CBGs are partitioned into 10 equally-spaced bins, defined by the proportion of each CBG's population having the characteristic in 10% increments. The numbers of CBGs belonging to each bin are printed along the top of each panel. The proportion of CBGs in each cluster is plotted as vertically stacked bars for each bin (with cluster A in dark orange on the bottom through cluster E in purple on top). 47

3.6 **Clusters in the Seattle metropolitan area.** Census block group boundaries are outlined. CBGs belonging to clusters D and E are highlighted in dark blue and purple, respectively. 48

3.7 **The fraction of devices that are *only* away from their homes each day.** The medians and inter-quartile range of each day's values are shown for each cluster in Washington State. 50

4.1 **Filter-wise parameter saliency profile.** ResNet-50 filter-wise saliency profile (without standardization) averaged over samples in ImageNet validation set. The filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis. 56

4.2 **Standardized filter-wise saliency profiles, correctly vs incorrectly classified samples.** Top: Standardized saliency profiles averaged over correctly classified samples in the ImageNet validation set. Bottom: Standardized saliency profiles averaged over incorrectly classified samples in the ImageNet validation set. On both panels, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis. Both profiles are generated on ResNet-50. 58

4.3 **Effect of turning salient filters off.** (a) Change in incorrect class confidence score. (b) Change in true class confidence score. (c) Percentage of samples that were corrected as the result of pruning filters. These trends are averaged across all images misclassified by ResNet-50 in the ImageNet validation set. The error bars represent 95% bootstrap confidence intervals. . 60

4.4 **Examples of nearest neighbors in parameter saliency space (from ImageNet).** 63

4.5 **Neighbors in parameter saliency space found using only early or only deep layers.** The reference image is in the first column. Images in the top row resemble the reference image in the saliency on early layers of VGG-19, and images in the bottom row are found using deeper layers. . . . 65

4.6 **Effect of updating a limited number of filters.** (a) Percentage of misclassified samples corrected after fine-tuning. (b) Average percentage of nearest neighbors that are also corrected after fine-tuning. (c) Average change in the confidence score of the true class among nearest neighbors. The horizontal dashed line in each plot is the effect of updating the entire network. 66

4.7 **Interaction between input features and salient filters.** (a) Reference image of "great white shark" misclassified by ResNet-50 as "killer whale" with confidence scores. (b) Input-space saliency visualization. Pixels that cause the top 10 salient filters to have high saliency. (c) Change in saliency values of the erroneous filters across masking experiments. The vertical bars represent the standard deviation of the change across 10 most salient filters. (d)-(f) Masking experiments. 67

4.8 **Different types of network mistakes.** All of the presented images are misclassified by ResNet-50. The correct class label is specified in the top row and the incorrect class label – in the bottom row of the subcaption on each panel. (a)-(b) The target object is confused with another object in the image. (c) A regular mistake. The salient pixels are focused on the target object features which confuse the network. (d) Background features confuse the network. (e) An example of a noisy label where the network is "more correct" than the target label. These are examples where masking top 5% of the salient pixels corrects the misclassification. 69

5.1 **Tabular transfer learning pipeline with MetaMIMIC.** We pre-train deep tabular neural networks on abundant upstream patient data with 11 diagnosis targets via multi-label classification. Then, we fine-tune the pre-trained models on limited downstream data with similar features to predict the new target diagnosis. 77

5.2 **Average model ranks across all downstream tasks.** Deep tabular models and GBDT performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks. FT-Transformer fine-tuned end-to-end outperforms all GBDT models, including GBDT with stacking, at all data levels. MLP is highly competitive in low data regimes. 81

5.3 **Comparison of supervised and self-supervised pre-training strategies for FT-Transformer.** The left panel illustrates end-to-end fine-tuning with linear head and the right plot illustrates end-to-end fine-tuning with MLP head, the two most effective strategies for FT-Transformer. Within each panel, columns represent pre-training strategies and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. Contrastive pre-training outperforms from-scratch trained models, while MLM pre-training is not effective. Supervised pre-training outperforms all self-supervised pre-training strategies in our experiments. 83

5.4 **Pseudo-Feature method for aligning upstream and downstream feature sets.** Left: Diagram illustrating strategy for handling mismatches between the upstream and downstream features. When upstream data is missing a feature present in downstream data, it is predicted with a model pre-trained on upstream data and fine-tuned to predict the new feature on the downstream data. Right top: Scenario with missing feature in the upstream data. Comparison of ranks of FT-Transformer model trained on data with missing feature, with pseudo-feature and with the original feature. Right bottom: Scenario with missing feature in the downstream data. Comparison of ranks of FT-Transformer model trained and fine-tuned on data with missing feature, fine-tuned with pseudo and with original feature. In both scenarios, using the pseudo-feature is better than training without the feature but worse than worse than the original ground truth feature values. 88

6.1 **Patient-level ROC Curves.** (a) FT-Transformer (b) CatBoost (c) XGBoost. The error bars represent the standard error across 5 seeds. The positive label corresponds to plan failure. 96

B.1 **Linear Dimensionality Reduction Performance.** Left: fraction of explained variance vs. number of PCA components for every state. Right: 3D visualization of PCA projection of the mobility time series data for Washington State with 11 clusters highlighted in color. Clustering was done in 8D PCA space. 131

B.2 **Nonlinear Dimensionality Reduction Embeddings and Corresponding Maps for Washington State.** Top Left: Laplacian Eigenmaps, Top Middle: Locally Linear Embedding, Top Right: Isomap. Bottom Left: Laplacian Eigenmaps clustering map, Bottom Middle: Locally Linear Embedding clustering map, Bottom Right: Isomap clustering map. 131

B.3 **Optimal Embedding Dimensionality for Washington State.** Left: Laplacian Eigenmaps (optimal dimensionality 14), Middle: Locally Linear Embedding (optimal dimensionality 12), Right: Isomap (optimal dimensionality 16). 132

B.4 **Laplacian Eigenmaps Hyperparameter Robustness.** Each panel presents a 3D Laplacian Eigenmaps embedding of Washington state mobility data for `n_neighbors` in $\{20, 30, 40, 50\}$ respectively. 132

B.5 **Optimal Laplacian Eigenmap Embedding Dimensionality for Every State.** Each panel presents trustworthiness vs number of Laplacian Eigenmap components for Washington (optimal dimensionality 14), Georgia (optimal dimensionality 14), Texas (optimal dimensionality 14), and California (optimal dimensionality 14 or 44) respectively. 132

B.6 **Comparison of two trustworthiness knee points for California.** Top: Clustering results using 14D Laplacian Eigenmap embedding (3D illustration of the embedding, geographic map and average mobility time series per cluster with clusters highlighted in color). Bottom: Clustering results using 44D Laplacian Eigenmap embedding (3D illustration of the embedding, geographic map and average mobility time series per cluster with clusters highlighted in color). 133

B.7 **Nonconvexity robustness.** Different types of GMM clustering results obtained by refitting GMM several times. 133

B.8 **GMM model selection for every state based on BIC.** Top: BIC curves for different parametrizations of the GMM model as described in [179]. Bottom: Optimal number of GMM components identified using knee-point detection on the best BIC curve for Washington (optimal number of clusters 4), Georgia (optimal number of clusters 5), Texas (optimal number of clusters 4), and California (optimal number of clusters 4) respectively. 134

B.9 **Clustering with the optimal number of clusters for every state** . . . 135

B.10 **Bigger number of clusters results in finer partitioning of the embedding.** Panels present clustering for the number of clusters in $\{4, 5, 6, 7, 8\}$ respectively. 135

B.11 **Continuous Colormap.** Smooth transition across urban, periurban, suburban, and rural areas in Washington, Georgia, Texas, and California. 136

B.12 **Clustering in metropolitan areas in Georgia** 137

B.13 **Clustering in metropolitan areas in California** 138

B.14 **Response Speed Distributions.** Top: The response speed is quantified by the slope of a linear fit of the CBG mobility time series during the transition period of March 10 – March 31, dashed line represents that linear fit for an example CBG. Bottom: Response speed distributions for every state. 139

B.15 **The fraction of devices that are *only* away from their homes each day.** The medians and inter-quartile range are shown for each cluster. 140

C.1 **Filter-wise saliency profiles for other architectures.** (a) VGG-19 saliency profile (without standardization). (b) Inception v3 saliency profile (without standardization). (c) DenseNet saliency profiles (without standardization). In each panel the filter-wise saliency profile is averaged over the ImageNet validation set. In every panel, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis. 142

C.2 **Standardized saliency profiles averaged over correctly vs incorrectly classified samples.** (a) VGG-19 saliency profiles. (b) Inception v3 saliency profiles. (c) DenseNet saliency profiles. In each panel, the top row presents the standardized saliency profiles averaged over correctly classified samples and the bottom row shows standardized saliency profiles averaged over incorrectly classified samples. On every panel, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis. 154

C.3 **Examples of nearest neighbors in the feature representation space (from ImageNet).** 155

C.4 **CIFAR-10 examples of nearest neighbors in parameter saliency space**. On CIFAR-10 images that cause similar filters to malfunction are often misclassified in a similar way. 155

C.5 **ImageNet examples of nearest neighbors in parameter saliency space**. In every panel, the reference image is in the left column and its nearest neighbors are in the right column. Panels are captioned by the true label of their reference image. 155

C.6 **Effect of updating a small number of filters on VGG-19.** (a) Percentage of samples that are corrected after fine-tuning. (b) Average percentage of nearest neighbors that are also corrected after fine-tuning. (c) Average change in the confidence score of the true class among nearest neighbors. The horizontal line in each plot is the effect of updating the entire network. . . . 156

C.7 **Effect of randomly perturbing filters.** (a) Change in incorrect class confidence score. (b) Change in true class confidence score. (c) Percentage of samples that were corrected as the result of pruning filters. These trends are averaged across all images misclassified by ResNet-50 in the ImageNet validation set. The error bars represent 95% bootstrap confidence intervals. . 156

C.8 **Masking non-salient parts of the image.** (a) Reference image of "great white shark" misclassified by the model as "killer whale" and the corresponding confidence scores. (b) Pixels that cause the top 10 most salient filters to have high saliency. (c) Masked (non-salient) human. (d) Masked non-salient water region. 157

C.9 **Sample "great white shark" images with boat and seal.** The salient region from the case study image pasted onto other "great white shark" images. 157

C.10 **ImageNet examples of "killer whale".** 158

C.11 **Pixels responsible for mistakes focused on the target object.** (a)-(b) Masking the salient pixels corrects the misclassification where masking confusing features (e.g. dog ears or spot patterns) helps distinguish animals. (c)-(d) Masking the salient pixels results in correct class confidence decrease, when the salient pixels are densely focused on the target object. The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel. 158

C.12 **Comparison to GradCAM.** Top row: original image. Middle row: GradCAM input-space saliency map for the predicted label. Bottom row: our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network). The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel. 159

C.13 **Comparison to GradCAM.** Top row: original image. Middle row: GradCAM input-space saliency map for the predicted label. Bottom row: our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network). The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel. 160

C.14 **Sanity checks.** (a) No randomization of ResNet-50. (b) Only stage 4 of ResNet-50 is randomized. (c) Stages 3-4 of ResNet-50 are randomized. (d) Stages 2-4 of ResNet-50 are randomized. (e) The entire ResNet-50 is randomized. 160

D.1 **Average model ranks including TabTransformer.** Deep tabular models and GBDT performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks. 173

D.2 **Model-wise ranks for GBDT and neural networks.** Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is computed across training setups for each model and is averaged across all downstream tasks. 174

D.3 **Average ROC-AUC improvements over Catboost baseline.** Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. ROC-AUC improvement is computed as difference in ROC-AUC with Catboost model and is averaged across all downstream tasks. 175

LIST OF TABLES

Table Number Page

2.1 **Comparison of various radiation types.** All are relative to photon external beam radiotherapy, which is currently the most widely used radiation type in practice. 7

2.2 **Constraint type and tolerance biological effect for organs-at-risk (OAR) used in treatment planning.** 25

2.3 **Optimal BE improvement with various tumor doubling time.** Parameters are fixed at $\alpha_2 = 0.35$ Gy$^{-1}, r = 1.0$. 27

2.4 **Optimal BE improvement with various α_2.** Parameters are fixed at $r = 1, T_d = 5$ days. 28

2.5 **Optimal BE improvement with various r.** Parameters are fixed at $T_d = 5$ days and $\alpha_2 = 0.35/\text{Gy}^{-1}$. 28

2.6 **Optimal BE improvement for a range of tumor doubling times when an extra margin to the tumor is used for M_2.** Parameters are fixed at $\alpha_2 = 0.35$ Gy$^{-1}, r = 1.0$ 29

2.7 **Optimal BE improvement with various α_2 when an extra margin to the tumor is used for M_2.** Parameters are fixed at $r = 1, T_d = 5$ days. . . 29

2.8 **Optimal BE improvement with various r when an extra margin to the tumor is used for M_2.** Parameters are fixed at $T_d = 5$ days and $\alpha_2 = 0.35/\text{Gy}^{-1}$. 30

3.1 **Number of census block groups (CBGs) in each cluster.** 41

6.1 **Regression results.** Rows correspond to models and columns correspond to regression metrics. 94

6.2 **Classification results.** Rows correspond to models and columns correspond to classification metrics. 95

B.1 **Quartiles of the cluster assignment uncertainty based on a GMM with 11 clusters and the SVD with 8 modes.** 126

B.2 **Quartiles of uncertainty of the cluster assignment based on GMM with 5 clusters.** 127

D.1 **Optuna hyperparameter search space and default configuration for FT-Transformer** . . . 167

D.2 **Optuna hyperparameter search space and default configuration for ResNet** . . . 167

D.3 **Optuna hyperparameter search space and default configuration for MLP** . . . 168

D.4 **Optuna hyperparameter search space and default configuration for TabTransformer** . . . 168

D.5 **Optuna hyperparameter search space for Catboost** . . . 169

D.6 **Optuna hyperparameter search space for XGBoost** . . . 169

D.7 **Comparison of tuned and default configurations of GBDT models.** The table displays ranks computed pair-wise for default/tuned configurations of XGBoost and Catboost models. . . . 170

D.8 **Comparison of tuned and default configurations of deep tabular baselines.** The table displays ranks computed pair-wise for default/tuned configurations of deep tabular neural networks. . . . 170

D.9 **Comparison of tuned and default configurations of deep tabular neural networks with transfer learning.** The table displays ranks computed pair-wise for default/tuned configurations of deep tabular models fine-tuned with 4 different transfer-learning setups. . . . 172

D.10 **ROC-AUC scores for all models for "Diabetes" and "Hypertensive" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. . . . 176

D.11 **ROC-AUC scores for all models for "Ischematic" and "Heart" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. . . . 177

D.12 **ROC-AUC scores for all models for "Overweight" and "Anemia" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. 178

D.13 **ROC-AUC scores for all models for "Respiratory" and "Hypotension" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. 179

D.14 **ROC-AUC scores for all models for "Lipoid" and "Atrial" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. 180

D.15 **ROC-AUC scores for all models for "Purpura" and "Alcohol" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. 181

E.1 **Optuna hyperparameter search space and default configuration for FT-Transformer** 182

E.2 **Optuna hyperparameter search space for Catboost** 183

E.3 **Optuna hyperparameter search space for XGBoost** 183

ACKNOWLEDGMENTS

Graduate school is the time to define oneself as a scientist as it provides endless opportunities for exploration, much more than any other time in one's career. I am deeply thankful to my advisor, Sasha Aravkin, for giving me complete academic freedom and for his guidance, support, and encouragement during both my challenging and joyful times in graduate school.

I would also like to thank my incredible mentors. I am thankful to Minsun Kim for her mentorship, support, research guidance and a very enjoyable collaboration in radiation oncology. I am beyond grateful to Joshua Proctor for being a wonderful mentor during my summer internship at the Institute for Disease Modeling (IDM), for teaching me so much and always being there for me during a pivotal time in my academic career. I owe my thanks to Micah Goldblum and Tom Goldstein for introducing me to the world of deep learning, for so many teaching and mentoring moments, and for an amazing collaboration. I would like to thank Nathan Kutz for his endless encouragement and support in critical moments of my PhD. I would like to thank Maryam Fazel for working with me on my supervisory committee and for her valuable suggestions and interesting questions both at my general exam and my defense.

I also owe a big thank you to my collaborators and co-authors. I am thankful for exciting and fruitful collaborations to Dennis Chao and Edward Wenger of IDM and to the team at the University of Maryland: Valeriia Cherepanova, Manli Shu, Eitan Borgnia, Avi Schwarzschild, Arpit Bansal, and Furong Huang. I would also like to thank my co-authors Bayan Bruss from Capital One and Andrew Gordon Wilson from New York University.

Big thanks to Lauren Lederer for all her help navigating the graduate school.

Finally, I am thankful to my family. My PhD would not have been possible without the support of my wife and my co-author Valeriia Cherepanova.

DEDICATION

To my family

Chapter 1

INTRODUCTION

Diseases and healthcare crises present big challenges for humanity. The global COVID-19 pandemic took countless lives, caused deepest economic crises, strained even the world's most developed healthcare systems, and changed the life of almost every person on the planet. Other diseases also pose significant problems. According to the Centers for Disease Control and Prevention [224], the number of new cancer diagnoses will reach 2,286,300 in 2050, almost 1.5 times more than were diagnosed in 2015, and the cancer incidence is still growing. These challenges require constant scientific progress in many fields including applied mathematics, machine learning and data science.

As the field of healthcare becomes increasingly data-driven, optimization and machine learning methods provide the scientific community and practitioners with powerful tools to extract insights from the data and use them to improve clinical practice. However, medical applications have a set of unique challenges and opportunities. Medical data is often limited since accumulating large amounts of patient data with labels is difficult, especially for rare conditions or hospital-specific tasks. Additionally, the high stakes healthcare environment requires development of reliable explainability methods to ensure the safe application of machine learning models. On the other hand, abundant unlabelled data on the population level provides unique opportunities to guide policy efforts in public health.

In this work we leverage these opportunities and develop methods to address the above challenges.

We first consider applications in radiation oncology. We propose a novel optimization framework for multi-modality radiation therapy and propose optimization methods to solve the underlying non-convex optimization problems.

After that, we discuss our work on leveraging unlabelled cell phone mobility data and unsupervised manifold learning to gain insights into population behavior during the COVID-19 pandemic. We show that these insights can help guide policy efforts and non-pharmaceutical interventions during COVID-19.

We then proceed to the methodological part of our work. With the growing use of deep learning black-box models in medicine and other high-stakes applications, it is imperative to develop methods to understand the decisions of neural models. To address this, we propose a novel parameter-saliency explainability method for deep neural networks which can be used to analyze model mistakes.

After that, we present our work on transfer learning with deep tabular models where in a realistic medical setting we show that representation learning with neural networks provides a definitive advantage over the traditionally dominant gradient boosted decision tree tabular methods when downstream data is limited as is often the case in medical applications. We also develop a method to address the tabular-specific problems arising in the settings of transfer learning with tabular data.

Finally, we present an application of deep tabular models to improve patient-specific quality assurance (PSQA) process in radiation oncology which a standard step in the current clinical practice.

1.1 Organization

- Chapter 2 is devoted to the radiation oncology applications. It introduces our optimization framework for multi-modality treatment planning in radiation therapy of cancer and optimization methods to solve the underlying non-convex problems.

- Chapter 3 focuses on the unsupervised data-driven analysis of population mobility behavior during the beginning of the COVID-19 pandemic. We leverage manifold learning and clustering to extract insights from the population-level cell phone mobility data and explain how those insights can aid public health policymakers.

- Chapter 4 presents our work on explainability of deep learning models. We develop a method to compute saliency of neural network parameters and leverage it to explore the interplay between neural network parameters, inputs and mistakes.

- Chapter 5 explores transfer learning with deep tabular models. We conduct experiments in a realistic medical test bed and show that representation learning with recently developed tabular neural networks is more powerful than traditionally leading gradient boosted tree approaches. We further compare the supervised and self-supervised pretraining strategies and provide practical advice on transfer learning with tabular models. Finally, we propose a pseudo-feature method for cases where the upstream and downstream feature sets differ, a tabular-specific problem widespread in real-world applications.

- Chapter 6 leverages the leading deep tabular models to predict the patient-specific quality assurance failures of treatment plans in radiation therapy. The models we develop have a direct potential to improve the current clinical practice by reducing the need for patient rescheduling and treatment delays in cases of plan failures and reducing the load on hospital resources.

Chapter 2

MULTI-MODALITY RADIOTHERAPY OPTIMIZATION

This chapter is based on joint work with Minsun Kim and Aleksandr Aravkin [115].

2.1 Introduction

According to the Centers for Disease Control and Prevention [224], the number of new cancer diagnoses will reach 2,286,300 in 2050, almost 1.5 times more than were diagnosed in 2015 and the cancer incidence is still growing. More than half of all cancer patients go through radiotherapy in the course of their cancer treatment. Radiotherapy utilizes ionizing radiation to kill cancer cells and is used as the primary treatment modality for certain cases or as a secondary modality before or after other treatment modalities such as surgery and chemotherapy. Although radiation kills cancer cells, it also damages normal tissue that is on its path. Therefore, the goal of radiotherapy is to maximize the differential in the damages between the tumor and normal tissue.

External beam radiation therapy (EBRT) is a non-invasive type of radiotherapy, where the radiation generated by linear accelerators or cyclotrons is targeted at the tumor from outside the patient's body. The left panel of Figure 2.1 shows a linear accelerator that produces X-rays and electrons to treat the patient lying on the table. There are currently multiple radiation types used in EBRT. The most widely used radiation type is photons (X-rays) and the therapeutic effect of photons on the tumor and normal tissue is well-established due to their long history of use. However, there are emerging interests in the EBRT using heavy charged particles such as protons and carbon ions due to their unique dosimetric and biological characteristics. For example, the right panel of Figure 2.1 shows the percent depth dose (PDD) of photons and protons. Unlike photons, which deposit the maximum dose near

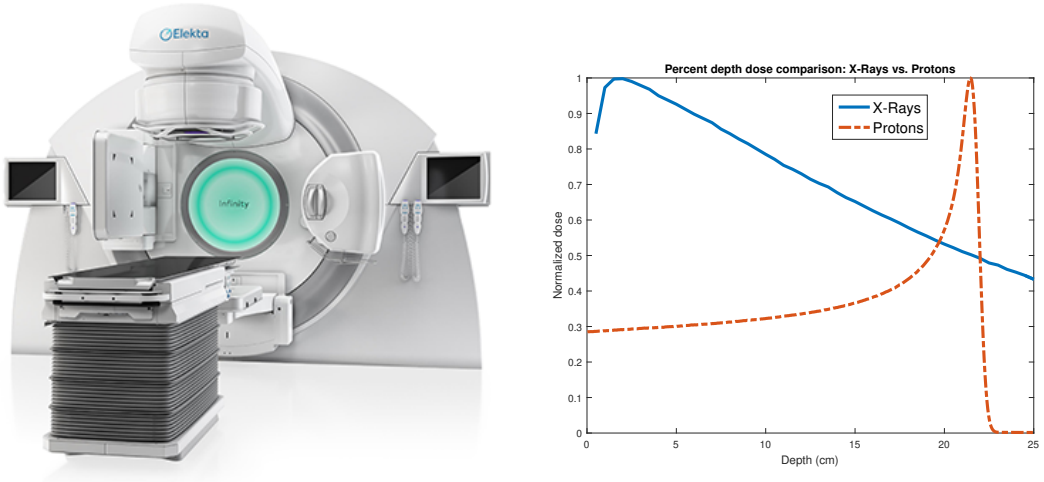


Figure 2.1: **Introduction to radiation therapy.** Left: Elekta medical linear accelerator[1]. Right: Percent depth dose comparison between X-rays and protons.

the patient's surface, typically within 3 cm from the beam entrance for photons with the energy of less than 18 MV used in clinical practice, protons deposit the maximum dose at the end of their range. Since the range of protons is dependent on their energy, we can theoretically aim to deposit a large amount of dose in the tumor, leaving the normal tissue behind it almost no dose by adjusting the energy of the particles. This superior dosimetric effect of heavy charged particles compared to conventional photons can be easily seen in the isodose distributions in Figure 2.2 for a patient with craniospinal irradiation. The patient treated with protons (Figure 2.2, top panel) receives much less dose to the normal tissue outside the target area (whole brain and spinal cord) compared to the patient treated with photons (Figure 2.2, bottom panel).

On the other hand, certain radiation types such as neutrons or carbon ions have superior biological effects compared to photons as quantified by relative biological effectiveness. Relative biological effectiveness (RBE) is defined as the ratio of the absorbed dose of a reference radiation type (often photons) and another radiation type, which kills the same number of

[1]image source: `www.elekta.com`

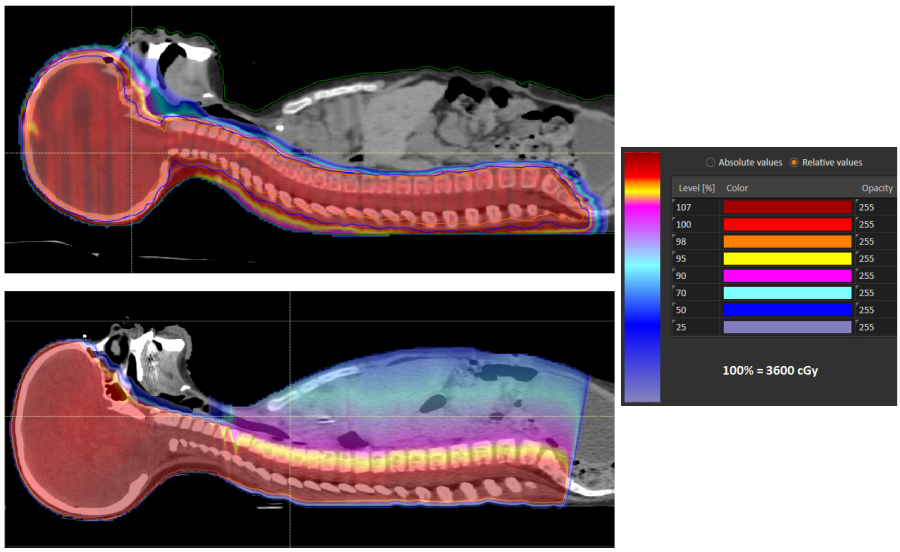


Figure 2.2: **Craniospinal irradiation patient plans.** Top: Isodose distributions with a single posterior-anterior proton beam. Bottom: Isodose distributions with a single posterior-anterior photon beam. Both plans are normalized to 3600 cGy at 100 %.

cells. For example, the RBE of neutrons is defined as $D_{\mathrm{X}}/D_{\mathrm{n}}$, where D_{X} and D_{n} are the dose that kills the same number of cells using photons and neutrons respectively. Therefore, a radiation type with a higher RBE kills more cells than photons given the same physical dose (absorbed energy per unit mass). By definition, RBE is dependent on the radiation type, tissue type (what type of cells is considered in computing RBE), and the environment conditions such as the level of oxygen [77]. For example, fast neutrons used for radiotherapy have an RBE of 2-5 [22, 217] and it is particularly high in a hypoxic condition such as in the tumor. It is noteworthy that radiation with a higher RBE implies that it damages both tumor and normal tissue more than conventional photons but the differential is not uniformly scaled throughout various tissue types. Therefore, there is an opportunity to exploit the differential to maximize the therapeutic effect. The summary of radiation types currently used in practice worldwide and their dosimetric and biological effect relative to conventional photons is presented in Table 2.1.

Due to the unique dosimetric and biological effect of each radiation type, and practical

Table 2.1: **Comparison of various radiation types.** All are relative to photon external beam radiotherapy, which is currently the most widely used radiation type in practice.

Radiation Type	Dosimetric effect	Biological damage	Cost [156]
Protons	Superior	Similar	3-4 times more
Neutrons	Similar	Superior	Similar
Carbon ions	Superior	Superior	5-6 times more

considerations, there is no single modality that is superior to others in all aspects. Furthermore, it is not obvious which radiation type is optimal for a specific patient. Current efforts to determine an optimal radiation type or combination of radiation types are mostly empirical and largely depend on clinical intuitions [249, 76, 34, 214]. There were some recent efforts to systematically optimize proton and photon treatments combined. Nourollahi, Ghate, and Kim studied a simplified scenario, where the dose from two different radiation types was expressed in a scalar form, and proposed a mathematical framework to find an optimal fractionation for each radiation modality [146], as well as its robust counter part in [147]. Unkelbach et al. minimized the mean biologically effective dose (BED) of organs-at-risk while prescribing a fixed BED to the tumor with fixed fractionation for both protons and photons [211]. Gao et al. studied the hybrid proton-photon inverse planning optimization in [67], where they also optimized the dose distribution from both modalities using a fixed prescription dose to the tumor and fixed fractionation schedules. Eikelder et al. studied the fluence map and fractionation schedule optimization of proton and photon combined using a sparing factor [52]. The sparing factor idea in the fractionation schedule optimization is explained in detail in [168] with a single modality case. In summary, the relative dose of OAR in each voxel is fixed as a fraction of the tumor dose through a sparing factor. Eikelder et al. took a heuristic approach to proton-photon modality fractionation optimization problem, where the maximum feasible BED to the tumor was computed for all possible combinations of fractions for each modality to find an optimal fractionation schedule. In particular, their approach separates the fractionation schedule optimization from the fluence

map optimization because the sparing factors are not optimization variables.

The purpose of this chapter is to:

1. Set up a full-scale rigorous mathematical framework to simultaneously optimize fluence maps and fractionation schedules for two or more radiation modalities. In other words, we seek the solutions to the problem, which exploit the full flexibility of treatment parameters.

2. Develop an efficient optimization algorithm to solve the underlying non-convex constrained optimization problem to find optimal fractionation and corresponding optimal fluence maps for each modality.

3. Test the feasibility and clinical relevance of the proposed framework on a small-scale phantom, where clinical intuition can be used to validate the results of the numerical simulation.

2.2 Multi-Modality Treatment Planning Optimization Model

We start by developing an optimization formulation for the radiation treatment planning problem using M radiation modalities. Consider optimizing the dose distributions using M modalities, where each modality m delivers N_m fractions ($m = 1, 2, \cdots, M$). Biological effect (BE) based on the linear-quadratic (LQ) cell-survival model is widely used in radiation oncology to characterize the effect of the physical dose (the energy absorbed per unit mass) combined with the fractionation effect [77]. We use BE to compare the effect of two different fractionation schedules on cell-killing since the same total (physical) dose could lead to a different biological outcome depending on the fractionation schedule.

BE of delivering dose d_m to the tumor in N_m fractions for a single radiation modality m is given by

$$\mathrm{BE}_m = N_m \alpha_m^T d_m + N_m \beta_m^T d_m^2 - \tau(N_m), \tag{2.1}$$

where α_m and β_m are radiation modality and tissue-specific radiobiological parameters in the LQ model and $\tau(N_m)$ is the tumor proliferation term which only depends on the total duration of the treatment. The dose distribution d_m is the image of fluence map u_m under the dose mapping. We use a linear dose mapping for the dose calculation, so that $d_m = A_m u_m$, where A_m is the dose deposition matrix of modality m.

Using BE in the multi-modality setting with M radiation modalities and N_m fractions for each modality, the problem could be formulated as follows:

Maximize the total tumor BE (the sum of BEs across all modalities and across all tumor voxels) while keeping each organ-at-risk (OAR) total BEs under tolerance.

OAR constraint types considered in this chapter are mean dose constraints and maximum dose constraints, which are common in practice for parallel[2] and serial[3] normal tissue types respectively. Another common constraint type for OAR in practice is dose-volume (DV) constraints, which specify a critical volume of OAR that must receive less than a certain critical dose value. DV constraints can be handled using the constraint generation method, where the initial optimization is done without DV constraints and then the maximum constraints are applied to specific voxels in the subsequent optimization if DV constraints are violated in the first optimization [100, 168, 169].

2.2.1 Notation

We use the following notation to describe the proposed model:

- M: total number of radiation modalities
- u_m: fluence map (beamlet intensities) for modality m
- N_m: number of fractions for modality m

[2]The organ remains functional when part of it is damaged by radiation.

[3]If any part of an organ is damaged, the organ becomes no longer functional.

- T_m: tumor dose deposition matrix for modality m such that $T_m u_m$ gives the dose distribution delivered to the tumor using a fluence map u_m
- $T_m(j)$: j-th row of matrix T_m
- H_m^i: i-th OAR dose coefficient matrix for modality m
- $H_m^i(j)$: j-th row of matrix H_m^i
- n: number of OARs
- l: number of voxels in the tumor
- J_0: voxel index set for the tumor (so $|J_0| = l$)
- J_i: voxel index set for the i-th OAR, $i = 1, 2, \cdots, n$
- I_{mean}: index set of OARs with mean-dose constraint
- I_{max}: index set of OARs with max-dose constraint
- α_m, β_m: vectors of the linear and quadratic radiobiological coefficients in the tumor BE for modality $m; \alpha_m, \beta_m \in \mathbb{R}_{++}^{|J_0|}$
- γ_m^i, δ_m^i: vectors of the linear and quadratic radiobiological coefficients in the i-th OAR BE for modality $m; \gamma_m, \delta_m \in \mathbb{R}_{++}^{|J_i|}$
- $\alpha_m(j)$, $\beta_m(j)$, $\gamma_m^i(j)$, $\delta_m^i(j)$: j-th element of the corresponding vector
- C_{mean}^i: tolerance BE for the i-th OAR with a mean dose constraint
- C_{max}^i: tolerance BE for the i-th OAR with a maximum dose constraint

2.2.2 Optimization Framework

We now formulate the fully general multi-modality radiotherapy framework: M radiation modalities with N_m fractions for the m-th modality. We denote tumor dose coefficient matrices which map beamlet intensities u_m to the dose distribution for each modality by T_m, and OAR dose coefficient matrices that map u_m to the dose distribution delivered to the i-th OAR by H_m^i. Our objective is to maximize the total tumor BE, subject to the mean and max constraints on the OAR BEs:

$$\text{(P0)} \qquad \max_{\{u_m, N_m\}_{m=1}^M} \quad \sum_{m=1}^{M} \sum_{j \in J_0} N_m \alpha_m(j)(T_m(j)u_m) + N_m \beta_m(j)(T_m(j)u_m)^2 - \tau(N_m) \tag{P0.1}$$

s.t.

$$\sum_{m=1}^{M} \sum_{j \in J_i} N_m \gamma_m^i(j)(H_m^i(j)u_m) + N_m \delta_m^i(j)(H_m^i(j)u_m)^2 \leq C_{\text{mean}}^i, \quad \forall i \in I_{\text{mean}}, \tag{P0.2}$$

$$\max_{j \in J_i} \left\{ \sum_{m=1}^{M} N_m \gamma_m^i(j)(H_m^i(j)u_m) + N_m \delta_m^i(j)(H_m^i(j)u_m)^2 \right\} \leq C_{\text{max}}^i, \quad \forall i \in I_{\text{max}}, \tag{P0.3}$$

$$1 \leq \sum_{m=1}^{M} N_m \leq N_{\text{max}}, \tag{P0.4}$$

$$N_m \in \mathbb{Z}_{\geq 0}, \qquad m = 1, \ldots, M, \tag{P0.5}$$

$$u_m \succeq 0, \qquad m = 1, \ldots, M \tag{P0.6}$$

This problem is a non-convex mixed integer program. We tackle this problem by devising

a bilevel optimization algorithm where in the upper level we relax integrality constraints and optimize the fractionation schedule over $\{N_m\}_{m=1}^M$ using the optimal fluence map $\{u_m\}_{m=1}^M$ obtained from the lower level for a given fractionation schedule. The details of the algorithms are presented in the following section.

2.3 Optimization Algorithm

In this section we develop a bilevel optimization algorithm to solve Problem (P0). The upper level optimizing over fractionation schedules ($\{N_m\}_{m=1}^M$) is detailed in Section 2.3.1 and the lower level approach (used as a subroutine in the upper level) to compute the optimal fluence map ($\{u_m^*\}_{m=1}^M$) for fixed $\{N_m\}_{m=1}^M$ is presented in Section 2.3.2.

2.3.1 Upper Level: Fractionation Schedule Optimization

In the upper level, we optimize the number of fractions of each modality. We first convert the maximization problem in Problem (P0) to a minimization problem. The tumor proliferation term in Equations (2.1) and (P0.1) depends on the total treatment duration, $\sum_{m=1}^M N_m$. Assuming that there is no tumor lagging time[4], the tumor proliferation term can be defined as

$$\tau(N) = \frac{\ln 2 \left(\sum_{m=1}^M N_m - 1\right) l}{T_d}, \tag{2.3}$$

where T_d is the tumor doubling time[5] [77]. Using (2.3), we can rewrite (P0) as follows:

---

[4]time it takes for the tumor to start proliferation after the treatment starts

[5]time it takes for the tumor to double in the number of cells

$$\text{(P1)} \quad \min_{\{u_m, N_m\}_{m=1}^M} \quad \sum_{m=1}^{M} N_m \left\{ -{\alpha_m}^T T_i u_m - (T_m u_m)^T \mathrm{diag}(\beta_m)(T_m u_m) \right\} + \tau(N) \tag{P1.1}$$

s.t.

$$\sum_{m=1}^{M} N_m \left\{ {\gamma_m^i}^T H_m^i u_m + (H_m^i u_m)^T \mathrm{diag}(\delta_m^i)(H_m^i u_m) \right\} \leq C_{\text{mean}}^i, \quad \forall i \in I_{\text{mean}}, \tag{P1.2}$$

$$\max_{j \in J_i} \left\{ \sum_{m=1}^{M} N_m \gamma_m^i(j)(H_m^i(j) u_m) + N_m \delta_m^i(j)(H_m^i(j) u_m)^2 \right\} \leq C_{\text{max}}^i, \quad \forall i \in I_{\text{max}}, \tag{P1.3}$$

$$1 \leq \sum_{m=1}^{M} N_m \leq N_{\text{max}}, \tag{P1.4}$$

$$N_m \in \mathbb{Z}_{\geq 0}, \qquad m = 1, \ldots, M, \tag{P1.5}$$

$$u_m \succeq 0, \qquad m = 1, \ldots, M \tag{P1.6}$$

where l is the total number of voxels in the tumor. Let $F(\{u_m\}_{m=1}^M, \{N_m\}_{m=1}^M)$ denote the objective function in (P1.1). To solve the problem, we relax integrality constraints and solve the continuous optimization problem.

$$\text{(P2)} \quad \min_{N_1, \cdots, N_M} \quad V(N_1, \cdots, N_M) \tag{P2.1}$$

s.t.

$$1 \leq \sum_{m=1}^{M} N_m \leq N_{\text{max}}, \tag{P2.2}$$

$$N_m \geq 0, \qquad m = 1, \ldots, M, \tag{P2.3}$$

where $V(N_1, \ldots, N_M)$ is the *value function* of $\{N_m\}_{m=1}^M$ defined by

$$V(N_1, \ldots, N_M) := F(\{u_m^*(N_1, \ldots, N_M)\}_{m=1}^M, N_1, \ldots, N_M) \tag{2.6}$$

where each $u_m^*(N_1, \ldots, N_m)$ is the optimal fluence map solution for fixed $(N_1, \ldots, N_m)$:

$$\{u_m^*(N_1, \ldots, N_M)\}_{m=1}^M \quad = \quad \arg\min_u(\text{P1}(N_1, \ldots, N_M)). \tag{2.7}$$

Every evaluation of the value function V requires solving an optimization problem in the fluence maps (Equation (2.7)), and this is done using the lower level soluton approach discussed in Section 2.3.2. Problem (P2) has a nonlinear nonconvex objective with simple linear inequality constraints, and we solve it using a trust region method [39] for constrained optimization as implemented in Python package, SciPy [213]. The solution of (P2) is rounded to the nearest integers, N_f^* in a post-processing step. Once we have an integral N_f^*, we also update fluence maps $u_f^* = \arg\min_u(\text{P1}(N_f^*))$ to ensure that the optimality and feasibility are enforced for the integer solutions. The upper level solution approach is described in Algorithm 1. Since the problem is nonconvex, we repeat Algorithm 1 for multiple initial guesses of $\{N_m^{(0)}\}$ and the best solution is chosen as the final optimal solution.

Algorithm 1 $\{N_m\}$ Fractionation Schedule Optimization

Input: $u^{(0)}, N_1^{(0)}, \ldots, N_M^{(0)}$
function ObjectiveFun($u, N1, \ldots, N_M$)
    return $\sum_{i=1}^M N_i(\alpha_i{}^T T_i u_i - (T_i u_i)^T \text{diag}(\beta_i)(T_i u_i)) + l\left(\sum_{j=1}^M N_j - 1\right)\ln 2/T_d$.
function ValueFun($N_1, \ldots, N_M$) ▷ Define the value function to optimize
    $u_N^* \leftarrow$LowerLevelSolver($u^{(0)}, N_1, \ldots, N_M$)
    return ObjectiveFun($u_N^*, N1, \ldots, N_M$)
$N_1^*, \ldots, N_M^* \leftarrow$ TrustRegionConstr(ValueFun, $N_1^{(0)}, \ldots, N_M^{(0)}$, $\sum_{j=1}^M N_j \leq 25$, $\{N_j \geq 0\}_1^M$)
$u^* \leftarrow$ LowerLevelSolver($u^{(0)}, [N_1^*], \ldots, [N_M^*]$)
Output: $u^*, N_1^*, \ldots, N_M^*$

2.3.2 Lower Level: Fluence Map Optimization for Fixed Fractionation

In this section, we describe the lower level solution required to compute (2.7) for a given fractionation schedule. The tumor proliferation term is independent of u_m and does not affect the problem for fixed N_m. Next, every maximum dose constraint in (P1.3) can be viewed as a mean dose constraint applied to every single voxel of a given OAR, essentially treating each of those voxels as a mean-dose OAR in its own right. The dose mapping matrices for those "new OARs" are comprised of the corresponding rows of the original OAR dose mapping matrices. Let us specify the dose mapping and BE coefficient matrices in terms of these new OARs. First, noting that all the voxel index sets $J_1, \ldots, J_n$ are disjoint, define the index set of all OAR voxels with maximum dose constraints as follows:

$$\tilde{I}_{\max} = \{i = (j,k) | k \in I_{\max}, j \in J_k\}. \tag{2.8}$$

$\tilde{I}_{\max}$ is the index set of our new "mean-dose OARs". Now, for every OAR $i = (j,k) \in \tilde{I}_{\max}$, define the corresponding dose coefficient matrix for modality m as $H_m^k(j)$ (in fact, this would be a row-vector) and arrange M dose coefficient matrices into a single block-diagonal generalized dose matrix H^i for the OAR i. Stack the corresponding linear BE coefficients into vectors and include the fixed N_m to obtain the following generalized linear BE coefficients:

$$H^i = \begin{bmatrix} H_1^k(j) & & & \\ & H_2^k(j) & & \\ & & \ddots & \\ & & & H_M^k(j) \end{bmatrix}, \tilde{\gamma}^i = \begin{bmatrix} N_1\gamma_1^k(j) \\ N_2\gamma_2^k(j) \\ \vdots \\ N_M\gamma_M^k(j) \end{bmatrix}. \tag{2.9}$$

Similarly, for every OAR $i \in I_{\text{mean}}$ arrange the dose matrices of each modality into block-diagonal generalized dose matrices and stack the linear BE coefficients into large generalized linear BE coefficient vectors, do the same for the tumor dose matrices and BE coefficients

getting:

$$H^i = \begin{bmatrix} H_1^k & & & \\ & H_2^k & & \\ & & \ddots & \\ & & & H_M^k \end{bmatrix}, \tilde{\gamma}^i = \begin{bmatrix} N_1\gamma_1^k \\ N_2\gamma_2^k \\ \vdots \\ N_M\gamma_M^k \end{bmatrix}, T = \begin{bmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_M \end{bmatrix}, \tilde{\alpha} = \begin{bmatrix} -N_1\alpha_1 \\ -N_2\alpha_2 \\ \vdots \\ -N_M\alpha_M \end{bmatrix}. \tag{2.10}$$

Rearranging the sums in equations (P0.1) and (P0.2) of the Problem (P0) and rewriting the max-dose constraints (P0.3) in terms of the new one-voxel mean-dose-constrained OARs, we reformulate the Problem (P0) for fixed $\{N_m\}_{m=1}^M$ as follows:

$$\begin{aligned} \min_u \quad & \tilde{\alpha}^T(Tu) - (Tu)^T B(Tu) \\ \text{s.t.} \quad & \\ & \tilde{\gamma^i}^T H^i u + (H^i u)^T D^i (H^i u) \leq C^i_{\text{mean}}, \quad \forall i \in I_{\text{mean}}, \\ & \tilde{\gamma^i}^T H^i u + (H^i u)^T D^i (H^i u) \leq C^i_{\text{max}}, \quad \forall i \in \tilde{I}_{\text{max}}, \\ & u \succeq 0. \end{aligned}$$

where

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{bmatrix}, B = \begin{bmatrix} N_1\text{diag}(\beta_1) & & & \\ & N_2\text{diag}(\beta_2) & & \\ & & \ddots & \\ & & & N_M\text{diag}(\beta_M) \end{bmatrix},$$

and

$$D^i = \begin{bmatrix} N_1\text{diag}(\delta_1^i) & & & \\ & N_2\text{diag}(\delta_2^i) & & \\ & & \ddots & \\ & & & N_M\text{diag}(\delta_M^i) \end{bmatrix} \quad \forall i \in I_{\text{mean}},$$

$$D^i = \begin{bmatrix} N_1\delta_1^k(j) & & & \\ & N_2\delta_2^k(j) & & \\ & & \ddots & \\ & & & N_M\delta_M^k(j) \end{bmatrix} \qquad \forall i = (j,k) \in \tilde{I}_{\max}.$$

Here, $\mathrm{diag}(v)$ denotes the diagonal matrix formed from a vector v, and the matrix B is block-diagonal with blocks $N_m\mathrm{diag}(\beta_m)$. Likewise, the matrices D^i for $i \in I_{\text{mean}}$ are block-diagonal with blocks $N_m\mathrm{diag}(\delta_m^i)$. Finally, D^i for $i \in \tilde{I}_{\max}$ corresponding to the one-voxel mean-dose-constrained OARs are diagonal $M \times M$ matrices. For notation brevity, we denote the quadratic form with the matrix B as

$$f(x) := x^T B x.$$

We now restate the optimization formulation for the fixed $\{N_m\}_{m=1}^M$:

$$\begin{aligned}
(\text{P3}) \min_u \quad & \tilde{\alpha}^T(Tu) - f(Tu) && \text{(P3.1)}\\
\text{s.t.} \quad & \\
& \tilde{\gamma^i}^T H^i u + (H^i u)^T D^i (H^i u) \le C^i_{\text{mean}}, \quad \forall i \in I_{\text{mean}}, && \text{(P3.2)}\\
& \tilde{\gamma^i}^T H^i u + (H^i u)^T D^i (H^i u) \le C^i_{\max}, \quad \forall i \in \tilde{I}_{\max}, && \text{(P3.3)}\\
& u \succeq 0. && \text{(P3.4)}
\end{aligned}$$

Non-Convex Relaxation

(P3) is a non-convex constrained optimization problem since we are minimizing a concave objective over a convex set. To approach this problem, we follow the ideas of [245] and introduce auxiliary variables, w_0 and w_i with $i = 1, 2, \cdots, \tilde{n}$, which gives us a more tractable

relaxed problem:

$$\text{(P4)} \quad \min_{u,\, w_0,\, \{w_i\}_{i=1}^{\tilde{n}}} \quad \tilde{\alpha}^T w_0 - f(w_0) + \frac{1}{2\eta_0}\|w_0 - Tu\|^2 + \sum_{i=1}^{\tilde{n}} \frac{1}{2\eta_i}\|w_i - H^i u\|^2 \tag{P4.1}$$

$$\text{s.t.}$$

$$\tilde{\gamma}^T w_i + w_i^T D^i w_i \leq C^i_{\text{mean}}, \quad \forall i \in I_{\text{mean}}, \tag{P4.2}$$

$$\tilde{\gamma}^T w_i + w_i^T D^i w_i \leq C^i_{\text{max}}, \quad \forall i \in \tilde{I}_{\text{max}}, \tag{P4.3}$$

$$u \succeq 0. \tag{P4.4}$$

where $\tilde{n} = |I_{\text{mean}}| + |\tilde{I}_{\text{max}}|$. The norm penalties force w_0 and w_i to be close to Tu and $H^i u$. The parameters η_0 and η_i control the degree of closeness, and, as $\eta_0, \eta_1, \ldots, \eta_{\tilde{n}}$ go to zero, we recover (P3) from (P4). We develop an automatic approach to select these parameters in Section 2.3.2. By design, the auxiliary variables w_i always meet the original mean or maximum BE constraints for every OAR.

To solve (P4), we use block-coordinate descent and iteratively update u, w_0, and w_is. We now describe each update in detail.

Update w_0, w_i

For fixed u, the problem we solve is given by

$$\text{(P5)} \quad \min_{w_0,\, \{w_i\}_{i=1}^{\tilde{n}}} \quad \tilde{\alpha}^T w_0 - f(w_0) + \frac{1}{2\eta_0}\|w_0 - Tu\|^2 + \sum_{i=1}^{\tilde{n}} \frac{1}{2\eta_i}\|w_i - H^i u\|^2 \tag{P5.1}$$

$$\text{s.t.}$$

$$\tilde{\gamma}^T w_i + w_i^T D^i w_i \leq C^i_{\text{mean}}, \quad \forall i \in I_{\text{mean}}, \tag{P5.2}$$

$$\tilde{\gamma}^T w_i + w_i^T D^i w_i \leq C^i_{\text{max}}, \quad \forall i \in \tilde{I}_{\text{max}} \tag{P5.3}$$

This problem is decoupled in w_0 and w_is and is therefore equivalent to solving for w_0 and w_i separately. For w_0, dropping the constant terms, we have

$$\min_{w_0} \alpha^T w_0 - f(w_0) + \frac{1}{2\eta_0}\|w_0 - Tu\|^2, \tag{2.15}$$

which is equivalent to

$$\min_{w_0} -f(w_0) + \frac{1}{2\eta_0}\|w_0 - (Tu - \eta_0\tilde{\alpha})\|^2. \tag{2.16}$$

The solution of this minimization step for w_0 can be written compactly as

$$w_0^+ = \text{prox}_{-\eta_0 f}(Tu - \eta_0\tilde{\alpha}), \tag{2.17}$$

where the *proximal operator*, or prox, is defined by

$$\text{prox}_{-\eta_0 f}(y) = \arg\min_x \left\{ -f(x) + \frac{1}{2\eta_0}\|x - y\|^2 \right\}. \tag{2.18}$$

Equation (2.17) gives us the update w_0^+ for w_0. The proximal operator always exists for closed convex functions, but care must be taken in the nonconvex case, and in particular for the concave $-f(x)$. By analyzing the problem, we find the range of values η_0 for which the proximal operator is well-defined (see Appendix A.1):

$$(\text{prox}_{-\eta_0 f}(y))_j = \begin{cases} \infty, & \max_i(B_{ii} - \frac{1}{2\eta_0}) > 0 \;\&\; j = \arg\max_i(B_{ii} - \frac{1}{2\eta_0}) \\ 0, & \max_i(B_{ii} - \frac{1}{2\eta_0}) > 0 \;\&\; j \neq \arg\max_i(B_{ii} - \frac{1}{2\eta_0}) \\ \frac{1}{\eta_0} y_j / (-2B_{jj} + \frac{1}{\eta_0}), & \max_i(B_{ii} - \frac{1}{2\eta_0}) \leq 0. \end{cases} \tag{2.19}$$

In order for $\text{prox}_{-\eta_0 f}$ to be well-defined, we must have $\max_i(B_{ii} - \frac{1}{2\eta_0}) \leq 0$. This assumption forces a lower limit on the penalty parameter, making sure problems (P3) and (P4) are close, and gives us a starting point for the parameter selection process discussed in Section 2.3.2.

Next, we consider the optimization problem with respect to each w_i keeping only those terms that depend on w_i. We have $\tilde{n}$ number of problems with the same structure:

$$\text{(P6)} \min_{w_i} \quad \|w_i - H^i u\|^2 \tag{P6.1}$$

$$\text{s.t.}$$

$$\tilde{\gamma}^T w_i + w_i^T D^i w_i \leq C^i_{\text{mean/max}}, \quad \forall i \in I_{\text{mean}}/\tilde{I}_{\text{max}} \tag{P6.2}$$

The solution of this problem is the projection of $H^i u$ onto the convex set $\Omega_i = \{w_i : \tilde{\gamma}^T w_i + w_i^T D w_i \leq C^i_{\text{mean/max}}\}$. The solution method to find the projection

$$\text{proj}_{\Omega_i}(v) = \arg\min_{w \in \Omega_i} \|w - v\|^2$$

is presented in Appendix A.2. Projection onto a closed convex set is a special case of the prox operator, and is always well-defined and single-valued.

Update u

We now consider the subproblem for u for fixed w_0 and w_is. Dropping the constant terms, we have:

$$\min_{u \succeq 0} \frac{1}{2\eta_0}\|w_0 - Tu\|^2 + \sum_{i=1}^{\tilde{n}} \frac{1}{2\eta_i}\|w_i - H^i u\|^2 \tag{2.21}$$

This is a non-negative least squares problem, which we solve using the Fast Non-Negative Least Squares algorithm [23]. Algorithm 2 summarizes all updates of this block-coordinate descent algorithm. There is only one block (with respect to w_0) that does not necessarily have a unique minimum, and so Algorithm 2 converges to a stationary point by the results of [207].

Algorithm 2 Fluence Map Optimization with Fixed Parameters and Fractions

Input: $u^{(0)}, \eta_0, \eta_1, \ldots, \eta_{\tilde{n}}, N_1, \ldots, N_M$

function LowerLevelFixedParams($u^{(0)}, \eta_0, \eta_1, \ldots, \eta_{\tilde{n}}, N_1, \ldots, N_M, I_{\text{mean}}, \tilde{I}_{\max}$)

  Initialize: $k = 0$

  $\tilde{\alpha} := \begin{bmatrix} -N_1\alpha_1 \\ -N_2\alpha_2 \\ \vdots \\ -N_M\alpha_M \end{bmatrix}, B := \begin{bmatrix} N_1\text{diag}(\beta_1) & & & \\ & N_2\text{diag}(\beta_2) & & \\ & & \ddots & \\ & & & N_M\text{diag}(\beta_M) \end{bmatrix}$

  for $i = 1, \ldots, \tilde{n}$ **do**

    $\tilde{\gamma}^i := \begin{bmatrix} N_1\gamma_1^i \\ N_2\gamma_2^i \\ \vdots \\ N_M\gamma_M^i \end{bmatrix}, D^i := \begin{bmatrix} N_1\text{diag}(\delta_1^i) & & & \\ & N_2\text{diag}(\delta_2^i) & & \\ & & \ddots & \\ & & & N_M\text{diag}(\delta_M^i) \end{bmatrix}$

  while not converged **do**

    $k \leftarrow k + 1$

    $w_0^{(k)} \leftarrow \text{prox}_{-\eta_0 f}(Tu - \eta_0\tilde{\alpha})$

    for $i \in I_{\text{mean}}$ **do**

      $\Omega_i = \{w_i : (\tilde{\gamma}^i)^T w_i + w_i^T D^i w_i \leq C^i_{\text{mean}}\}$

      $w_i^{(k)} \leftarrow \text{proj}_{\Omega_i}(H^i u)$

    for $i \in \tilde{I}_{\max}$ **do**

      $\Omega_i = \{w_i : (\tilde{\gamma}^i)^T w_i + w_i^T D^i w_i \leq C^i_{\max}\}$

      $w_i^{(k)} \leftarrow \text{proj}_{\Omega_i}(H^i u)$

    $u^{(k)} \leftarrow \arg\min_{u \succeq 0} \frac{1}{2\eta_0}\|w_0^{(k)} - Tu\|^2 + \sum_{i=1}^{\tilde{n}} \frac{1}{2\eta_i}\|w_i^{(k)} - H^i u\|^2$

  $u^* \leftarrow u^{(k)}$

  return u^*

Output: u^*

Automated Parameter Selection

The relaxed problem (P4) uses multiple parameters: η_0 and $\{\eta_i\}_{i=1}^{\tilde{n}}$. The auxiliary variable w_0 corresponds to the tumor and w_i corresponds to the i-th OAR with $\tilde{n}$ number of OARs in the problem (including the introduced one-voxel maximum-dose-constraint OARs). Decreasing η_0 enforces the optimality of the solution (i.e. maximizes the tumor BE) because Tu is forced to be closer to w_0, while decreasing η_i increases the penalty corresponding to the i-th OAR constraint and thus enforces the feasibility of our solution making $H^i u$ closer to w_i. The existence of the proximal operator of the function $(-f)$ imposes an upper bound on η_0:

$$\eta_0 \leq \frac{1}{2\max_i B_{ii}}. \tag{2.22}$$

That is, the optimality penalty $\frac{1}{2\eta_0}$ should be big enough for the proximal operator to exist. This gives us an initial value for the parameter selection procedure. Combining these ideas, we develop an algorithm for the automated selection of the parameters η_i:

1. Start η_0 from the threshold in Equation (2.22). Initialize $\{\eta_i\}_{i=1}^{\tilde{n}}$ with the same value. Find the solution u.

2. Check if any OAR constraints are violated by u found in Step 1.

3. Enforce feasibility: If there are any violated constraints, decrease η_i by setting $\eta_i^+ = \Delta_\eta \eta_i$ with $\Delta_\eta < 1$ and solve for new u^+. Repeat steps 2-3 until all OAR constraints are satisfied.

4. Enforce optimality, that is, decrease η_0 by setting $\eta_0^+ = \Delta_\eta \eta_0$ and resolve for u^+ until u^+ fails to satisfy any constraint within the required tolerance for the OAR constraint.

Algorithm 3 summarizes the lower-level optimization solution algorithms including the automated parameter selection with the fixed fractionation schedule $N_1, \ldots, N_M$.

Algorithm 3 Lower Level Optimization with Automated Parameter Selection (fixed fractions)

Input: $u^{(0)}, N_1, \ldots, N_M$
function LowerLevelSolver($u^{(0)}, N_1, \ldots, N_M$)
    Initialize: $\eta_0 = \eta_1 = \cdots = \frac{1}{2 \max_i B_{ii}}$
    $u \leftarrow$ LowerLevelFixedParams($u^{(0)}, \eta_0, \eta_1, \ldots, \eta_{\tilde{n}}$)
    while there exists a violated constraint **do**
        for i in the violated constraints index set **do**
            $\eta_i \leftarrow \eta_i \cdot \Delta_\eta$ ▷ Decrease η_i, enforce feasibility
            $u \leftarrow$ LowerLevelFixedParams($u^{(0)}, \eta_0, \eta_1, \ldots, \eta_{\tilde{n}}$)
    while there is no violated constraints **do**
        $\eta_0 \leftarrow \eta_0 \cdot \Delta_\eta$ ▷ Decrease η_0, enforce optimality
        $u \leftarrow$ LowerLevelFixedParams($u^{(0)}, \eta_0, \eta_1, \ldots, \eta_{\tilde{n}}$)
    $u^* \leftarrow u$
    return u^*
Output: u^*

2.4 Numerical Simulations

We apply the proposed framework with two different radiation modalities, M_1 and M_2, to the 2D phantom geometry shown in Figure 2.3. Since photons are currently the most widely used radiation modality in practice, we investigate the impact of combining photons (M_1) with a second modality (M_2), which has distinctive dosimetric characteristics as shown in Figure 2.2. In Section 2.4.1, we explain treatment planning and evaluation. Our framework is applicable to an arbitrary combination of different modalities with unique dosimetric and biological characteristics, but to build intuition for the proposal we investigate the impact of the difference between two modalities on the optimal BE in simple stages. In Section 2.4.2, we consider the simple scenario of combining M_1 with M_2 that has a dosimetric difference only, i.e. all radiobiological parameters between M_1 and M_2 are identical but the dose mapping matrices are different, i.e., $T_1 \neq T_2$ and $H_1^i \neq H_2^i$. In Section 2.4.3, we also add a radiobiological difference between M_1 and M_2. We vary the tumor's linear coefficients in the LQ model for M_2 (α_2) and then vary the differential in the damage done by M_2

between the tumor and OARs (r). Finally, in Section 2.4.5, we present the performance of the optimal fractionation algorithm by comparing it to the true solution found using the brute-force technique, where all possible combinations of (N_1, N_2) are individually used to find an optimal solution.

In all studies, we compute the initial guess, u_0 in Algorithm 1, to give a uniform dose of 70 Gy to the tumor without any OAR constraints, which is commonly used in practice for head-and-neck tumors. The codes for our numerical simulations are available upon request.

2.4.1 Phantom Geometry and Treatment Planning

The 2D phantom geometry in Figure 2.3 reflects a head-and-neck tumor surrounded by the spinal cord, right parotid, and left parotid glands. The unspecified tissue 1 cm inside of the external contour represents the skin of the patient. Adding a constraint for the unspecified tissue in the optimization ensures that the dose outside the tumor and OAR does not exceed the tolerance level.

The dose mapping matrices for the first modality (T_1, H_1^i) were computed using the Elekta linear accelerator with 6 MV photons at the University of Washington. Radiation dose using M_1 is assumed to be delivered using seven equally spaced beams, i.e. gantry angles of 0°, 51°, 103°, 153°, 206°, 257°, and 309°. The number of beamlets used is 195. The dose mapping matrices for the second modality (T_2, H_2^i) were computed using the proton beams with 250 MeV at the Seattle Cancer Care Alliance Proton Therapy Center with 40 spot positions within the tumor. The radiation dose using M_2 is assumed to be delivered in one beam (gantry angle of 0°) as is often done in the proton therapy practice. We assume that the maximum number of fractions allowed is 25 fractions (i.e. $N_{\max} = 25$).

Normal tissue tolerance BE and the radiobiological parameters used in computing BE were obtained from published literature using photons and conventional fractionation schedules [136]. The α_1/β_1 ratio is 10 Gy for the tumor with reference modality M_1. The γ_1/δ_1 ratio is 2 Gy for the cord and unspecified tissue, and 5 Gy for the right and left parotid glands with M_1. The linear coefficients α_1 and γ_1 of the LQ model are all assumed to be

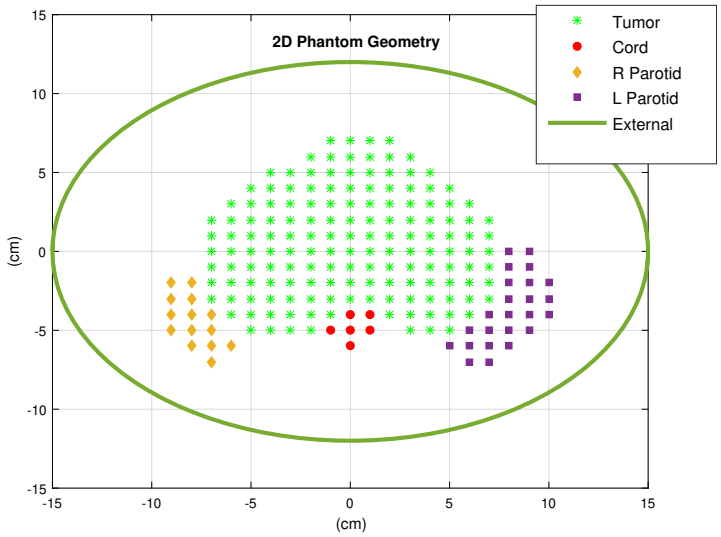


Figure 2.3: **Phantom geometry**

0.35 Gy^{-1} for M_1. The tolerance BE for all OARs used is summarized in Table 2.2.

Table 2.2: **Constraint type and tolerance biological effect for organs-at-risk (OAR) used in treatment planning.**

	Constraint type	δ_1/γ_1	Tolerance BE
Cord	Maximum dose	2 Gy	35
Right Parotid	Mean dose	5 Gy	12
Left Parotid	Mean dose	5 Gy	12
Unspecified tissue	Maximum dose	2 Gy	13.125

We introduce a relative damage factor of $r = \delta_2/\alpha_2$ for M_2 to capture the relative effect of M_2 on the tumor and OAR. When a modality has a larger biological effect on the tumor, it also damages OAR more. However, the magnitude of the damage depends on tissue type, cell cycles, and other conditions. Therefore, r is used to capture the differential in the damage between the tumor and OAR by M_2. As r increases, the damage to OAR is relatively greater than the damage to the tumor, making M_2 clinically undesirable. We used $r \in \{0.5, 1.0, 1.5\}$.

The evaluation of the proposed framework is performed using percentage improvement

in BE compared to BE obtained from two other methods with a single modality: (1) a conventional treatment course and (2) a single modality treatment course with an optimal fractionation schedule.

Denote by T_{conv} the conventional treatment course with fixed 25 fractions using photons only, i.e., $N_1 = 25$ fixed. Denote by T_{single} the treatment course with a single modality with photons and an optimal number of fractions $N_1^{\dagger}$. Then the percentage improvement of the objective function value (= BE) using the optimal double-modality treatment course T^* with (N_1^*, N_2^*), where $1 \leq N_1^* + N_2^* \leq 25$, is given by

- Relative to a conventional course T_{conv} with $N_1 = 25$ fixed

$$\text{pObj}_{\text{conv}} = \frac{\text{BE using } T^*}{\text{BE using } T_{\text{conv}}} \times 100\%.$$

- Relative to a single modality course T_{single} with the optimal N_1^*

$$\text{pObj}_{\text{single}} = \frac{\text{BE using } T^*}{\text{BE using } T_{\text{single}}} \times 100\%.$$

2.4.2 Dosimetric Difference

In this section, we investigate the scenario where the only difference between M_1 and M_2 is the dose mapping matrices for the tumor and OAR. All other radiobiological parameters are set to be the same for the two modalities. The tumor doubling time, T_d, is varied between 2 and 100 days, that is, $T_d \in \{2, 5, 20, 50, 100\}$. The percentage improvement in the optimal BE with T^* ranges between 105.7 % and 107.9 % compared to T_{single}, and between 107.7 % and 122.9 % compared to T_{conv} depending on T_d. The optimal number of fractions increases as T_d increases, which agrees with clinical intuition. Slowly growing tumors (i.e. larger T_d) benefit from a long treatment course to reduce the long-term normal tissue side effect since normal tissues have better capability to recover from radiation damage than a tumor between fractions [77]. The additional improvement seen in $\text{pObj}_{\text{conv}}$ compared to $\text{pObj}_{\text{single}}$

implies that the benefit of using T^* comes from both using optimal fractionation schedule and multiple modalities. However, for a tumor with a large T_d, there is no benefit of using two modalities compared to a single modality with an optimal fractionation schedule. The complete results are shown in Table 2.3.

Table 2.3: **Optimal BE improvement with various tumor doubling time.** Parameters are fixed at $\alpha_2 = 0.35$ Gy$^{-1}, r = 1.0$.

T_d(days)	Dual modality (N_1^*, N_2^*)	Single modality $(N_1^\dagger)$	$\text{pObj}_{\text{single}}(\%)$	$\text{pObj}_{\text{conv}}(\%)$
2	(2, 2)	2	105.7	122.9
5	(6, 6)	6	106.8	110.0
10	(13, 12)	13	107.9	108.2
50	(12, 12)	25	107.8	107.8
100	(12, 12)	25	107.8	107.8

2.4.3 Radiobiological Difference

In this section we present the results when M_2 has a radiobiological difference from M_1 in addition to the dosimetric difference. First, we increase the linear coefficients in the LQ model, which means that M_2 kills more cells than M_1. We used $\alpha_2 \in \{0.35, 0.55, 0.75\}$ with r fixed at 1. The percentage improvement in BE with T^* ranges between 106.6 % and 107.2% compared to the single modality course with optimal fractionation T_{single} and between 109.6 % and 110.2 % compared to the conventional treatment course T_{conv} with 25 fractions. There is no significant difference in the results using different α_2 values, and therefore the benefit of using M_2 is not clear. This is likely because killing more cells applies to both tumor and OAR. The complete results are shown in Table 2.4.

We next varied the relative damage factor r, that is the differential in the damage between the tumor and OAR by M_2. We used $r \in \{0.5, 1.0, 1.5\}$. When $r = 1$, the tumor's linear coefficient in the LQ model is the same as the OAR's linear coefficient, i.e. $\alpha_2 = \delta_2$. A smaller $r < 1$ implies that M_2 damages the tumor more than OAR ($\alpha_2 < \delta_2$) and $r > 1$ implies

Table 2.4: **Optimal BE improvement with various α_2.** Parameters are fixed at $r = 1, T_d = 5$ days.

$\alpha_2(Gy^{-1})$	Dual modality (N_1^*, N_2^*)	Single modality $(N_1^\dagger)$	$\text{pObj}_{\text{single}}(\%)$	$\text{pObj}_{\text{conv}}(\%)$
0.35	(6, 6)	6	106.8	110.0
0.55	(10, 10)	8	106.6	109.6
0.75	(8, 7)	8	107.2	110.2

Table 2.5: **Optimal BE improvement with various r.** Parameters are fixed at $T_d = 5$ days and $\alpha_2 = 0.35/\text{Gy}^{-1}$.

r	Dual modality (N_1^*, N_2^*)	Single modality $(N_1^\dagger)$	$\text{pObj}_{\text{single}}(\%)$	$\text{pObj}_{\text{conv}}(\%)$
0.5	(2, 19)	5	125.4	129.0
1.0	(6, 6)	6	106.8	110.0
1.5	(4, 4)	8	102.2	105.1

the reverse relation. The percentage improvement in BE with T^* ranges between 102.2 % and 125.7% compared to the single modality course with optimal fractionation T_{single} and between 105.1 % and 129.4 % compared to the conventional treatment course T_{conv} with 25 fractions fixed. The results are shown in Table 2.5. As r decreases, the benefit of using a combination of M_1 and M_2 increases since M_2 damages the tumor more than OAR. This agrees with clinical intuition.

2.4.4 Uncertainty of M_2

A modality with superior dosimetric or radiobiological characteristics often comes with a larger degree of uncertainty. For example, protons deposit almost no dose beyond the Bragg peak but the uncertainty in the location of the Bragg peak makes them less desirable since depositing a large dose at a position slightly off target may result in an unacceptable dose to the tumor. Therefore, in practice, one strategy to mitigate uncertainty is to ensure that a larger area around the tumor receives adequate dose by adding an extra margin around

the clinical target volume. Delivering a larger dose in a bigger target volume often increases the dose to OAR, which is disadvantageous. To investigate the effect of this extra margin used for M_2 in our formulation, with hard constraints on OAR BEs, we add an extra margin to the tumor for M_2, compute the average BE per tumor voxel, and use it as an evaluation criteria. The average BE per tumor voxel for M_1 is computed without the extra margin, and therefore we expect M_1 to contribute a larger BE to the total tumor BE compared to the scenario, where the extra margin is not used for M_2. The results of using the average tumor BE per voxel to compute the $\text{pObj}_{\text{single}}$ and $\text{pObj}_{\text{conv}}$ reflect this clinical intuition and the optimal BE improvement of using multi-modality is less than what was achieved in Section 2.4.3. The results are shown in Tables 2.6-2.8.

Table 2.6: **Optimal BE improvement for a range of tumor doubling times when an extra margin to the tumor is used for** M_2. Parameters are fixed at $\alpha_2 = 0.35$ Gy$^{-1}, r = 1.0$

T_d(days)	Dual modality (N_1^*, N_2^*)	Single modality $(N_1^\dagger)$	$\text{pObj}_{\text{single}}(\%)$	$\text{pObj}_{\text{conv}}(\%)$
2	(1, 1)	2	102.5	119.2
5	(3, 4)	3	103.1	105.7
10	(12, 13)	12	102.9	103.2
50	(10, 14)	25	103.1	103.1
100	(11, 14)	25	103.4	103.4

Table 2.7: **Optimal BE improvement with various** α_2 **when an extra margin to the tumor is used for** M_2**.** Parameters are fixed at $r = 1, T_d = 5$ days.

$\alpha_2(Gy^{-1})$	Dual modality (N_1^*, N_2^*)	Single modality $(N_1^\dagger)$	$\text{pObj}_{\text{single}}(\%)$	$\text{pObj}_{\text{conv}}(\%)$
0.35	(3, 4)	3	103.2	105.7
0.55	(6, 8)	6	102.3	105.4
0.75	(4, 5)	4	103.2	106.0

Table 2.8: **Optimal BE improvement with various r when an extra margin to the tumor is used for M_2.** Parameters are fixed at $T_d = 5$ days and $\alpha_2 = 0.35/\text{Gy}^{-1}$.

r	Dual modality (N_1^*, N_2^*)	Single modality $(N_1^\dagger)$	$\text{pObj}_{\text{single}}(\%)$	$\text{pObj}_{\text{conv}}(\%)$
0.8	(7, 11)	6	104.7	107.9
1	(3, 4)	3	103.2	105.7
1.2	(3, 3)	3	102.0	104.7

2.4.5 Optimal Fractionation

In this section, we demonstrate the performance of our approach for the fractionation schedule optimization. We compute the "ground truth" using the brute-force technique, where (P3) is solved for all possible integer (N_1, N_2) combinations with the constraint $1 \leq N_1 + N_2 \leq N_{\max}$ (that is, we sample the value function $V(N_1, N_2)$ for all the points of the integer grid $1 \leq N_1 + N_2 \leq N_{\max}$). Figure 2.4 shows the value function $V(N_1, N_2)$ defined in (2.6) in 3D (left) and the level sets of $V(N_1, N_2)$ (right) obtained from the brute-force technique. The iterates (N_1, N_2) of Algorithm 1 starting from the following four different initial guesses are shown as colored dots on Figure 2.4 (the colors correspond to the different initial points):

1. $(N_1^0, N_2^0) = (1, N_{\max} - 1)$
2. $(N_1^0, N_2^0) = (N_{\max} - 1, 1)$
3. $(N_1^0, N_2^0) = (\lfloor N_{\max}/2 \rfloor, \lceil N_{\max}/2 \rceil)$
4. $(N_1^0, N_2^0) = (1, 1)$

As shown in Figure 2.4, the output of Algorithm 1 is dependent on the initial guess, however, simple initial guesses as implemented in our algorithm may be sufficient to find a clinically relevant optimal solution. In our experiments, the above 4 initial guesses and 8 iterations of the upper level Algorithm 1 were sufficient to identify solutions close to the optimal.

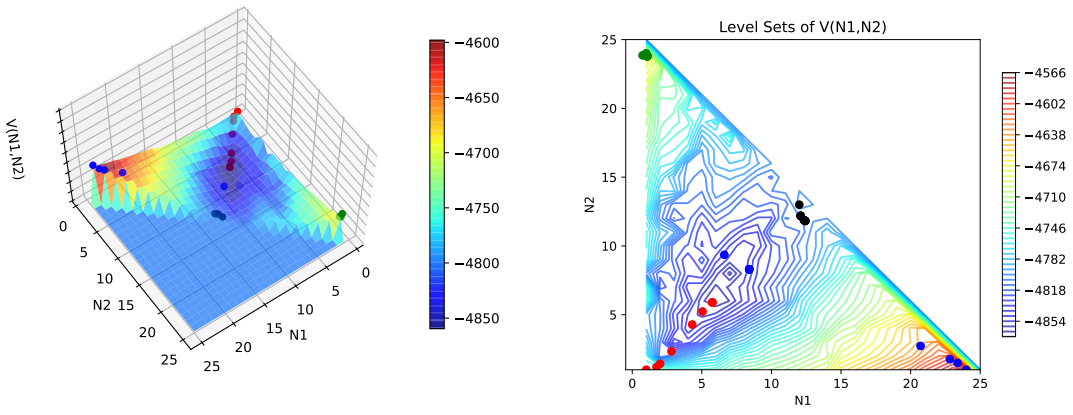


Figure 2.4: **Value function optimization.** Left: The surface represents $V(N_1, N_2)$ computed by brute force. The dots represent iterates starting from four different initial guesses. Right: Level sets of $V(N_1, N_2)$ and the iterates.

2.5 Conclusion

We developed a novel framework for radiation treatment planning with multiple radiation modalities. The principle idea is to maximize the biological effect using the multiple modalities by exploiting their unique dosimetric and biological characteristics captured through the dose mapping matrices and radiobiological parameters in the LQ dose-response model. Our framework allows $N_m = 0$ as long as the sum of fractions of all modalities is equal to or greater than 1 fraction, and therefore it will correctly identify an optimal combination even when a single modality is optimal. The proposed framework ultimately offers an opportunity for an optimal multi-modality treatment planning paradigm. The feasibility of the proposed method was demonstrated using a simple phantom case and two modalities with varying only one parameter at a time such that clinical intuition can be applied. The agreement of the outcome with clinical intuition validates the potential use of our algorithm in more clinically relevant, complicated scenarios, where the clinical intuition is not readily available.

The challenge of this approach is the dependence of optimal solutions on the radiobiological parameters, which is indeed a challenge of biological treatment planning in general.

Active research on advanced imaging could relieve some of the concerns in estimating these parameters [139, 61]. A novel algorithm we proposed to optimize the fractionation schedule with multiple modalities can be easily adapted for a single modality case. A good choice of the initial guess is prudent for the optimal fractionation schedule algorithm to converge to an optimal solution. Applications of our algorithm to actual patient cases to evaluate the clinical significance and learning a good initial guess in the optimal fractionation schedule algorithm from a large patient database are left for future work.

Chapter 3

CELL PHONE MOBILITY DATA AND MANIFOLD LEARNING: INSIGHTS INTO POPULATION BEHAVIOR DURING THE COVID-19 PANDEMIC

This chapter is based on joint work with Dennis L. Chao, Edward Wenger, and Joshua L. Proctor [116, 117].

3.1 Introduction

The ongoing COVID-19 pandemic has had a devastating impact on mortality [230] and economic activity [59] leading to increased food insecurity, poverty, and gender inequity [145]. Most public health interventions attempting to arrest or mitigate the spread of the disease caused by the Severe Acute Respiratory Syndrome Coronavirus 2 are non-pharmaceutical interventions aimed at decreasing transmission by changing people's behavior. For example, every state in the United States (US) issued mandatory or advisory *stay-at-home* orders between March and May of 2020 [143]. However, characterizing changes in behavior during the COVID-19 pandemic, whether due to adherence to stay-at-home orders, loss of employment, or non-pandemic-related factors, is challenging. In this chapter, we use cell-phone mobility data from SafeGraph Inc. to identify the heterogeneous mobility behaviors during COVID-19 in four states and reveal consistent motifs across states, within a state, and even within urban centers. Moreover, the modeling and analyses also point to geographic areas with populations that are young and highly mobile. We believe the approach and insights in the work can be leveraged by local public health officials to better target educational campaigns by geographic area and socioeconomic status.

Cell phone location data is a relatively new but promising way to quantify human move-

ment. The locations of cell phones can be tracked by service providers or applications installed on the phones by users, but data that is shared with scientists is typically anonymized and aggregated to protect the privacy of individuals [102, 72]. Mobility data offers a unique measurement instrument to link public health statements and related legislative actions taken to reduce population mobility with an actual effect on population behavior. Cell-phone mobility data has provided early evidence that these orders were indeed associated with reductions in movement [223, 7, 87, 93, 92]; moreover, adherence was not uniform and may be associated with factors such as socioeconomic status and political leanings [223, 7, 87, 93, 92]. The keen interest in cell-phone mobility data to help inform policy makers during the COVID-19 pandemic has been widely discussed [25], with strong emphasis on the challenges facing data ascertainment bias, interpreting the link between mobility and behavior changes, and the lack of a single mathematical framework for analyzing this data [102, 72]. To date, most investigations of mobility data during the COVID-19 pandemic have compared summary statistics from mobility data, such as average cell-phone mobility within a region, between regions with different demographics. Here, we leverage the mobility data at full geographic and temporal resolution along with recently developed mathematical methods from dynamical systems and machine learning to identify patterns of behavior that are consistent across multiple geographic scales and provide insight into behavioral differences.

Analyzing and interpreting high-dimensional mobility time-series is a challenge. Model and dimensionality reduction has a rich history in the analysis of dynamical systems, with early theoretical work on bifurcation analyses enabling the categorization of qualitatively different dynamic regimes [75] to the more recent data-driven, equation-free approaches [161] enabled by advances in machine-learning and pattern analysis [24]. The standard approach typically involves a linear dimensionality reduction technique, such as the singular value decomposition (SVD) [51], in conjunction with a statistical clustering model to identify similarities across time series [109]. Despite the broad success of this approach, substantial limitations have been identified due to the underlying assumptions associated with the SVD and the mismatch with characteristics of data collected from a complex, temporally evolving

system; this discrepancy has motivated the development of a diverse set of nonlinear dimensionality reduction techniques for time-series data [37]. Methods such as diffusion maps and Laplacian eigenmaps, popular in statistical and computational analyses [38, 19] and contained in a class of machine-learning methodologies called *manifold learning*, have been utilized by the dynamical systems community to identify nonlinear embeddings of the dynamics directly from observational data from the system [37]. Success has been demonstrated with these methods using data generated from simulation models [232]. Here, we leverage these methodologies to identify a lower-dimensional embedding of the mobility time-series data providing a framework that highlights common mobility behaviors at the census-block-group scale, identifies the geographic connectivity of behavior at different spatial scales, and reveals insights into epidemiologically relevant subpopulations.

The outline of this chapter is as follows: the following section of the chapter provides a description of the SafeGraph mobility time-series and US census data (§3.2) along with the mathematical methods used for analysis §3.3. §3.4 highlights the heterogeneity and consistency of mobility patterns at the census block group level during the COVID-19 pandemic. In addition, we show these behaviors are correlated with socioeconomic indicators such as income and home ownership. We also describe a highly mobile population with distinctly different behavior revealed from the analysis. The final section §3.5 offers a discussion on how these findings provide insight into the connection between mobility, behavior, and transmission and how local public health officials can use this data to target information and education campaigns.

3.2 Data

3.2.1 SafeGraph mobility data

We obtained mobility data from SafeGraph, Inc. SafeGraph aggregates mobile device GPS data from various sources and produces anonymized datasets aggregated at the census block group (CBG) level. These data can be obtained free-of-charge for non-commercial use by

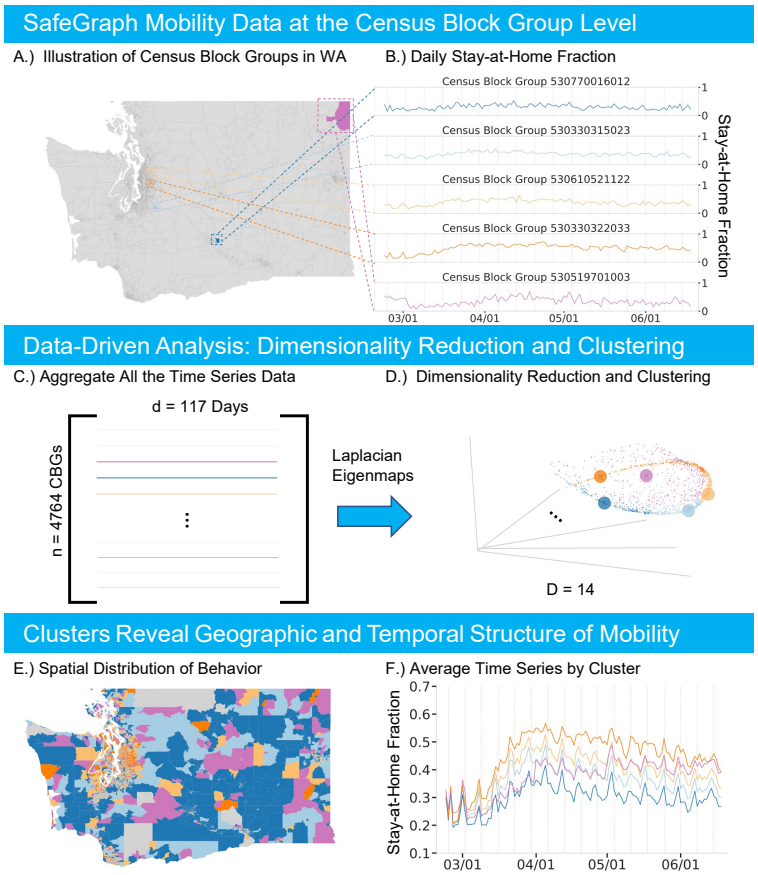


Figure 3.1: **Data-driven analysis overview.** A.) Example CBGs on the map of Washington. B.) Mobility time-series of the example CBGs. C.) Mobility time series are aggregated in a matrix form. D.) 3D visualization of the 14D Laplacian Eigenmaps embedding of the data with clusters highlighted in color. Large dots represent the example CBGs. E.) Clusters plotted on the Washington map. F.) Average mobility time-series for every cluster.

joining their COVID-19 Data Consortium[1]. In this study, we estimate the number of people who stay at home each day by dividing the number of mobile devices that do not leave their homes by the total number of devices in each CBG (i.e., `completely_home_device_count` divided by the `device_count`) [171]. We used data covering 117 days of mobility, starting from February 23, 2020.

We define the daily proportion of devices seen near their homes to be the number of de-

[1] `https://www.safegraph.com/covid-19-data-consortium`

vices in each CBG detected in their home CBG (`destination_cbg` = `origin_census_block_group`) divided by the number of devices associated with the CBG (*device_count*). The proportion of devices that are only detected *away from* their homes each day is 1 minus this proportion. Figure 3.1B. illustrates this daily stay-at-home fraction for five CBGs. We use the most recently released versions of the SafeGraph social distancing data, which is version 2.0 ("v2") for dates before May 10, 2020 and version 2.1 ("v2.1") for later dates [174]. Around May 17, SafeGraph began using "rolling windows" to assign the home census block group of devices instead of batch-updating only at the first of each month [172].

3.2.2 Census and geographic data

We obtained US population data from the 2018 American Community Survey (ACS) product of the US Census Bureau, accessed using the R package `tidycensus` [216]. We used Table B01001 for total population size and population by age estimates by CBG, Table B19013 for median household income, Table B14002 for number currently enrolled in college, Table B25008 for renter vs. owner-occupied housing units, and Tables B07201, B07202, and B07203 for "geographic mobility" (living in same house as last year). We computed a CBG's population density by dividing the 2018 population estimate by the land area of the CBG as reported by the cartographic boundary files.

The US Census provides *cartographic boundary files*, which define simplified shapes of geographic entities designed for plotting. The 2019 shapefiles were downloaded from the US Census Bureau website [2]. The detailed map of Seattle was generated using ESRI's "world topo map" [54] obtained using R's `OpenStreetMap` package [58].

---

[2] `https://www.census.gov/geographies/mapping-files/time-series/geo/cartographic-boundary.html`

3.3 Methods

3.3.1 Linear dimensionality reduction: singular value decomposition

The singular value decomposition (SVD) is a standard linear matrix factorization technique that can be used to reduce the dimensionality of a data matrix [154, 51]. Using the SafeGraph time-series data (§3.2.1), we construct a mobility data matrix for each state. Each state's data matrix has 117 columns (days of mobility data), but a different number of rows depending on the number of census block groups (Table 3.1). Figure 3.1C. illustrates the aggregation of mobility time-series into a data matrix for Washington state. For data matrix normalization, columns are mean subtracted. We perform a standard SVD to find a reduced order set of singular vectors and values for dimensionality reduction; see Supplement §B.1 for more details. The computational code to generate all results and figures in this article are publicly available [114].

3.3.2 Manifold learning and nonlinear dimensionality reduction: Laplacian eigenmaps

Laplacian eigenmaps are a nonlinear manifold learning method that can identify a low-dimensional embedding which optimally preserves local structure of a high-dimensional data manifold [19]. To construct an m-dimensional embedding, the method uses m eigenvectors of the nearest-neighbors graph Laplacian corresponding to the smallest non-zero eigenvalues. The resulting embedding is optimal in the sense that "close" data points on the original manifold are represented by points that are close in the m-dimensional Euclidean embedding space; see equation (3.1) in [19] for more details. We also investigated a wide variety of other nonlinear dimensionality reduction techniques (Supplement §B.2). The Laplacian Eigenmaps algorithm was implemented using the `SpectralEmbedding` function from `sklearn.manifold` module of `scikit-learn` package [155] in `Python 3`. In this work, we used 50 neighbors for the `n_neighbors` parameter. Varying the number of neighbors between 20 and 50 did not significantly change the Laplacian Eigenmap embedding for Washington state (Supplement §B.2).

The optimal effective dimensionality of the embedding was identified using the trustworthiness metric [212] which captures the extent to which a dimensionality reduction technique retains the local structure of the original data manifold from the higher-dimensional space. Trustworthiness was computed as a function of the Laplacian Eigenmap embedding dimensionality; a knee-point detection algorithm was then used to identify the optimal number of dimensions. The Supplement §B.2 provides a detailed description of this analysis for each state. To implement the trustworthiness metric, we used the function `trustworthiness` from `sklearn.manifold` of `scikit-learn` package [155] in `Python 3` with default parameters (5 neighbors, to capture the local structure). For the knee point detection, we used the Kneedle algorithm implemented in `kneed` package [176].

3.3.3 Gaussian Mixture Model Clustering

To interpret the low-dimensional structure revealed by Laplacian Eigenmaps, we apply Gaussian mixture model (GMM) clustering [144]. The GMM is a latent variable model which assumes that the data has sub-populations or clusters which follow Gaussian distributions with parameters governing the centroid location and covariance structure of each cluster. GMMs were implemented using the `mclust` [179] package of `R` (version 4.0 [163]). We leverage the probabilistic formulation of the GMM model as a natural way to quantify uncertainty of the cluster assignment. A more detailed description of the GMM model and uncertainty quantification is provided in Supplement §B.4. We used Bayesian Information Criterion (BIC) to identify the optimal number of GMM components [178, 64, 179]. We applied knee-point detection to the BIC curve using the Kneedle algorithm implemented in `kneed` package [176]; see Supplement §B.5 for more details.

3.3.4 Statistical Testing

To test the difference of the socio-economic covariates distributions between clusters, we used the Kolmogorov-Smirnov [104, 189] test as implemented in `kstest` function of `scipy.stats` package in `Python 3`. To determine the significance of trends of covariates associated with

CBGs in clusters identified by the GMM, we used `jonckheere.test` from `clinfun` package [181] in `R` using 1000 permutations and assuming decreasing trends from cluster A to cluster D (Table 3.1).

3.4 Results

3.4.1 Stay-at-home levels and trends vary across CBGs, but there are distinct motifs that are consistent across four states

The SafeGraph stay-at-home data offers insight into the levels and trends of human mobility at the census block group (CBG) geographic scale during the 2020 COVID-19 pandemic in the United States (Figure 3.1). Nonlinear dimensionality reduction of the time-series data from Washington state revealed a low-dimensional embedding providing insight into the consistency of mobility behavior across CBGs (Figure 3.1D.). Moreover, the embedding and stay-at-home behaviors for Washington are qualitatively similar to those of Georgia, Texas, and California (Figure 3.2). The optimal embedding dimension was 14 for all four states, determined by the trustworthiness metric (§3.3.2, Supplement §B.2). A similar low-dimensional structure in the time-series data can be found with a diversity of nonlinear dimensionality reduction methods; see §3.4.5 and Supplement §B.2 for more details.

The low-dimensional embedding provides insight into the similarity of stay-at-home behavior between CBGs. Figure 3.2 provides a visualization in three embedding dimensions of this coherent structure; note that for each state, certain CBG time-series are more similar to each other and the visualization indicates a large density of CBGs along a distinct, tubular data manifold. Fitting a Gaussian mixture model (GMM) to the stay-at-home time-series in the 14-dimensional embedding space identifies 4 clusters for California, Texas, and Washington and 5 clusters for Georgia (based on knee-point detection in BIC, see Supplement §B.5). For subsequent analyses, we chose 5 as the number of clusters for every state due to the optimal clustering for Georgia, comparisons across the four states (Table 3.1), and the geometric structure of the data (§3.4.2). Figure 3.2 illustrates how the clustering model

groups CBGs in the embedding space (left column); the average mobility time-series for each cluster (right column) highlight the difference in stay-at-home behavior by cluster within a state and also the consistency across all four states. The cluster assignments were robust to model initialization (§3.4.5, Supplement §B.3) and had low associated uncertainty values (quantified in §3.4.5, Supplement §B.2, Supplement §B.4).

Table 3.1: **Number of census block groups (CBGs) in each cluster.**

state	A	B	C	D	E	Total
California	2672	5457	7053	5319	2470	22971
Georgia	1577	1149	1177	960	647	5510
Texas	3154	3960	3687	3283	1635	15719
Washington	1120	1284	1032	674	642	4752

One clear difference between the clusters is their average level of mobility. For example in Washington, the average staying-at-home level increases from the CBGs in the dark blue cluster (cluster D) to the bright orange cluster (cluster A); see Figure 3.1F., representative CBG time-series in each cluster in Figure 3.1B., and Figure 3.2. The order of the clusters along the dense data manifold in the embedding space is aligned with their mean staying-at-home fraction (Figure 3.2). The average time series for clusters D through A do not intersect and are aligned in increasing order on the y-axis. However, the purple cluster E does not follow a similar trend with respect to the dense data manifold, nor the average time-series. For this cluster, we find that the fraction of the devices staying at home increases sharply in May 2020. A similar motif consistently occurs across each state (Figure 3.2); cluster E primarily captures outliers from the primary bulk trends that are continuously distributed across clusters A, B, C, and D. Those outliers are linked to a variety of important sub-populations, explored in more detail in §3.4.4.

The CBG clustering and average time-series by cluster also indicate that the change of behavior over time is different across clusters before April. The speed at which CBGs increased their stay-at-home behavior during a transition period between March and April

(quantified by the slope of a linear fit of the CBG mobility time series during the transition period) is directly correlated with CBG cluster assignment (see Supplement §8). Moreover, the distributions of that speed are statistically significantly different: for every pair of clusters, we were able to reject the null hypothesis that the speed distributions were the same at the significance level $\alpha = 0.001$ using Kolmogorov-Smirnov test. This is also directly evident by looking at this time period and the average stay-at-home trends by cluster (Figure 3.2). For example, the CBGs from the least mobile cluster A also increased their staying-at-home level the fastest.

3.4.2 CBGs with similar temporal mobility patterns are geographically consistent

The CBGs within each mobility cluster (defined in §3.4.1) are geographically connected and have consistent patterns across all four states. The second column of Figure 3.2 illustrates these broad trends which are most visually evident in the distinction between urban, peri-urban, and rural areas. For example in Washington, the Seattle area CBGs mostly belong to the bright orange and light orange most-staying-at-home clusters A and B, the same is true for nearby Bellevue and Redmond. Similarly, in Texas three large orange regions correspond to Dallas, Houston, and Austin. In Georgia, the distinct orange area on the map corresponds to Atlanta and in California we see orange colors around San Francisco, San Jose, and Los Angeles area. Likewise, blue colors – clusters C and D with lower stay-at-home levels – form continuous regions in rural areas on the state maps. CBGs that are close geographically tend to have similar mobility patterns.

Within each state, there is a stark contrast between urban and rural areas (Figure 3.2). For example in Washington, the large metropolitan areas around Seattle and Bellevue are colored orange (clusters A and B) as opposed to larger rural CBGs which belong to blue clusters (C and D). Large cities like Spokane or Yakima also have dense orange coloring (Figure 3.3) suggesting that changes in behavior within urban centers are similar despite being geographically quite distant from each other. The time series column of Figure 3.2 shows that urban areas (orange clusters A and B) stay at home significantly more than rural

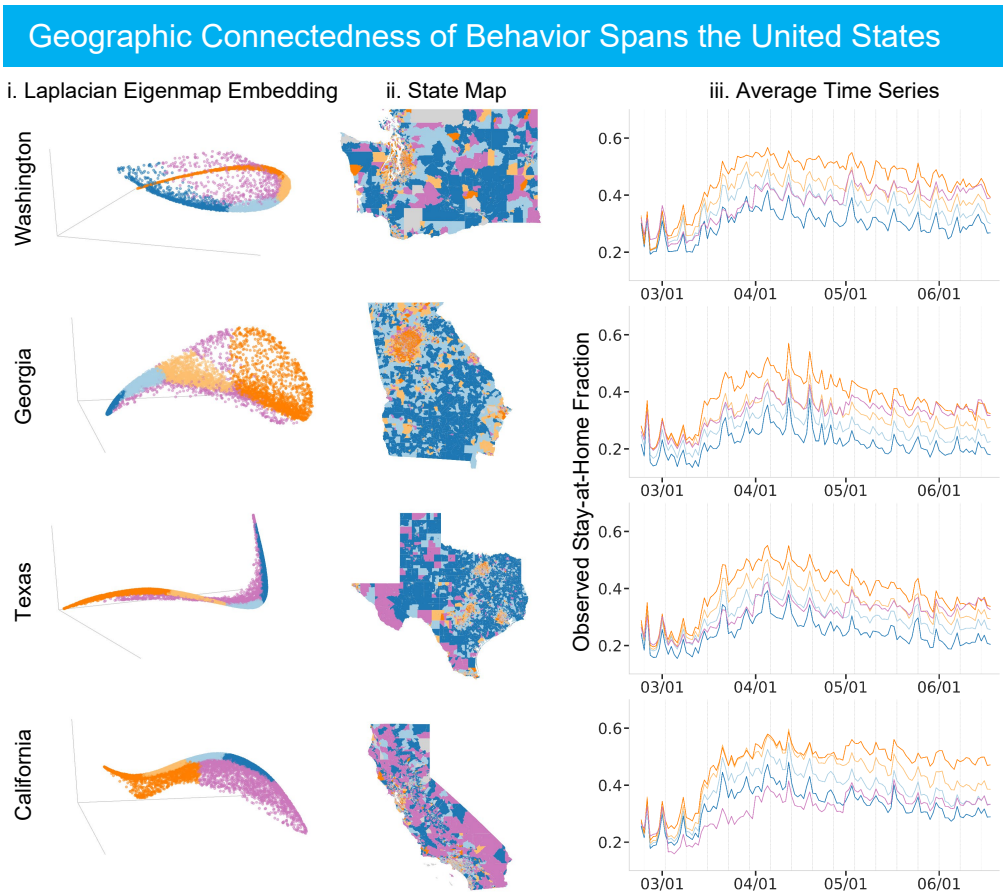


Figure 3.2: **Results are consistent across four states.** Column i. presents the 3D Laplacian Eigenmap visualization of the data manifold. Column ii. shows geographic maps. Column iii. presents average mobility time series for each cluster. Clusters are highlighted in color. It is noted that clusterig was done in 14D (optimal) embedding space.

areas (blue clusters C and D). This observation is consistent across all four states.

This analysis also identifies heterogeneity within the geographic scale of urban centers and rural areas. For example, in Dallas and Seattle there are urban CBGs that belong to blue clusters C and D indicating that they stay at home less than the surrounding areas (Figure 3.3). Moreover, populous cities such as Seattle, Atlanta, Austin and Dallas have distinct geographic groupings of CBGs for clusters A and B within the urban area (Figure 3.3, see Supplement §7 for clustering in Atlanta). The first column of Figure 3.2 clearly presents a

smooth transition in the Laplacian Eigenmap embedding space between the bright orange cluster A that stays at home the most to the dark blue cluster D that stays at home the least. Remarkably, we observe the same on the geographic map. For example, there is a rough radial pattern around Dallas and Austin: bright orange CBGs densely cover the city center and are replaced by light orange, then light blue and eventually dark blue as the distance from the city center increases (see Figure 3.3). That is, the transition is quite consistent – it covers the intermediate colors and the stay-at-home level gradually decreases as distance from the city increases suggesting a more nuanced interpretation about the continuity of behavior across CBGs within urban centers supported by the geometric structure of the data manifold. In the greater Seattle area, the transition is substantially less pronounced especially moving eastward from downtown; note that both a large urban area (Bellevue) and suburb (Redmond, the home to Microsoft) exist to the east of Seattle, both with a higher income population.

Despite the optimal number of clusters being 4 or 5 for each state (§3.4.1), relaxing this criteria and allowing for more clusters provides more granular information within and around urban areas while maintaining consistency with the optimal GMM. This also follows the intuition provided by the illustrations of the nonlinear embedding in three dimensions (left column, Figure 3.2); namely, the embedding is broken into finer grained clusters enabling higher resolution comparisons between CBGs. Supplement §B.6 provides details on altering the number of clusters. Further, a continuous mapping of the data along the dense tubular structure of the data manifold shows the smooth transition across urban, peri-urban, suburban, and rural areas; see Supplement §B.6 for more details. In contrast, the purple cluster E is not wholly on the tubular structure and does not exhibit the same geographically connected characteristics as the other clusters. More detail is provided on this cluster and the possible difference in its subpopulation structure in §3.4.4.

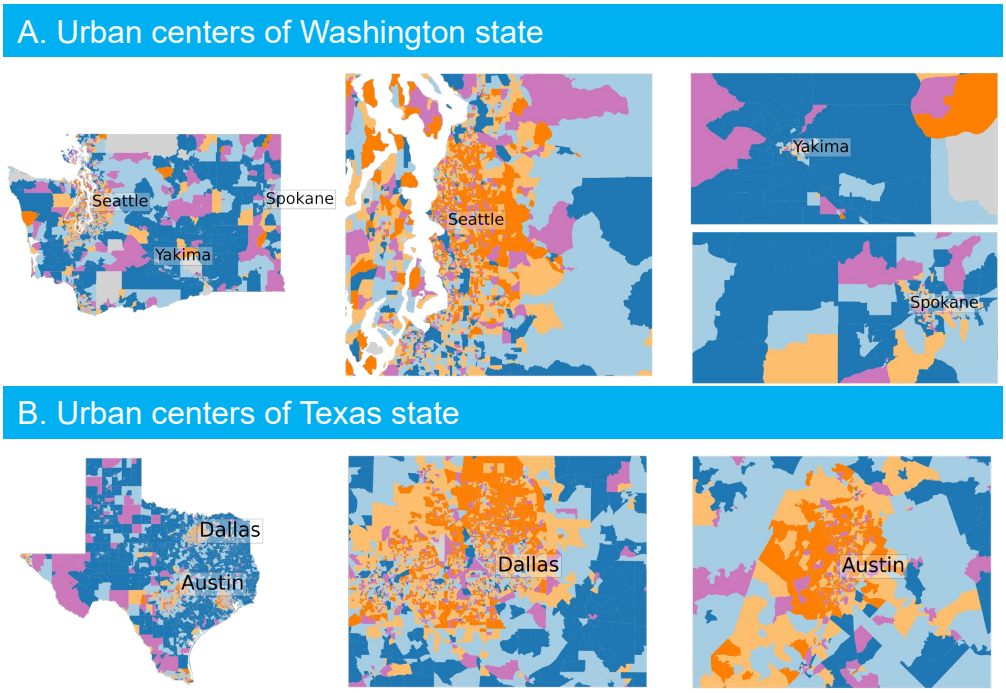


Figure 3.3: **Clustering in metropolitan areas.** Section A: clustering in Washington, Section B: clustering in Texas.

3.4.3 Income, population density, and behavioral data are correlated

Clusters A and B, which on average stayed home the most, included the most densely populated CBGs, while clusters C and D included the more sparsely populated ones (Figure 3.4). High population density is generally an indication of urban populations and low density of rural areas (see maps in Figure 3.2). The CBGs in clusters A and B also had the highest median household incomes (Figure 3.4). In all states, the median stay-at-home fraction, population density, and household income of CBGs had a consistently decreasing trend from clusters A to D, and the Jonckheere–Terpstra test rejects the null hypothesis that these four clusters come from the same distribution of values ($p < 0.01$). Cluster E did not follow these trends and appeared to cover a wider range of values (Figure 3.4).

Cluster E has a higher proportion of people who we expect to have high "geographic

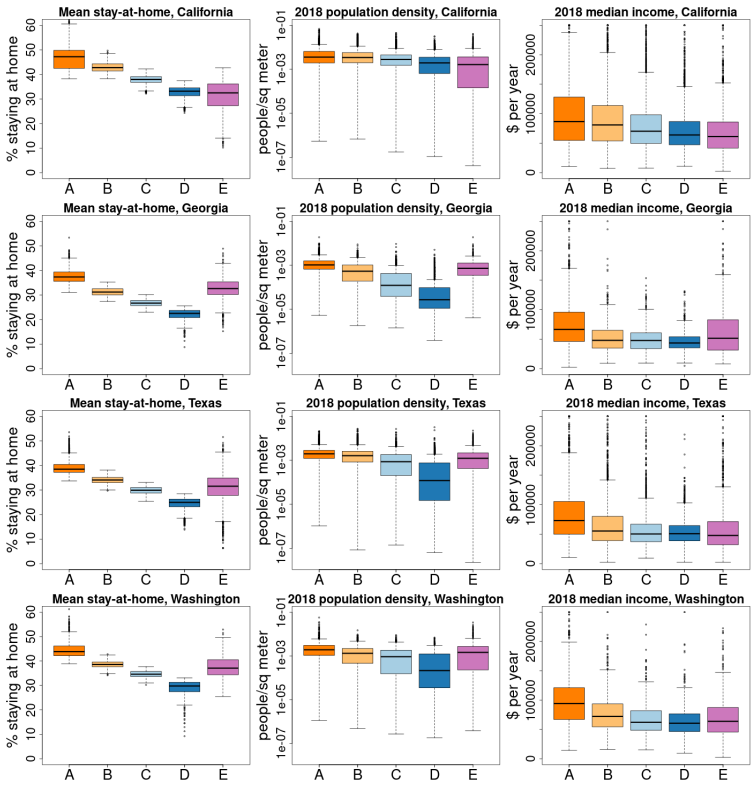


Figure 3.4: **Distribution of stay-at-home behavior and demographic covariates by cluster.** The boxplots present the interquartile range (boxes) and median values (center horizontal lines) of the covariate values for CBGs in each of the five clusters. Whiskers span the 95% range. The "mean stay at home" fraction of a CBG is the mean of the daily percent of mobile devices that stayed completely at home during the time period analyzed.

mobility" (i.e., change residences frequently). Using 2018 ACS estimates, CBGs with a low proportion living in the same house in the previous year or a high proportion of renters, people enrolled in undergraduate or professional degree programs, or who are "young adults" (18 to 29 years old) tended to be in cluster E (Figure 3.5). In California, the proportion of people with high geographic mobility appears to be higher in cluster A than in cluster B.

Upon closer investigation of the location of clusters in the city of Seattle, Washington, the spatial distribution of clusters D and E is consistent with the associations described above (Figure 3.6). The area surrounding the University of Washington, where a large number of undergraduate and graduate students live, is in cluster E, while the University

itself is in cluster D (Figure 3.6, center of map). Cluster E also includes downtown and Lake Union, where a recent influx of young tech workers fueled the development of new apartments. Interestingly, in addition to students and young tech workers in Seattle, cluster E also indicates some very high median income populations on the waterfront of Bellevue and Kirkland that were also highly mobile during this period. Cluster D includes "SODO", the industrial area southwest of downtown, which is less affluent than the populations to the west, east, and north.

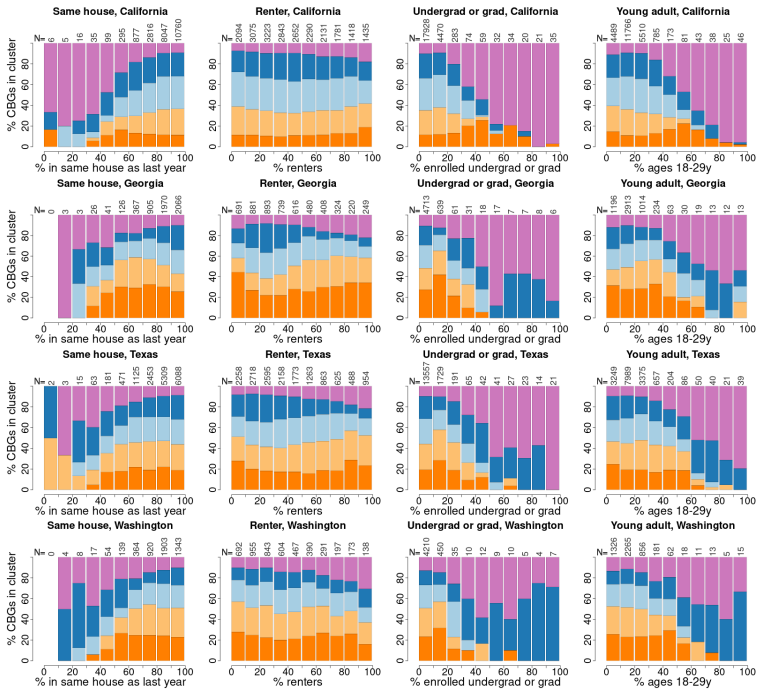


Figure 3.5: **Proportions of census block groups in each cluster by population characteristics associated with geographic mobility.** The fraction of a CBG's population associated with the characteristic is plotted on the x-axis. CBGs are partitioned into 10 equally-spaced bins, defined by the proportion of each CBG's population having the characteristic in 10% increments. The numbers of CBGs belonging to each bin are printed along the top of each panel. The proportion of CBGs in each cluster is plotted as vertically stacked bars for each bin (with cluster A in dark orange on the bottom through cluster E in purple on top).

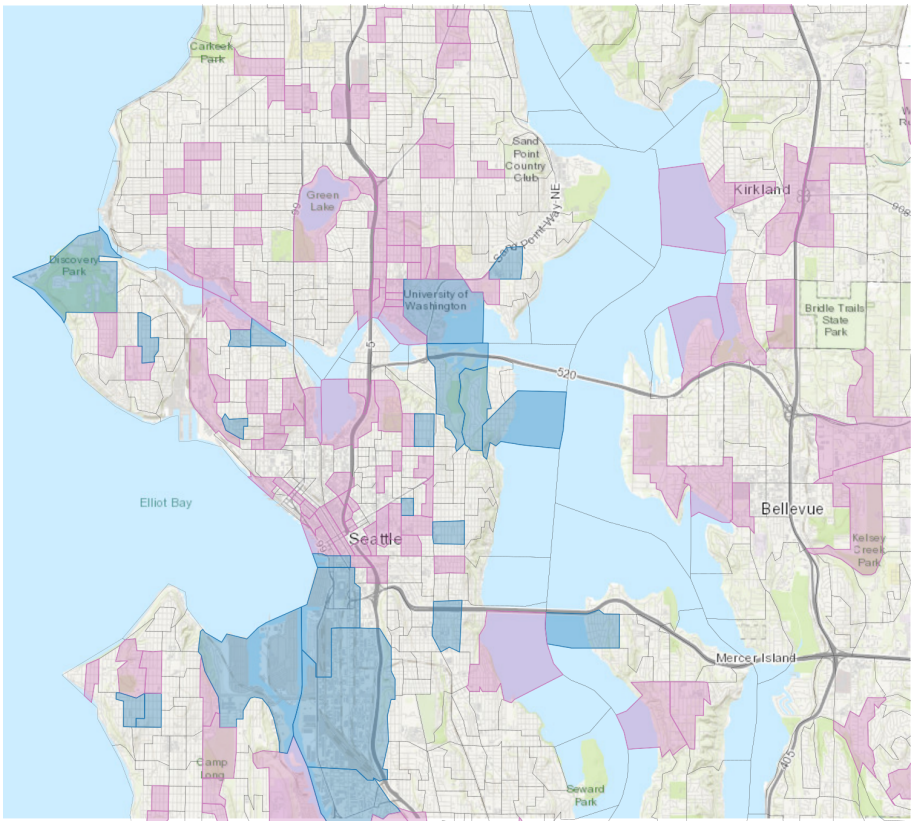


Figure 3.6: **Clusters in the Seattle metropolitan area.** Census block group boundaries are outlined. CBGs belonging to clusters D and E are highlighted in dark blue and purple, respectively.

3.4.4 Analysis reveals areas with likely population turnover early in the pandemic

The available SafeGraph dataset does not allow one to track the movements of individuals, but there are trends consistent with high population turnover. One can track the number of mobile devices that are detected by SafeGraph each day but not in their "home" CBG on a given day, which we call "never-near-home" devices. These devices could be on a trip away from home or they could have moved away entirely.

In March, the fraction of never-near-home devices was highest in cluster E (Figure 3.7 and Supplement §8). On April 1 and again on May 1, the number of devices never near home drops sharply in cluster E but not in the other clusters. This behavior is consistent

with the owners of these devices moving to a new residence and SafeGraph re-assigning these devices to the new residence on the first day of a subsequent month. SafeGraph defines a person's "home" to be the location where the mobile device is detected most at night (from 6pm to 7am) over a 6-week period [174]. If a person spends enough time in a new location, that new location can become the device's "home". These home locations were updated by SafeGraph at the start of each month until mid-May, when SafeGraph changed its procedure for assigning home locations to devices [172, 173]. The high proportion who were never near their "homes" in March and April and the sharp drops in these fractions on April 1 and May 1 in cluster E, and to a lesser extent in cluster D, are consistent with this population moving away. In California, cluster A also has a noticeable decline on May 1 (Supplement §8), which could indicate a high-income group that is geographically mobile. If a large number of people in a region move away, the devices will appear to be "away from home" because their home locations are out-of-date. These clusters will appear to be staying at home less than they really are. This batching artifact appears to be resolved in May 2020, and the stay-at-home fraction in cluster E rises relative to the other clusters.

3.4.5 The nonlinear embedding and clustering results are robust

We investigated the sensitivity of our results to the methodological approach. The cluster assignment for the GMM in the 14 dimensional embedding space is robust. For every state, the maximum uncertainty is below 50% while the third quartile of the cluster assignment uncertainty is close to zero; at least 75% of the CBGs are well separated by a GMM in the 14-dimensional embedding space (see Supplement §B.2, Supplement §B.3). However, given the intrinsic structure of the data (Figure 3.2), the quantification of uncertainty for clustering is consistent with the geometry of the mobility data being more continuous than discrete across clusters A,B,C,D. For example, the uncertain CBG assignments are linked to the boundaries of clusters on or near the tubular data structure. The number of clusters and cluster assignments were optimized according to a standard approach which balances model fit and parsimony(§3.3.3), but the number of clusters could be changed depending on

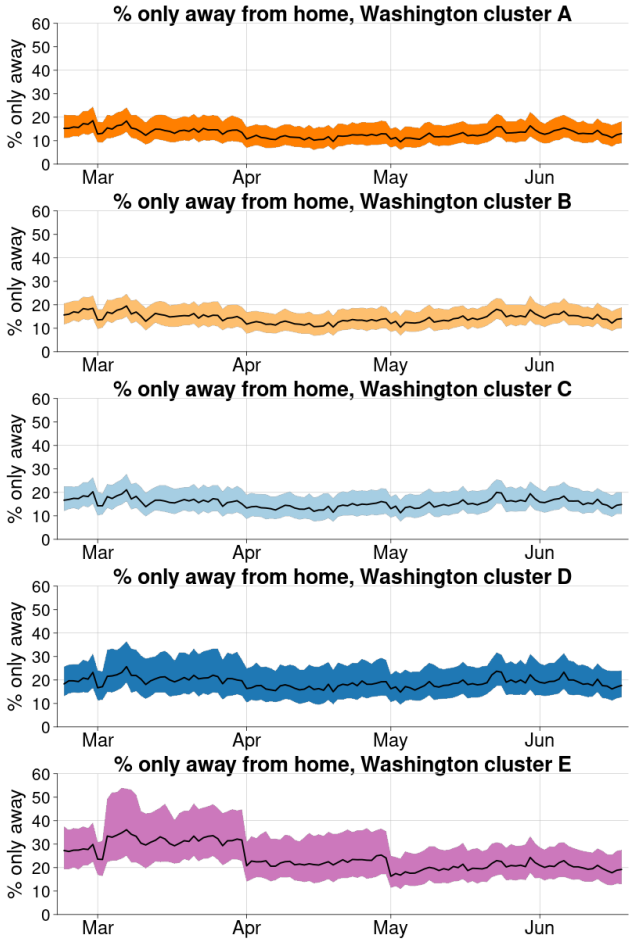


Figure 3.7: **The fraction of devices that are *only* away from their homes each day.** The medians and inter-quartile range of each day's values are shown for each cluster in Washington State.

a desired level of granularity or continuous colormap could be used (Supplement §B.6). It is worth noting in contrast that clustering in the linearly reduced space is highly uncertain; for more details see Supplement §B.1.

We also found consistent results using alternative dimensionality reduction techniques such as Locally Linear Embedding and Isomap. For both Isomap and Locally Linear Embedding, a similar dense tubular data manifold was present in the lower dimensional embedding space (Supplement §B.2). Furthermore, the trustworthiness metric and knee point detection

indicated that all three manifold learning methods agreed that the effective dimensionality of the embedding was between 14 and 16 (Supplement §B.2).

3.5 Discussion and Conclusion

Our results are consistent with other studies linking demographic characteristics to cellphone mobility data during the 2020 SARS-CoV-2 pandemic. Two recent studies using data from SafeGraph found mobile devices from areas with higher median household incomes stayed home more than devices from lower-income areas [87, 92], and this trend occurs in other mobile device datasets [223]. These studies hypothesize that the relationship between income and mobility is due in part to the ability of people with high-paying jobs to work from home. A survey found that about half of adults in Seattle switched to telework because of COVID, with high-income households making the change far more than lower-income (79.3% in households making >\$150,000 per year and 23.5% among those making <\$50,000) [15]. A recent study using another source of cell phone mobility data found that mobility was reduced more in urban than rural England [93], indicating that these trends could generalize beyond the United States.

Several related studies cluster mobility time series by a single demographic characteristic selected *a priori*, such as income [223, 87, 92] or population density [93] or party affiliation [7], to demonstrate behavioral differences with respect to that characteristic. Alternatively, one could reduce the time series to a summary statistic, such as average stay-at-home level over a particular time window, and study the relationship between that metric and several demographic covariates. In contrast, our methodological approach is broader; we measure similarities between complete time series, which allowed us to identify population clusters that had a distinct *change* in behavior, which would have been hidden if we had clustered by average behavior over time. Notably, we have identified features in the SafeGraph stay-at-home data that strongly suggests a mass migration out of several major metropolitan areas, especially in CBGs that have high proportions of young adults, renters, or students (§3.4.4). The closure of college of campuses and widespread job losses in March and April led many,

especially young adults, to move [36, 66]. Moreover, the map presented in Figure 3.6 also matches our own intuition of where students of the University of Washington live, both adjacent to the university as well as more distant rental housing along bike and metro commuting lines. Similarly, the high-migration census block groups identified near South Lake Union tends toward a younger, professional population working at technology companies such as Amazon, and CBGs on the waterfront with high income populations in nearby cities such as Bellevue and Kirkland have a similar outward migration trend.

Identifying the population that moved early in the pandemic is a direct consequence of using a data-driven, equation-free approach. The approach has been integral to revealing the heterogeneity, but also the consistency, of mobility patterns across California, Georgia, Texas, and California; it has enabled a multi-scale geographic perspective on behavior allowing insights at the state, urban-rural, peri-urban, and suburban scale. Recent efforts have also utilized clustering of mobility time-series data specifically for analyzing SafeGraph stay-at-home data in Atlanta [87]. Our approach, though, is substantially broader in scope; identifying the low-dimensional embedding of the data enables a characterization of the geometric structure and the relatedness of each CBG mobility behavior. Moreover, we found utilizing a nonlinear dimensionality reduction techniques such as Laplacian Eigenmaps for analyzing mobility time-series data is essential (Supplement §B.1) mirroring recent developments from dynamical systems focused on the development of equation-free methods for analyzing measurement data collected from complex systems [37]. We have also leveraged clustering as a tool to interpret the similarity of mobility behavior between CBGs even in the reduced nonlinear embedding; we found that clusters allowed for comparisons of mobility characteristics (Figure 3.1), generalization across four states (Figure 3.2), and also correlation with socioeconomic factors (Figure 3.4 and Figure 3.5). The nonlinear embedding, however, offers a more nuanced perspective about the similarity of mobility behavior between CBGs. For example, the visualization in three dimensions and the clustering results suggests a much smoother and continuous geometric structure of relatedness for CBGs assigned to clusters A,B,C,D (Figure 3.2 and Supplement §6). This helps frame the clustering

results and socioeconomic factor correlation analysis. Further, the embedding provides a richer characterization of the underlying complexity in mobility behavior.

We acknowledge several limitations of the mobility data and challenges in linking behavior to demographic variables. SafeGraph aggregates mobility data from many uncoordinated sources on the locations of millions of cell phones. These phones are not systematically tracked, and the GPS data might not be precise. The data are then aggregated by census block group and filtered to preserve the privacy of the mobile device owners. It is difficult to ascertain how well a set of mobility data represents the general population [102, 72]. Different states, and segments of the population, have different levels of coverage that are hard to correct for [193]. This is further complicated by likely gaps in coverage for high-risk populations such as migrant agricultural workers. However, the associations we found between mobility and other factors are consistent with those found in other datasets and are quite plausible. We studied the fraction of mobile devices that stayed at home each day, but this is just one metric than can be derived from the mobility data. Other measures, such as the mean length of time spent outside the home, the distance traveled from the home, or even the number of trips to stores, could provide additional insight into the population's response to the pandemic. The demographic data in this study was from the 2018 American Community Survey, which we believe generally reflects the population in 2020 but might not accurately characterize the demographics of the most rapidly changing areas. We cannot establish the direct cause of the differential reductions in mobility using these data. We use demographic and socioeconomic variables at the census block group level, which could lead us to ecological fallacies, and many of these variables are tightly linked, thus, disentangling their effects is not straightforward and could be counterproductive.

Despite these challenges, population mobility data and connections to behavior can inform public health policy makers. Population behavior is a key component to understanding disease transmission dynamics; mobility data and the methods contained in this chapter helps quantify the change in population behavior during the pandemic. Policy makers can use this tool to assess the impacts of policy, especially important as COVID-19 cases start to resurge

in the United States during a period of quarantine fatigue. We have also demonstrated that these data, analyses, and setting-specific information can provide epidemiologically relevant insights such as we uncovered around urban migration events. We believe the research in the chapter will provide insights for policymakers as they consider more modern, optimized, and targeted intervention strategies.

Chapter 4

WHERE DO MODELS GO WRONG? PARAMETER-SPACE SALIENCY MAPS FOR EXPLAINABILITY

This chapter is based on joint work with Manli Shu, Eitan Borgnia, Furong Huang, Micah Goldblum and Tom Goldstein [118].

4.1 Introduction

With the widespread deployment of deep neural networks in high-stakes applications such as medical imaging [98], credit score assessment [226], and facial recognition [45], practitioners need to understand why their models make the decisions they do. In fact, "right to explanation" legislation in the European Union and the United States dictates that relevant public and private organizations must be able to justify the decisions their algorithms make [210, 55]. Diagnosing the causes of system failures is particularly crucial for understanding the flaws and limitations of models we intend to employ.

Conventional saliency methods focus on highlighting sensitive pixels [186] or image regions that maximize specific activations [53]. However, such maps may not be useful in diagnosing undesirable model behaviors as they do not necessarily identify areas that specifically cause bad performance since the most sensitive pixels may not be the ones responsible for triggering misclassification.

We develop an alternative approach to saliency which highlights network parameters that influence decisions rather than input features. These parameter saliency maps yield a number of useful analyses:

- We verify that identified salient parameters are indeed responsible for misclassification by showing that turning these parameters off improves predictions on the associated

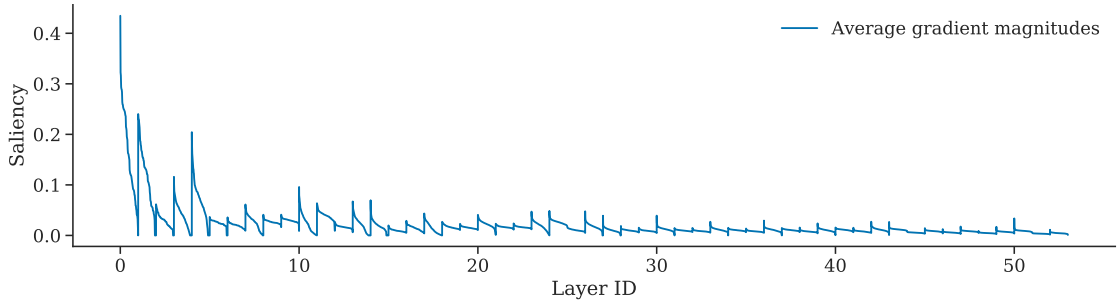


Figure 4.1: **Filter-wise parameter saliency profile.** ResNet-50 filter-wise saliency profile (without standardization) averaged over samples in ImageNet validation set. The filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis.

samples more than turning off the same number of random or least salient parameters.

- Nearest neighbors in parameter saliency space share common semantic information. That is, samples which are misclassified for similar reasons and cause similar parameters to malfunction are semantically similar.

- We further validate the link between salient parameters and network misclassification errors by observing that correcting a small number of the most salient parameters by fine-tuning them on *a single* sample results in error correction on *other* samples which were misclassified for similar reasons.

- By first identifying the network parameters responsible for an erroneous classification, we can then visualize the image regions that interact with those parameters and trigger the identified misbehavior obtaining interpretable insights into model mistakes and neural network's reliance on spurious correlations.

4.2 Related Work

4.2.1 Neural network interpretability and parameter importance

A major line of work in neural network interpretability focuses on convolutional neural networks. Works visualizing, interpreting, and analysing feature maps [242, 238, 148, 134] provide insight into the role of individual convolutional filters. These methods, together with other approaches for filter explainability [18, 246, 247] find that individual convolutional filters often are responsible for specific tasks such as edge, shape, and texture detection.

The ideas of measuring neural network parameter importance has been studied in multiple contexts. Notions of neuron [180], feature [142, 184, 195, 159] and parameter [194, 215] importance have been used for AI explainability, manipulating model behavior [17], model debugging [132, 243] and pruning [1, 123, 197, 170, 235] for improving model performance.

4.2.2 Input space saliency maps

A considerable amount of literature focuses on identifying input features that are important for neural network decisions. These methods include using deconvolution approaches [242] and data gradient information [186]. Several works build on these ideas and propose improvements such as Integrated Gradients [199], SmoothGrad [188], and Guided Backpropagation [192] which result in sharper and more localized saliency maps. Other approaches focus on the use of class activation maps [248] with improvements incorporating gradient information [180] and more novel approaches to weighting the activation maps [218]. In addition, various saliency methods are based on manipulating the input image [62, 242, 159, 4]. Another line of work is aimed at evaluating the effectiveness of saliency maps [2, 8].

Our work combines the ideas of saliency maps and parameter importance and evaluates saliency directly on model parameters by aggregating their absolute gradients on a filter level. We leverage the resulting parameter saliency profiles as an explainability tool and develop an input-space saliency counterpart which highlights image features that cause specific filters to malfunction to study the interaction between the image features and the erroneous filters.

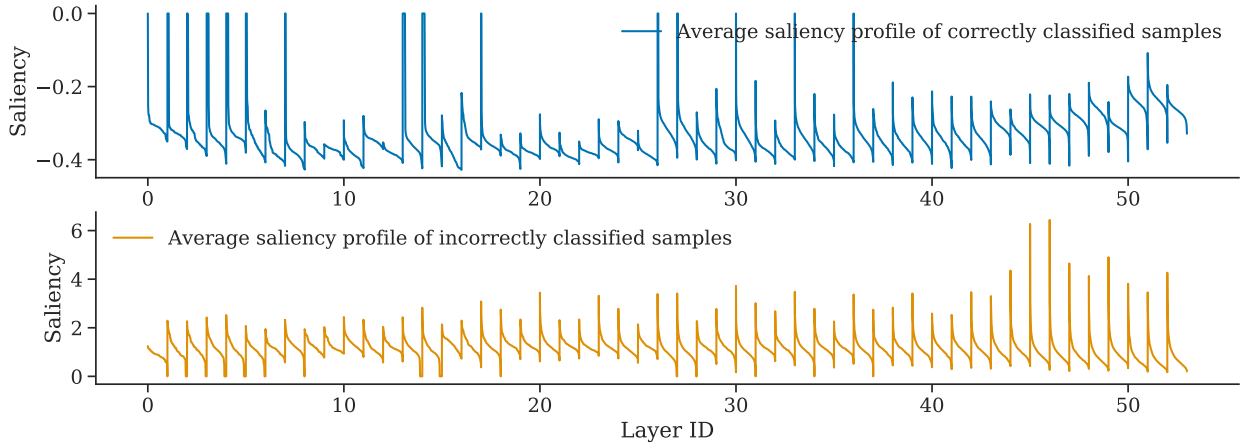


Figure 4.2: **Standardized filter-wise saliency profiles, correctly vs incorrectly classified samples.** Top: Standardized saliency profiles averaged over correctly classified samples in the ImageNet validation set. Bottom: Standardized saliency profiles averaged over incorrectly classified samples in the ImageNet validation set. On both panels, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis. Both profiles are generated on ResNet-50.

4.3 Method

It is known that different network filters are responsible for identifying different image properties and objects [242, 238, 148, 134]. This motivates the idea that mistakes made on wrongly classified images can be understood by investigating the network parameters, rather than only the pixels, that played a role in making a decision. We develop parameter-space saliency methods geared towards identifying and analyzing neural network parameters that are responsible for making erroneous decisions. Central to our method is the use of gradient information of the loss function as a measure of parameter sensitivity.

4.3.1 Parameter Saliency Profile

Let x be a sample in the validation set D with label y, and suppose a trained classifier has parameters θ that minimize a loss function $\mathcal{L}$. We define the *parameter-wise* saliency profile

of x as a vector $s(x, y)$ with entries $s(x, y)_i := |\nabla_{\theta_i} \mathcal{L}_\theta(x, y)|$, the magnitudes of the gradient of the loss with respect to each model parameter. Because the gradients on *training* data for a model trained to convergence are near zero, it is important to specify that D be a validation, or holdout, set. Intuitively, a larger gradient norm at the point (x, y) indicates a greater inefficiency in the network's classification of sample x, and thus each entry of $s(x, y)$ measures the suboptimality of individual parameters.

Aggregation of parameter saliency. Convolutional filters are known to specialize in tasks such as edge, shape, and texture detection [238, 18, 148]. We therefore choose to aggregate saliency on the filter-wise basis by averaging the gradient magnitudes of parameters corresponding to each convolutional filter. This allows us to isolate filters to which the loss is most sensitive (*i.e.* those which, when corrected, lead to the greatest reduction in loss).

Formally, for each convolutional filter $\mathcal{F}_k$ in the network, consider its respective index set α_k, which gives the indices of parameters corresponding to the filter $\mathcal{F}_k$. The *filter-wise* saliency profile of x is defined to be a vector $\overline{s}(x, y)$ with entries

$$\overline{s}(x, y)_k := \frac{1}{|\alpha_k|} \sum_{i \in \alpha_k} s(x, y)_i, \tag{4.1}$$

the parameter-wise saliency profile aggregated by averaging on the filter level.

Standardizing parameter saliency. Figure 4.1 exhibits the ResNet-50 [82] filter-wise saliency profile averaged over the ImageNet [43] validation set, where filters within each layer are sorted from highest to lowest saliency. One clear observation is the difference in the scale of gradient magnitudes – shallower filters are more salient than deeper filters. This phenomenon might occur for a number of reasons. First, early filters encode low-level features, such as edges and textures, which are active across a wide spectrum of images. Second, typical networks have fewer filters in shallow layers than in deep layers, making each individual filter more influential at shallower layers. Third, the effects of early filters cascade and accumulate as they pass through a network.

To isolate filters that uniquely cause *erroneous* behavior on particular samples, we find

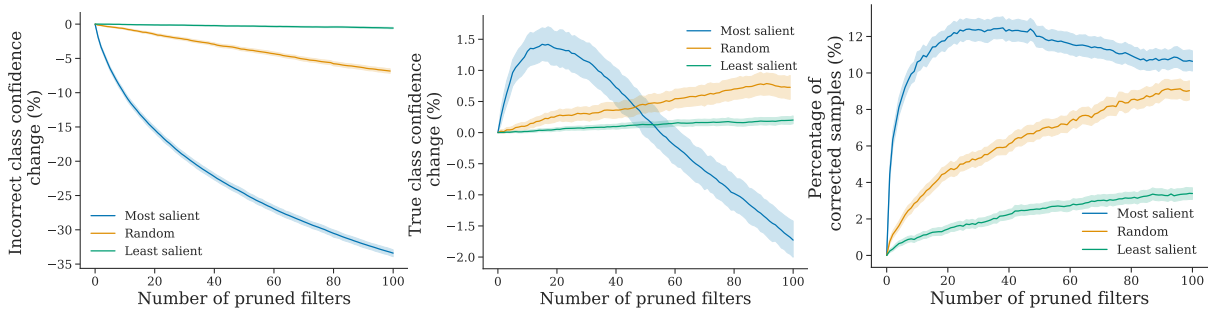


Figure 4.3: **Effect of turning salient filters off.** (a) Change in incorrect class confidence score. (b) Change in true class confidence score. (c) Percentage of samples that were corrected as the result of pruning filters. These trends are averaged across all images misclassified by ResNet-50 in the ImageNet validation set. The error bars represent 95% bootstrap confidence intervals.

filters that are abnormally salient for a sample, x, but not for others. That is, we further standardize the saliency profile of x with respect to all filter-wise saliency profiles of D.

Formally, let μ be the average filter-wise saliency profile across all $x \in D$, and let σ be an equal-length vector with the corresponding standard deviation for each entry. We use these statistics to produce the standardized filter-wise saliency profile as follows:

$$\hat{s}(x, y) := \frac{|\overline{s}(x, y) - \mu|}{\sigma}. \tag{4.2}$$

The resulting tensor $\hat{s}(x, y)$ is of length equal to the number of convolutional filters in the network, and we henceforth call it the saliency profile for sample x. By standardizing saliency profiles, we create a saliency map that activates when the importance of a filter is unusually strong relative to other samples in the dataset. This prevents the saliency map from highlighting filters that are uniformly important for all images, and instead focuses saliency on filters that are uniquely important and serve an image-dependent role. In the rest of the paper, unless explicitly noted otherwise, we use $\hat{s}(x, y)$ and refer to it as *parameter saliency*.

Incorrectly classified samples are more salient. Empirically, we observe the saliency

profiles of incorrectly classified samples exhibit, on average, greater values than those of correctly classified examples. This bolsters the intuition that salient filters are precisely those malfunctioning — if the classification is correct, there should be few malfunctioning filters or none at all. Moreover, we see deeper parts of the network appear to be most salient for the incorrectly classified samples while earlier layers are often the most salient for correctly classified samples. An example of these behaviors for ResNet-50 is shown in Figure 4.2 which presents standardized filter-wise saliency profiles averaged over the correctly and incorrectly classified examples from the ImageNet validation set. Additionally, we note the improved relative scale of the standardized saliency profile across different layers compared to the absolute gradient magnitudes in Figure 4.1. Saliency profiles for other architectures could be found in Appendix C. Henceforth, we will focus specifically on saliency profiles of *misclassified* samples in order to explore how neural networks make mistakes.

4.3.2 Input-Space Saliency for Visualizing How Filters Malfunction

The parameter saliency profile allows us to identify filters that are most responsible for mistakes and erroneous network behavior. In this section, we develop an input-space counterpart to our parameter saliency method to understand which features of the image affect the saliency of particular filters. [68] show that the gradient information of a network is invertible, providing a link between input space and parameter saliency space. This work, along with existing input-space saliency map tools [186, 192, 188, 248, 180], inspires our method.

Given a parameter saliency profile $\hat{s} = \hat{s}(x, y)$ for an image x with label y, our goal is to highlight the input features that drive large filter saliency values. That is, we would like to identify image pixels altering which can make filters more salient. To this end, we first select some set F of the most salient filters that we would like to explore. Then, we create a boosted saliency profile s'_F by increasing the entries of $\hat{s}$ corresponding to the chosen filters

F:

$$(s'_F)_i = \begin{cases} \hat{s}_i, & i \notin F, \\ k\hat{s}_i, & i \in F, \end{cases} \tag{4.3}$$

$k > 1$ (we used $k = 100$ in our experiments). Now, we can find pixels that are important for making the chosen filters F more salient and, equivalently, making the filter saliency profile $\hat{s}(x, y)$ close to the boosted saliency profile s'_F by taking the following gradients:

$$M_F = |\nabla_x D_C(\hat{s}(x, y), s'_F)|, \tag{4.4}$$

where $D_C(\cdot, \cdot)$ is cosine distance.

The resulting input saliency map M_F contains input features (pixels) that affect the saliency of the chosen filters F the most.

4.4 Experiments

In this section, we aim to validate the meaningfulness of our parameter saliency method. First, we verify that salient parameters are indeed responsible for misclassification by showing on the dataset level that turning them off improves predictions on the associated samples more than turning off the same number of random or least salient parameters.

We then find that samples which cause similar filters to malfunction are semantically similar. We also show on the dataset level that fine-tuning a small number of the most salient parameters on a *single* sample results in error correction on *other* samples which were misclassified for similar reasons and cause similar parameters to malfunction.

We then use our input-space saliency technique in conjunction with its parameter-space counterpart as an explainability tool to explore how neural networks make mistakes and how salient filters interact with visual input features.

We evaluate our saliency method in the context of image classification on CIFAR-10 [107] and ImageNet [43]. Images we use for visualization, unless otherwise specified, are sampled from ImageNet validation set. Throughout the experiments, we use a pre-trained ResNet-18

(a) Great pyrenees ↔ Kuvasz (b) Basset hound ↔ Beagle

Figure 4.4: **Examples of nearest neighbors in parameter saliency space (from ImageNet)**.

classifier [82] on CIFAR-10 and a pre-trained ResNet-50 on ImageNet. Both models are trained in a standard fashion on the corresponding dataset[1][2].

4.4.1 Turning Off Salient Filters

We begin validating our parameter-space saliency maps by observing the effect of turning off the salient filters. In order to remove the influence of a particular salient filter, we prune it – zero out the filter weights and also the biases of the associated batch normalization layers. This procedure guarantees that the corresponding input feature map to the next convolutional layer is always zero.

We gradually increase the number of pruned most salient filters and track three metrics: the change in the incorrect class confidence, the change in the true class confidence, and the percentage of the samples that flip their label to the correct class. In every case, we compare pruning the most salient filters against pruning the same number of random filters and the least salient filters. These experiments are performed on the dataset level: we average the trends across all misclassified images in the ImageNet validation set.

---

[1]https://github.com/kuangliu/pytorch-cifar (under MIT license)

[2]https://github.com/pytorch/vision (under BSD 3-Clause License)

As shown in Figure 4.3, pruning the most salient filters is significantly more effective for decreasing the incorrect class confidence than random or least salient filters. Specifically, gradually pruning the top 100 salient filters achieves up to 30% drop in the incorrect class confidence score while pruning random filters yields only about 7% decrease. We also note that pruning the least salient filters does not produce any effect on the incorrect class confidence.

We repeat the same experiment with the true class confidence and observe that the highest true class confidence gain occurs when we prune around 20 most salient filters. Pruning enough salient filters eventually leads to a gradual decrease in the true class confidence. We note that this behavior is expected since we are destroying, not correcting, the inference power of all of the most sensitive filters for an image, some of which may be essential for inference. Finally, pruning random filters provides a much slower increase in the true confidence class while the least salient filters again do not produce a significant effect.

In addition, we count the number of images that were corrected as a result of pruning and find that pruning around 30 most salient filters results in the best correct classification rate of 12%. Similar to the true class confidence, the trend decreases beyond this point. Pruning random filters increases the percentage of corrected samples at a much slower rate and does not perform better than the most salient filters when pruning up to 100 filters. Notably, pruning the least salient filters manages to correct a nontrivial number of samples but still much smaller than pruning random filters.

These experiments suggest that the identified salient parameters are indeed responsible for misclassification since turning these parameters off improves predictions on the associated samples more than turning off the same number of random or least salient parameters.

4.4.2 Nearest Neighbors in Parameter Saliency space

We validate the semantic meaning of our saliency profiles by clustering images based on the cosine similarity of their profiles. In this section, we present visual depictions of a nearest neighbor search among all images in the ImageNet validation set. We also conduct this

Figure 4.5: **Neighbors in parameter saliency space found using only early or only deep layers.** The reference image is in the first column. Images in the top row resemble the reference image in the saliency on early layers of VGG-19, and images in the bottom row are found using deeper layers.

analysis on CIFAR-10 images, and this can be found in Appendix C.

We find that the nearest neighbors of misclassified images in saliency space are mostly other misclassified images from the same pair of predicted and true classes but possibly in reverse order. For example, in Figure 4.4, the reference image in (a) is a great Pyrenees misclassified as kuvasz, and the 4 images with the most similar profiles exhibit either the same misclassification or the reverse (i.e., kuvasz misclassified as great Pyrenees). Intuitively, the common salient parameters across these neighbors are those which are important for discriminating between the two classes in question but are not well-tuned for this purpose.

Note that we find the concept of "being similar" in parameter saliency space to be different from the one in image space. The nearest neighbors we find are often not similar in a pixel-wise sense, but rather they are similar in their reason for causing misclassification. For example, images in Figure 4.4 (b) are beagles mistaken by a network for basset hounds and vice versa. We find that these pictures are either taken from a high angle or do not include the dog's legs, making the leg length, a major distinction between the two breeds, indistinguishable from the picture. We include more example images along with their nearest neighbors in Appendix C.

In addition, we compute nearest neighbors when only considering filters in a specific range

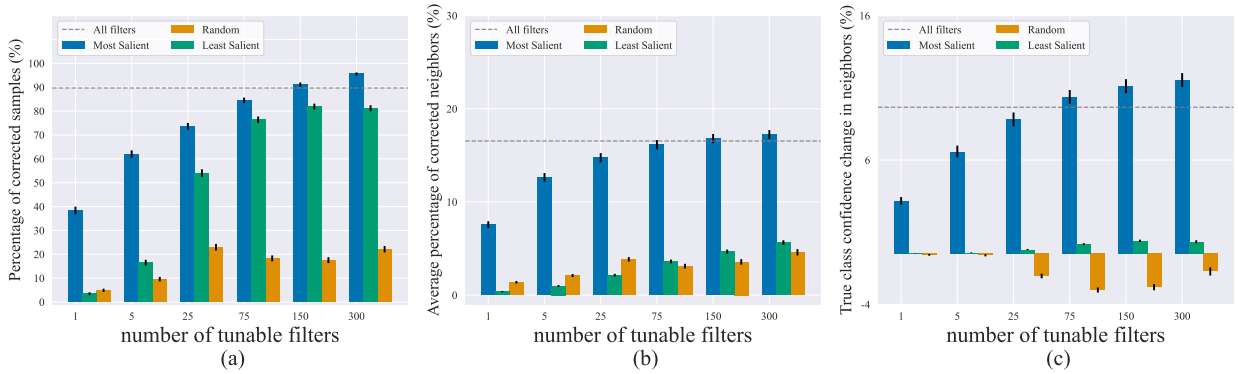


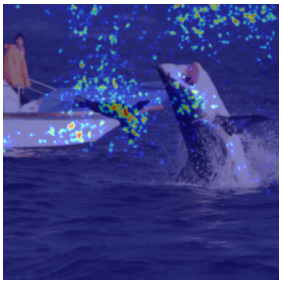
Figure 4.6: **Effect of updating a limited number of filters.** (a) Percentage of misclassified samples corrected after fine-tuning. (b) Average percentage of nearest neighbors that are also corrected after fine-tuning. (c) Average change in the confidence score of the true class among nearest neighbors. The horizontal dashed line in each plot is the effect of updating the entire network.

of layers in order to visualize the types of misbehavior triggered by network components (filters) at various network depths. We search for similar images using parameter saliency in the shallow and deep layers of a VGG-19 network vgg, which we divide into the shallow and deep parts that respectively occur up to and after layer `relu4_1`. The top row of figure Figure 4.5 shows neighbors found using shallow parameters, which share basic image attributes such as color histogram, while images in the bottom row share more abstract similarities.

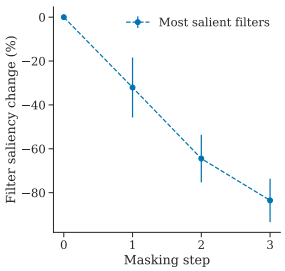
4.4.3 Correcting Mistakes by Fine-Tuning Salient Filters

To validate that salient filters are more responsible for the erroneous behavior of neural networks, we show that updating a limited number of salient filters on a single misclassified image can correct the mistakes made by a neural network on other images that were misclassified because of similarly malfunctioning filters – on nearest neighbors of that image in the parameter saliency space. In this experiment we fine-tune a pretrained ResNet-50 for one step on a single image for which the network makes a wrong prediction and track the

(a) Reference image
4.07% great white shark
78.36 % **killer whale**



(b) Input-space saliency



(c) Saliency values change of the most erroneous filters across masking experiments

(d) Mask seal
11.83% great white shark
53.47% **killer whale**

(e) Mask boat and seal
43.93% **great white shark**
24.24% killer whale

(f) Extended mask
58.73% **great white shark**
12.88% killer whale

Figure 4.7: **Interaction between input features and salient filters.** (a) Reference image of "great white shark" misclassified by ResNet-50 as "killer whale" with confidence scores. (b) Input-space saliency visualization. Pixels that cause the top 10 salient filters to have high saliency. (c) Change in saliency values of the erroneous filters across masking experiments. The vertical bars represent the standard deviation of the change across 10 most salient filters. (d)-(f) Masking experiments.

model's performance on the image's nearest neighbors found through the process introduced in Section 4.4.2. While filter saliency profiles are standardized with respect to the ImageNet validation set, we find the nearest neighbors from an independent test set – ImageNet-v2, collected by imagenetv2 separately from the original ImageNet data. This way, the label information of the nearest neighbors is not used in fine tuning in any way.

We restrict the number of tunable filters to be no more than 1.0% of the total number of filters in the network, and we update the chosen filters by taking one step of gradient descent

with a normalized step. We compare fine-tuning the most salient filters with two other choices of tunable filters: the least salient filters and random filters. We use misclassified images from the ImageNet validation set, making up to over 10,000 samples. We evaluate the effect of fine-tuning on each of these images independently and find the corresponding nearest neighbors from a holdout ImageNet-v2 test set where we limit the search scope exclusively to misclassified images too.

In Figure 4.6, we compare the average performance of our three choices of tunable filters under three metrics. Panel (a) presents a the percentage of samples that are corrected after fine-tuning as a sanity check and we see that updating 150 most salient filters ($\sim$0.6% of total filters) is enough to achieve the same percentage of corrections as updating the entire network.

The second and third metrics evaluate the effect on the nearest neighbors of the training sample (Figure 4.6 (b),(c)). Note that for a given training sample, its nearest neighbors are not involved in our one-step single sample fine-tuning process. By tracking model predictions and true class confidence scores among the 10 nearest neighbors of each sample, we find that fine-tuning salient filters is significantly more effective than other choices. Results in Figure 4.6, (b) and (c), also imply that the nearest neighbors found using our method are the images that are wrong for similar reasons and that they can be corrected altogether by only updating the salient filters on a single image. We note that we do not propose a new pruning or fine-tuning method. Rather, we use these experiments to verify that the salient filters are indeed responsible for misclassification.

4.4.4 Interpretable Network Failures: Visualizing Input Features Which Trigger Filter Malfunctions

Case study. To present an example of how our approach can be used as an explainability tool, we begin this section with a case study of an image misclassified by ResNet-50 as "killer whale" (Figure 4.7(a)). The correct label of the image is "great white shark". Our goal is to study the interaction between the most salient filters and input features. We first

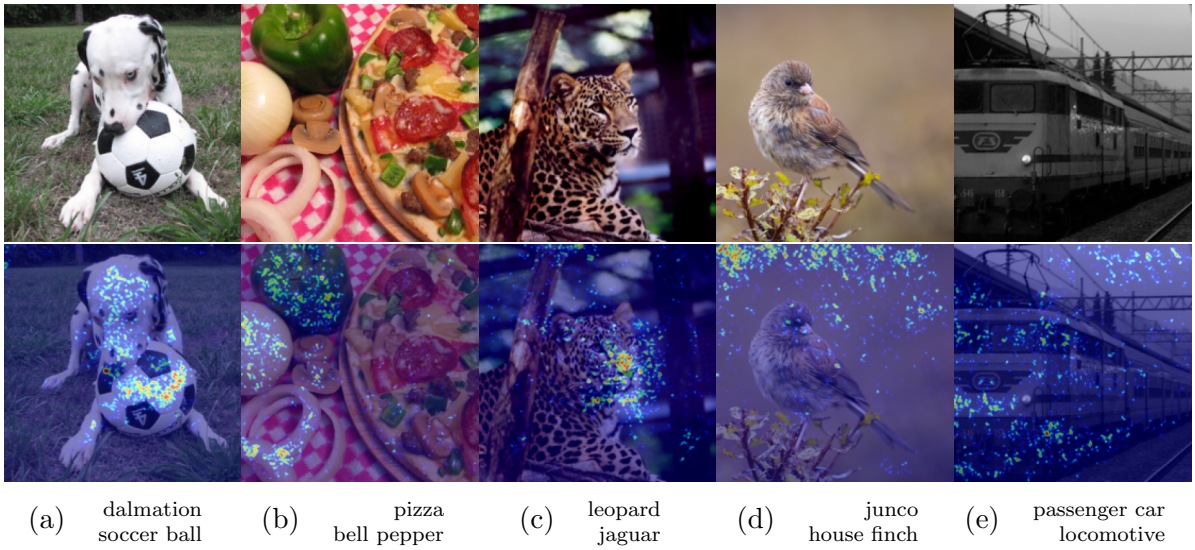


Figure 4.8: **Different types of network mistakes.** All of the presented images are misclassified by ResNet-50. The correct class label is specified in the top row and the incorrect class label – in the bottom row of the subcaption on each panel. (a)-(b) The target object is confused with another object in the image. (c) A regular mistake. The salient pixels are focused on the target object features which confuse the network. (d) Background features confuse the network. (e) An example of a noisy label where the network is "more correct" than the target label. These are examples where masking top 5% of the salient pixels corrects the misclassification.

identify filters most responsible for misclassification by computing the filter saliency profile and visualize parts of the image that drive the high saliency values for those filters using the input-space saliency counterpart (as introduced in Section 4.3.2).

Panel (b) of Figure 4.7 presents our image-space visualization for the ten most salient filters – the pixels which trigger misbehavior in these filters are highlighted. For example, we see that the seal and boat are both triggers. One natural hypothesis is that the seal looks like a killer whale to the network and is the source of the classification error. We check this by masking out the seal (Figure 4.7 (d)) . However, although the probability of "killer whale" goes down and the probability of the correct class increases, the network still misclassifies the image as "killer whale".

Now, if we mask out exactly the most salient areas of the image according to our visualization (see Figure 4.7 (b), (e)), the network manages to flip the label of the image and classify it correctly. If we extend our mask to the less pronounced, but still salient, areas of the image as in Figure 4.7 (f), we observe that the correct class confidence increases even more while the probability of the incorrect "killer whale" label further decreases. Additionally, we find that masking out the non-salient parts of the image results in even worse misclassification confidence than that of the original image (see Appendix C). In order to further investigate the effect of the salient region, we pasted it from this image onto other great white shark images (see Appendix C) and observed that this drives the probability of "killer whale" up for 39 out of 40 examples of great white sharks from the ImageNet validation set with an average increase of 3.75%.

Our experiments suggest that secondary objects in the image are associated with the misclassification. However, we see that the erroneous behavior of the model does not just stem from classifying a non-target object in the image. It is possible that the model correlates the combination of sea creatures (e.g. a seal) and man-made structures (e.g. a boat) with the "killer whale" label. We note that images of killer whales in ImageNet often have man-made structures which look similar to the boat (see Appendix C for examples).

Finally, at each step of our masking experiments, we recompute the saliency values of the

originally chosen 10 filters (i.e. the filters that caused erroneous behavior on the reference image). From Figure 4.7 (c), we observe that as we mask out the input features according to our input-saliency, the saliency values of those filters decrease gradually and reach an 80% drop, confirming that highlighted regions indeed drive the high saliency of the chosen filters.

Dataset-level masking experiment. We performed the experiment of masking the non-foreground image areas on the dataset level and observed similar trends to the case study. Specifically, we trained a ResNet-50 on half of the ImageNet train set, and then used the incorrectly classified images from the other half of the train set with target object bounding boxes for the experiment (we used annotations from the ImageNet website[3]). We computed our input saliency for every image and masked out the most salient regions of those images while preserving the target object. We observed that as a result of masking, the softmax output corresponding to the correct class increases ($p < 0.05$) on average and decreases for the incorrect class ($p < 0.05$), while masking the same number of randomly selected pixels (similarly outside of the target object bounding box) does not produce a statistically significant change in either correct or incorrect class confidence scores (p = 0.854 and p = 0.695, respectively).

Figure 4.8 showcases input space visualizations with different illustrative examples of neural network mistakes. For a thorough discussion of mistake categories we find using our method, we refer to Appendix C.1.7.

4.5 Discussion

Numerous applications demand that practitioners be able to understand the decisions their models make, especially when those decisions are incorrect. Existing methods for explainability focus on locating the input regions to which the network's output is sensitive or on associating network components with specific roles. In contrast, we develop a framework for finding the exact filters which are responsible for faulty predictions and studying the interac-

[3]https://image-net.org/download-bboxes.php

tions between these filters and images. This direction yields an interpretable understanding of model behaviors.

Chapter 5

TRANSFER LEARNING WITH DEEP TABULAR MODELS

This chapter is based on joint work with Valeriia Cherepanova, Avi Schwarzschild, Arpit Bansal, Bayan Bruss, Micah Goldblum, Tom Goldstein and Andrew Gordon Wilson.

5.1 Introduction

Tabular data is ubiquitous throughout diverse real-world applications, spanning medical diagnosis [96], housing price prediction [3], loan approval [13], and robotics [227], yet practitioners still rely heavily on classical machine learning systems. Recently, neural network architectures and training routines for tabular data have advanced significantly. Leading methods in tabular deep learning [71, 70, 190, 106] now perform on par with the traditionally dominant gradient boosted decision trees (GBDT) [65, 162, 32, 99]. On top of their competitive performance, neural networks, which are end-to-end differentiable and extract complex data representations, possess numerous capabilities which decision trees lack; one especially useful capability is *transfer learning*, in which a representation learned on *pre-training* data is reused or fine-tuned on one or more *downstream* tasks.

Transfer learning plays a central role in industrial computer vision and natural language processing pipelines, where models learn generic features that are useful across many tasks. For example, feature extractors pre-trained on the ImageNet dataset can enhance object detectors [166], and large transformer models trained on vast text corpora develop conceptual understandings which can be readily fine-tuned for question answering or language inference [47]. One might wonder if deep neural networks for tabular data, which are typically shallow and whose hierarchical feature extraction is unexplored, can also build representations that are transferable beyond their pre-training tasks. In fact, a recent survey paper on deep

learning with tabular data suggested that efficient knowledge transfer in tabular data is an open research question [21]. In this work, we show that deep tabular models with transfer learning definitively outperform their classical counterparts when auxiliary upstream pre-training data is available and the amount of downstream data is limited. Importantly, we find representation learning with tabular neural networks to be more powerful than gradient boosted decision trees with stacking – a strong baseline leveraging knowledge transfer from the upstream data with classical methods.

Some of the most common real-world scenarios with limited data are medical applications. Accumulating large amounts of patient data with labels is often very difficult, especially for rare conditions or hospital-specific tasks. However, large related datasets, e.g. for more common diagnoses, may be available in such cases. We note that while computer vision medical applications are common [90, 175, 30, 208], many medical datasets are fundamentally tabular [69, 95, 96, 113]. Motivated by this scenario, we choose a realistic medical diagnosis test bed for our experiments both for its practical value and transfer learning suitability. We first design a suite of benchmark transfer learning tasks using the MetaMIMIC repository [74, 231] and use this collection of tasks to compare transfer learning with prominent tabular models and GBDT methods at different levels of downstream data availability. We explore several transfer learning setups and lend suggestions to practitioners who may adopt tabular transfer learning. Additionally, we compare supervised pre-training and self-supervised pre-training strategies and find that supervised pre-training leads to more transferable features in the tabular domain, contrary to findings in vision where a mature progression of self-supervised methods exhibit strong performance [81].

Finally, we propose a pseudo-feature method which enables transfer learning when upstream and downstream feature sets differ. As tabular data is highly heterogeneous, the problem of downstream tasks whose formats and features differ from those of upstream data is common and has been reported to complicate knowledge transfer [119]. Nonetheless, if our upstream data is missing columns present in downstream data, we still want to leverage pre-training. Our approach uses transfer learning in stages. In the case that upstream data

is missing a column, we first pre-train a model on the upstream data without that feature. We then fine-tune the pre-trained model on downstream data to predict values in the column absent from the upstream data. Finally, after assigning pseudo-values of this feature to the upstream samples, we re-do the pre-training and transfer the feature extractor to the downstream task. This approach offers appreciable performance boosts over discarding the missing features and often performs comparably to models pre-trained with the ground truth feature values.

Our contributions are summarized as follows:

1. We find that recent deep tabular models combined with transfer learning have a decisive advantage over strong GBDT baselines, even those that also leverage upstream data.

2. We compare supervised and self-supervised pre-training strategies and find that the supervised pre-training leads to more transferable features in the tabular domain.

3. We propose a pseudo-feature method for aligning the upstream and downstream feature sets in heterogeneous data, addressing a common obstacle in the tabular domain.

4. We provide advice for practitioners on architectures, hyperparameter tuning, and transfer learning setups for tabular transfer learning.

5.2 Related Work

Deep learning for tabular data. The field of machine learning for tabular data has traditionally been dominated by gradient-boosted decision trees [65, 32, 99, 162]. These models are used for practical applications across domains ranging from finance to medicine and are consistently recommended as the approach of choice for modeling tabular data [185].

An extensive line of work on tabular deep learning aims to challenge the dominance of GBDT models. Numerous tabular neural architectures have been introduced, based on the ideas of creating differentiable learner ensembles [160, 79, 234, 105, 14], incorporating attention mechanisms and transformer architectures [190, 71, 12, 88, 191, 106], as well as a

variety of other approaches [220, 221, 20, 103, 60, 177]. However, recent systematic benchmarking of deep tabular models [71, 185] shows that while these models are competitive with GBDT on some tasks, there is still no universal best method. gorishniy2021revisiting show that transformer-based models are the strongest alternative to GBDT and that ResNet and MLP models coupled with a strong hyperparameter tuning routine [5] offer competitive baselines. Similarly, kadra2021regularization find that carefully regularized MLPs are competitive. In a follow-up work, gorishniy2022embeddings show that transformer architectures equipped with advanced embedding schemes for numerical features bridge the performance gap between deep tabular models and GBDT.

Transfer learning. Transfer learning [150, 225, 251] has been incredibly successful in domains of computer vision and natural language processing (NLP). Large fine-tuned models excel on a variety of image classification [48, 42] and NLP benchmarks [47, 85]. ImageNet [44] pre-trained feature extractors are incorporated into the complex pipelines of successful object detection and semantic segmentation models [29, 166, 165, 164]. Transfer learning is also particularly helpful in applications with limited data availability such as medical image classification [9, 83, 31, 10].

In the tabular data domain, a recent review paper [21] finds that transfer learning is underexplored and that the question of how to perform knowledge transfer and leverage upstream data remains open. In our work, we seek to answer these questions through a systematic study of transfer learning with recent successful deep tabular models.

Multiple works mention that transfer learning in the tabular domain is challenging due to the highly heterogeneous nature of tabular data [91, 119]. Several papers focus on converting tabular data to images instead [182, 250, 196] and leveraging transfer learning with vision models [196]. Other studies explore designing CNN-like inductive biases for tabular models [94], transferring XGBoost hyperparameters [231, 74], and transferring whole models [57, 6, 122] in the limited setting of shared label and feature space between the upstream and downstream tasks. Stacking could also be seen as a form of leveraging upstream knowledge in classical methods [229, 204].

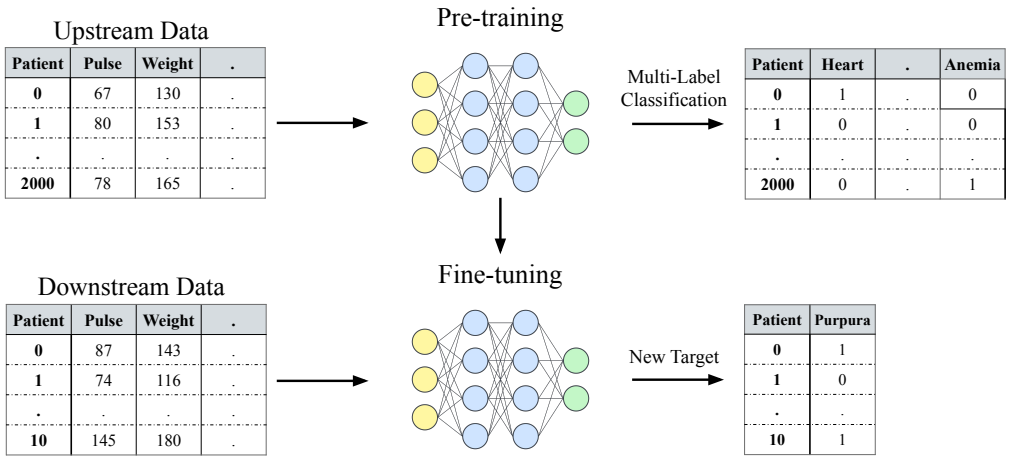


Figure 5.1: **Tabular transfer learning pipeline with MetaMIMIC.** We pre-train deep tabular neural networks on abundant upstream patient data with 11 diagnosis targets via multi-label classification. Then, we fine-tune the pre-trained models on limited downstream data with similar features to predict the new target diagnosis.

Self-supervised learning. Self-supervised learning (SSL) aimed at harnessing unlabelled data through learning its structure and invariances has accumulated a large body of works over the last few years. The prominent SSL methods, such as Masked Language Modeling (MLM) [47] from NLP and contrastive pre-training from computer vision [33] have revolutionized the fields making SSL the pre-training approach of choice [47, 112, 124, 120, 33, 81, 26, 16, 141]. In fact in vision, SSL pre-training has been shown to produce more transferable features than supervised pre-training on ImageNet [81].

Recently, SSL has been adopted in the tabular domain [236, 237, 209, 190, 88]. Contrastive pre-training on auxilary unlabelled data [190] and MLM-like approaches [88] have been shown to provide gains over training from scratch for transformer tabular architectures in cases of limited labelled data.

5.3 Transfer Learning Setup in Tabular Domain

To study transfer learning in the tabular domain, we need to choose benchmark tasks and training pipelines. In this section, we detail both our upstream-downstream datasets as well as the tools we use to optimize transfer learning for tabular data.

5.3.1 MetaMIMIC Test Bed for Transfer Learning Experiments

MetaMIMIC repository. As medical diagnosis data often contains similar test results (i.e. features) across patients, and some diseases (i.e. tasks) are common while others are rare, this setting is a realistic use-case for our work. We thus construct transfer learning benchmarks using the MetaMIMIC repository [74, 231] which is based on the MIMIC-IV [95, 69] clinical database of anonymized patient data from the the Beth Israel Deaconess Medical Center ICU admissions. MetaMIMIC contains 12 binary prediction tasks corresponding to different diagnoses such as hypertensive diseases, ischematic heart disease, diabetes, alcohol dependence and others. It covers 34925 patients and 172 features, including one categorical feature (gender), related to the medical examination of patients hand-selected by the authors to have the smallest number of missing values [74, 231]. The 12 medical diagnosis targets are related tasks of varied similarity and make MetaMIMIC a perfect test bed for transfer learning experiments.

Upstream and downstream tasks. A medical practitioner may possess abundant annotated diagnosis data corresponding to a number of common diseases and want to harness this data to assist in diagnosing another disease, perhaps one which is rare or for which data is scarce. To simulate this scenario, we create transfer learning problems by splitting the MetaMIMIC data into upstream and downstream tasks. Specifically, we reserve 11 targets for the upstream pre-training tasks and one target for the downstream fine-tuning tasks, thus obtaining 12 upstream-downstream splits – one for each downstream diagnosis. Additionally, we limit the amount of downstream data to 4, 10, 20, 100, and 200 samples corresponding to several scenarios of data availability.

In total, we have 60 combinations of upstream and downstream datasets for our transfer learning experiments. We pre-train our models as multi-label classifiers on the upstream datasets with 11 targets and then transfer the feature extractors onto the downstream binary diagnosis tasks, Figure 5.1 presents a diagram illustrating the pipeline.

5.3.2 Tabular Models

We conduct transfer learning experiments with four tabular neural networks, and we compare them to two GBDT implementations.

For neural networks, we use transformer-based architectures found to be the most competitive with GBDT tabular approaches [71, 88, 70]. The specific implementations we consider include the recent FT-Transformer [71] and TabTransformer [88]. We do not include implementations with inter-sample attention [190, 106] in our experiments since these do not lend themselves to scenarios with extremely limited downstream data. In addition, we use MLP and ResNet tabular architectures as they are known to be consistent and competitive baselines [71].

For GBDT implementation, we use the popular Catboost [162] and XGBoost libraries [32].

5.3.3 Transfer Learning Setups and Baselines

In addition to a range of architectures, we consider several setups for transferring the upstream pre-trained neural feature extractors onto downstream tasks. Specifically, we use either a single linear layer or a two-layer MLP with 200 neurons in each layer for the classification head. We also either freeze the weights of the feature extractor or fine-tune the entire model end-to-end. To summarize, we implement four transfer learning setups for neural networks:

- Linear head atop a frozen feature extractor
- MLP head atop a frozen feature extractor

- End-to-end fine-tuned feature extractor with a linear head
- End-to-end fine-tuned feature extractor with an MLP head

We compare the above setups to the following baselines:

- Neural models trained from scratch on downstream data
- CatBoost and XGBoost with and without stacking

We use stacking for GBDT models to build a stronger baseline which leverages the upstream data. To implement stacking, we first train upstream GBDT models to predict the 11 upstream targets and then concatenate their predictions to the downstream data features when training downstream GBDT models.

5.3.4 Hyperparameter Tuning

The standard hyperparameter tuning procedure for deep learning is to randomly sample a validation set and use it to optimize the hyperparameters. In contrast, in our scenario we often have too little downstream data to afford a sizeable validation split. However, we can make use of the abundant upstream data and leverage the hyperparameter transfer which is known to work for GBDT on related tasks [231, 74].

We tune the hyperparameters of each model with the Optuna library [5] using Bayesian optimization. In particular, for GBDT models and neural baselines trained from scratch, we tune the hyperparameters on a single randomly chosen upstream target with the same number of training samples as available in the downstream task, since hyperparameters depend strongly on the sample size. The optimal hyperparameters are chosen based on the upstream validation set, where validation data is plentiful. We find this tuning strategy to be especially effective for GBDT and provide comparison with default hyperparameters in Appendix D.3. The benefits of this hyperparameter tuning approach are less pronounced for deep baselines.

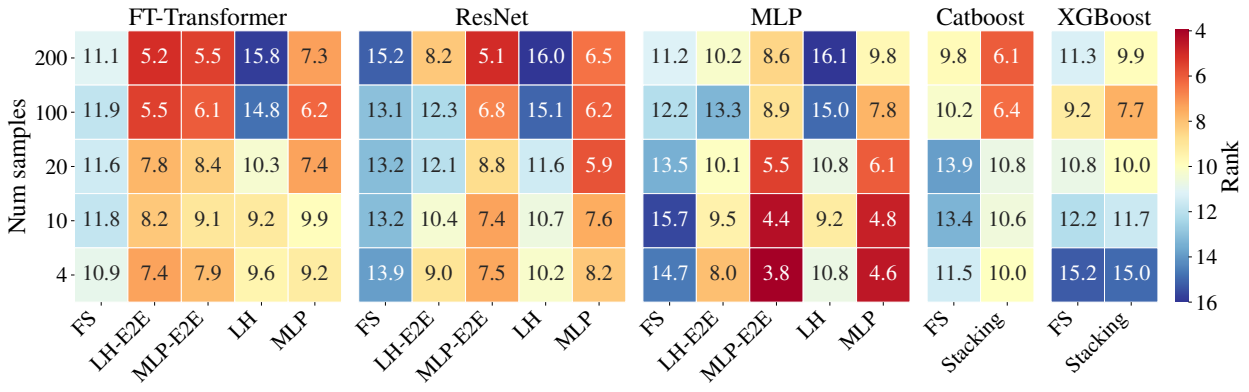


Figure 5.2: **Average model ranks across all downstream tasks.** Deep tabular models and GBDT performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks. FT-Transformer fine-tuned end-to-end outperforms all GBDT models, including GBDT with stacking, at all data levels. MLP is highly competitive in low data regimes.

For deep models with transfer learning, we tune the hyperparameters on the full upstream data using the available large upstream validation set with the goal to obtain the best performing feature extractor for the pre-training multi-target task. We then fine-tune this feature extractor with a small learning rate on the downstream data. As this strategy offers considerable performance gains over default hyperparameters, we highlight the importance of tuning the feature extractor and present the comparison with default hyperparameters in Appendix D.3 as well as the details on hyperparameter search spaces for each model.

5.4 Results for Transfer Learning with Deep Tabular Models

In this section, we compare transfer learned deep tabular models with GBDT methods at varying levels of downstream data availability. We note that here we present the aggregated results in the form of the average rank of the models across all of the twelve downstream

tasks at each data level. We choose this rank aggregation metric since it does not allow a small number of high-variance tasks to dominate comparisons, unlike, for example, average accuracy. Ranks are computed taking into account statistical significance of the performance differences between the models. We further report the detailed results for all of the models on all datasets and results for TabTransformer in Appendix D.4.

Figure 5.2 presents average model ranks on the downstream tasks as a heatmap where the warmer colors represent the better overall rank. The performance of every model is shown on the dedicated panel of the heatmap with the results for different transfer learning setups presented in columns. First, noting the color pattern in the Catboost and XGBoost columns, we observe that deep tabular models pre-trained on the upstream data outperform GBDT at all data levels and especially in the low data regime of 4-20 downstream samples. Interestingly, the most competitive model in the low data regime is MLP, the simplest deep tabular model, achieving the best results in the setup of end-to-end fine tuning with an MLP head (column MLP-E2E in the MLP panel of Figure 5.2). In the higher data regimes, FT-Transformer and ResNet are more competitive with the end-to-end fine-tuned FT-Transformer consistently outperforming GBDT in both linear and MLP head settings (LH-E2E and MLP-E2E columns in the FT-Transformer panel).

We emphasize that knowledge transfer with stacking, while providing strong boosts compared to from-scratch GBDT training (see Stacking and FS columns of GBDT), still decisively falls behind the deep tabular models with transfer learning, suggesting that representation learning for tabular data is significantly more powerful and allows neural networks to transfer richer information than simple predictions learned on the upstream tasks.

We summarize the main practical takeaways of Figure 5.2 below:

- Simpler models such as MLP with transfer learning are very competitive in extremely low data regimes. However, more complex architectures like FT-Transformer offer consistent performance gains over GBDT across all data levels.

- Representation learning with deep tabular models provides significant gains over strong

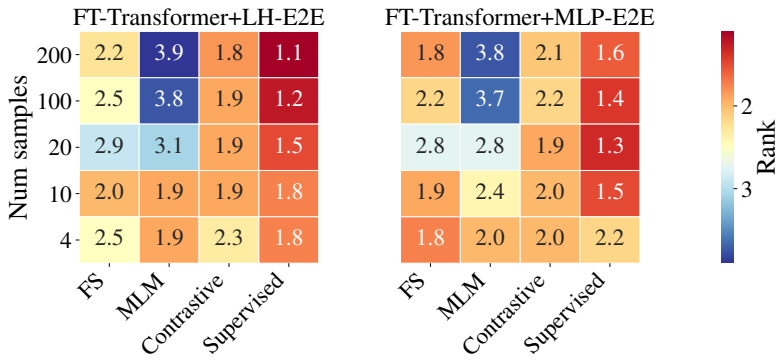


Figure 5.3: **Comparison of supervised and self-supervised pre-training strategies for FT-Transformer.** The left panel illustrates end-to-end fine-tuning with linear head and the right plot illustrates end-to-end fine-tuning with MLP head, the two most effective strategies for FT-Transformer. Within each panel, columns represent pre-training strategies and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. Contrastive pre-training outperforms from-scratch trained models, while MLM pre-training is not effective. Supervised pre-training outperforms all self-supervised pre-training strategies in our experiments.

GBDT baselines leveraging knowledge transfer from the upstream data through stacking. The gains are especially pronounced in low data regimes.

- Regarding transfer learning setups, we find that using an MLP head with a trainable or frozen feature extractor is effective for all deep tabular models. Additionally, a linear head with an end-to-end fine-tuned feature extractor is one of the best transfer-learning setups for FT-Transformer architecture.

5.5 Self-Supervised Pre-training

In domains where established SSL methods are increasingly dominant, such as computer vision, self-supervised learners are known to extract more transferable features than models trained on labelled data [81, 80]. In this section, we compare supervised pre-training with unsupervised pre-training and find that the opposite is true in the tabular domain. We use the Masked Language Model (MLM) pre-training recently adapted to the tabular data [88]

and the tabular version of contrastive learning [190]. Since both methods were proposed for tabular transformer architectures, we conduct the experiments with the FT-Transformer model.

5.5.1 Tabular MLM Pretraining

Masked Language Modeling (MLM) was first proposed for language models by devlin2019bert as a powerful unsupervised learning strategy. MLM involves training a model to predict tokens in text masked at random, thus facilitating learning good representations. In the tabular domain, instead of masking tokens, a random subset of features is masked for each sample, and the masked values are predicted in a multi-target classification manner [88]. In our experiments, we mask one randomly selected feature for each sample, asking the network to learn the structure of the data and form representations from $n-1$ features that are useful in producing the value in the n-th feature. For more detail, see Appendix D.2.

5.5.2 Contrastive Pre-Training

Contrastive pre-training uses data augmentations to generate positive pairs, or two different augmented views of a given example, and the loss function encourages a feature extractor to map positive pairs to similar features. Meanwhile, the network is also trained to map negative pairs, or augmented views of different base examples, far apart in feature space. We utilize the implementation of contrastive learning from somepalli2021saint. In particular, we generate positive pairs by applying two data augmentations: CutMix [241] in the input space and Mixup [244] in the embedding space. For more details, see Appendix D.2.

5.5.3 Comparing Supervised and Self-Supervised Pre-training

While self-supervised learning makes for transferable feature extractors in other domains, our experiments show that supervised pre-training is consistently better than the recent SSL pre-training methods we try that are designed for tabular data. In Figure 5.3, we

compare supervised pre-training with contrastive and MLM pre-training strategies and show that supervised pre-training always attains the best average rank. Contrastive pre-training produces better results than training from scratch on the downstream data when using a linear head, but it is still inferior to supervised pre-training. Tabular MLM pretraining also falls behind the supervised strategy and performs comparably to training from scratch in the lower data regimes but leads to a weaker downstream model in the higher data regimes.

5.6 Aligning Upstream and Downstream Feature Sets with Pseudo-Features

While so far we have worked with upstream and downstream tasks which shared a common feature space, in the real world, tabular data is highly heterogeneous and upstream data having a different set of features from downstream data is a realistic problem [119]. In our medical data scenario, downstream patient data may include additional lab tests not available for patients in the upstream data. In fact, the additional downstream feature may not even have a natural meaning for the upstream data, such as medical exams only available for downstream patients of biological sex different from the upstream patients. In this section, we propose a pseudo-feature method for aligning the upstream and downstream data and show that the data heterogeneity problem can be addressed more effectively than by simply taking the intersection of the upstream and downstream feature sets for tabular transfer learning, which would throw away useful features.

Suppose our upstream data (X_u, Y_u) is missing a feature x_{new} present in the downstream data (X_d, Y_d). We then use transfer learning in stages. As the diagram on the left panel of Figure 5.4 shows:

1. Pre-train feature extractor $f : X_u \to Y_u$ on upstream data (X_u, Y_u) without feature x_{new}.

2. Fine-tune the feature extractor f on the downstream samples X_d to predict x_{new} as the target and obtain a model $\hat{f} : X_d \setminus \{x_{\text{new}}\} \to x_{\text{new}}$.

3. Use the model $\hat{f}$ to assign pseudo-values $\hat{x}_{\text{new}}$ of the missing feature to the upstream samples: $\hat{x}_{\text{new}} = \hat{f}(X_u)$ thus obtaining augmented upstream data $(X_u \cup \{\hat{x}_{\text{new}}\}, Y_u)$.

4. Finally, we can leverage the augmented upstream data $(X_u \cup \{\hat{x}_{\text{new}}\}, Y_u)$ to pre-train a feature extractor which we will fine-tune on the original downstream task (X_d, Y_d) using all of the available downstream features.

Similarly, in scenarios with a missing feature in downstream data we can directly train a feature predictor on the upstream data and obtain pseudo-values for the downstream data.

This approach offers appreciable performance boosts over discarding the missing features and often performs comparably to models pre-trained with the ground truth feature values as shown in the right panel of Figure 5.4. The top heatmap represents the experiment where downstream data has an additional feature missing from the upstream data. The bottom heatmap represents the opposite scenario of the upstream data having an additional feature not available in the downstream data. To ensure that the features we experiment with are meaningful and contain useful information, for each task we chose important features according to GBDT feature importances. We observe that in both experiments using the pseudo feature is better than doing transfer learning with the missing feature. Additionally, we observe that in a few cases the pseudo-feature approach performs comparably to using the original ground truth feature (10-100 samples on the top heatmap and 20 samples on the bottom heatmap). Interestingly, the pseudo-feature method is more beneficial when upstream features are missing, which suggests that having ground-truth values for the additional feature in the downstream data is more important.

5.7 Discussion

In this paper, we demonstrate that deep tabular models with transfer learning definitively outperform strong GBDT baselines with stacking in a realistic scenario where the target downstream data is limited and auxillary upstream pre-training data is available. We

highlight that representation learning with neural networks enables more effective knowledge transfer than leveraging upstream task predictions through stacking. Additionally, we present a pseudo-feature method to enable effective transfer learning in challenging cases where the upstream and downstream feature sets differ. We provide suggestions regarding architectures, hyperparameter tuning, and setups for tabular transfer learning and hope that this work serves as a guide for practitioners.

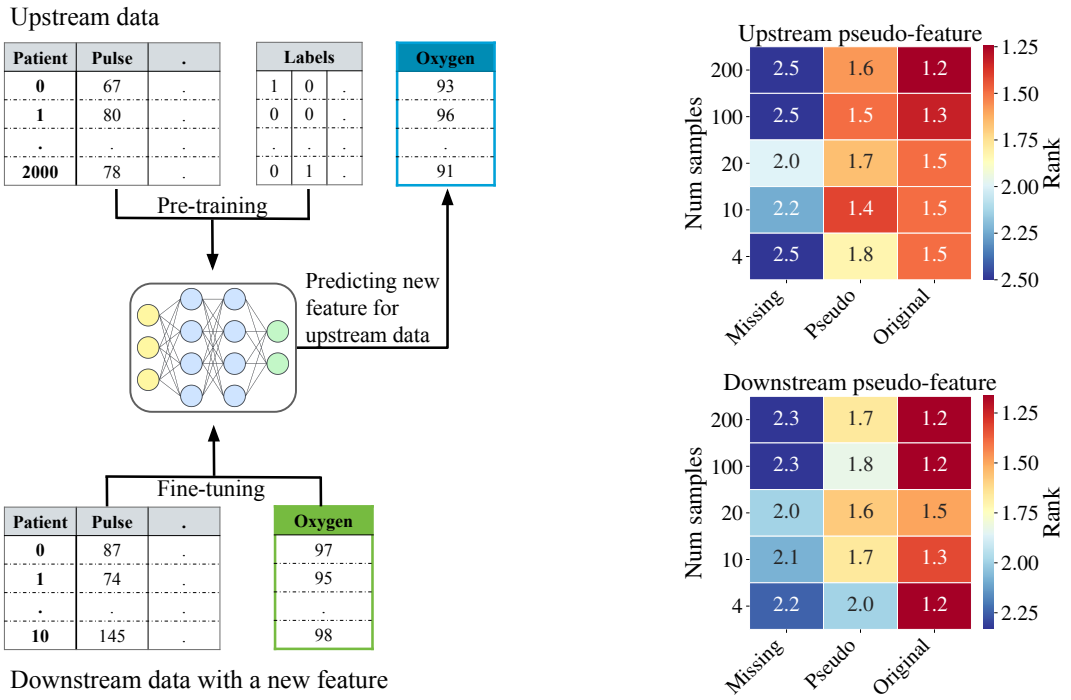


Figure 5.4: **Pseudo-Feature method for aligning upstream and downstream feature sets.** Left: Diagram illustrating strategy for handling mismatches between the upstream and downstream features. When upstream data is missing a feature present in downstream data, it is predicted with a model pre-trained on upstream data and fine-tuned to predict the new feature on the downstream data. Right top: Scenario with missing feature in the upstream data. Comparison of ranks of FT-Transformer model trained on data with missing feature, with pseudo-feature and with the original feature. Right bottom: Scenario with missing feature in the downstream data. Comparison of ranks of FT-Transformer model trained and fine-tuned on data with missing feature, fine-tuned with pseudo and with original feature. In both scenarios, using the pseudo-feature is better than training without the feature but worse than worse than the original ground truth feature values.

Chapter 6

IMPROVING PATIENT-SPECIFIC QUALITY ASSURANCE: RADIOTHERAPY PLAN FAILURE PREDICTION WITH DEEP TABULAR MODELS

This chapter is based on joint work with Minsun Kim and Aleksandr Aravkin.

6.1 Introduction

Intensity-modulated radiation therapy (IMRT) [149] achieves a dose distribution that is highly conformal to the target while minimizing the dose to normal tissue by modulating beam intensities with radiation fields. The modulation is performed using multi-leaf collimators (MLC) located within the gantry of a linear accelerator by varying the speed and position of each leaf and gantry. A complex intensity map within each radiation field, called fluence map, is obtained as the result. The target dose distribution is found using optimization algorithms implemented in a treatment planning system (TPS) [222, 35] and the leaf positions to deliver this dose are optimized using leaf sequencing algorithms [240, 126, 27, 183, 73, 50, 78].

IMRT is a complex multi-step process with a number of possible sources of noise ranging from computational approximations in the underlying algorithms to physical effects in the linear accelerator. Therefore, an extensive quality assurance (QA) process is required to prevent any unintended error from reaching the patient and affecting the patient's clinical outcome. It is current practice in many clinics to perform a patient-specific QA (PSQA) for each patient's radiation treatment plan [127] to ensure that the linear accelerator delivers the correct dose distributions as designed and shown by TPS. Another complex treatment planning technique which requires careful evaluation of the resulting dose distribution is volumetric-modulated arc therapy (VMAT) [202]. Both IMRT and VMAT include PSQA as

a standard step in the treatment process[56, 131].

One of the prevalent ways to perform PSQA is using a 2D or 3D phantom with an embedded array of detectors to measure the dose delivered using the patient's treatment beams. Then the computed dose distribution in the TPS is compared with the measured dose distribution, and a gamma analysis is performed to quantify the agreement between the two [130, 129]. Sometimes, PSQA fails due to a poor agreement between the computed and measured dose distributions requiring a replanning process and another PSQA, which is often done outside clinic hours. Therefore, PSQA failure can cause increased workload and stress for hospital staff members, delay in patient treatment or compromise patient safety if the work has to be rushed to preserve the patient's original treatment schedule.

To mitigate those issues and improve patient safety, a number of studies explore PSQA failure prediction and successfully demonstrate the feasibility of machine learning approaches in these efforts.

6.1.1 Related work

Machine learning for QA failure prediction. An extensive line of research focuses on developing non-learned treatment plan complexity metrics and investigating their correlation with PSQA failure [239, 41, 153, 40, 137, 152, 11]. A large number of papers further extend these approaches by developing deep learning models and classical machine learning models to predict the PSQA failure based on a vast array of the plan complexity metrics as well as other heuristic features [121, 219, 84, 233, 125, 97, 108, 111]. Other studies use MLC texture analysis and boosting algorithms for predicting gamma evaluation results [203]. Several works use target metrics alternative to gamma passing rates, such as dose difference [101, 89]. Other works leverage convolutional neural networks to predict the PSQA failure directly from fluence maps [205] or dose distributions [138, 206] obtained from TPS.

While many existing studies leverage heuristic feature engineering such as plan complexity metrics [121, 219, 84, 233, 125, 97, 108, 111], in our work, we develop tabular machine learning models based directly on MLC leaf positions to predict VMAT PSQA failure. We evaluate

the leading tabular models and demonstrate that reliable PSQA failure predictors can be successfully developed based solely on MLC leaf positions. One of our models is a tabular transformer neural network which is end-to-end differentiable and provides a differentiable map from MLC leaf positions to the probability of PSQA plan failure. As the MLC leaf positions are the output of leaf sequencing optimization algorithms [240, 126, 27, 183, 73, 50, 78], our model could be directly leveraged as a differentiable regularizer to improve them by encouraging to produce deliverable treatment plans, this is especially useful for the leaf sequencing algorithms which employ gradient-based optimization[27, 78] some of which are implemented in commercial treatment planning systems[78].

Tabular machine learning. Gradient boosted decision trees (GBDT) [65, 32, 99, 162] are the traditionally dominant machine learning approaches for tabular data. These models are commonly used in practice and are widely deployed in industry in various domains [185].

To challenge the prevalence of GBDT, numerous models have been proposed, based on using differentiable ensembles [160, 79, 234, 105, 14], leveraging attention-based transformer neural networks [190, 71, 12, 88, 191, 106], as well as other approaches [220, 221, 20, 103, 60, 177]. Recent work on systematic evaluation of deep tabular models [71, 185] shows that there is still no universally best model capable of consistently outperforming GBDT. Transformer-based models have been shown to be the strongest competitor of GBDT[71, 70, 190, 106], especially when coupled with a powerful hyperparameter tuning toolkit [5, 71].

In our work, we leverage the leading tabular machine learning models – gradient boosted decision trees (in their CatBoost and XGBoost implementations [162, 32]) and an attention-based neural network FT-Transformer [71]. We note that CatBoost and FT-Transformer have never been used before in the context of PSQA failure prediciton.

Our proposed approach is distinguished from the previous efforts in that we predict PSQA failure directly from MLC leaf positions. Additionally, the FT-Transformer model we apply is end-to-end differentiable and uses no heuristic feature engineering. It could therefore be leveraged as a differentiable regularizer for improving leaf sequencing algorithms [240, 126, 27, 183, 73, 50, 78]. Regularizing the MLC leaf optimization with our model could

lead to robust sequencing algorithms capable of producing more deliverable treatment plans.

6.2 Materials and Methods

In this section, we describe the pipeline of our study including the description of data collection and processing as well as the models, evaluation metrics and hyperparameter tuning approaches we use.

6.2.1 Data Description

In our study we leverage historical data of PSQA gamma analysis.

We collected anonymized radiotherapy plans of patients treated with VMAT (3% dose difference at 3 mm distance) in our clinic between 2019-2021. We constructed a tabular dataset on beam level leveraging the RT-DICOM [113] files of the treatment plans to form the samples: for each beam in a treatment plan, we used the leaf and jaw positions of the MLC collimators at each gantry angle. For the labels we used the beam-level result of the gamma analysis performed as part of the standard PSQA process in our clinic. We aggregated the MLC positions by averaging every 10 neighboring MLC pairs and averaging the gantry angles at every 8 degrees. As the result, we obtain a tabular regression dataset with 360 purely numerical features and 1873 samples.

For our ultimate goal of PSQA failure prediction, we consider the same data in the classification context. We define the action threshold to be at 95 % as is common in clinical practice [28, 151, 140] and obtain binary classification labels by thresholding the gamma pass rates.

We reserve 65% of the samples for the training set, 15% for the validation set and 20% for the test set. To preprocess the data, we normalize the features and regression targets by subtracting their mean over the training set and dividing by their standard deviation over the training set.

6.2.2 Models

In this study we use the recent transformer-based tabular deep learning method FT-Transformer as well as the well-established gradient boosted tree models. These approaches have been shown to be the most performant on tabular data in recent benchmarking [71, 70]. To implement gradient boosted trees, we use the popular CatBoost [162] and XGBoost [32] packages. For FT-Transformer, we use the official implementation [71].

Additionally, we report the classification performance of the non-learned baseline approach based on the average MLC gap – a plan complexity metric used in the current clinical practice at UWMC.

6.2.3 Model Evaluation

We evaluate our models in the regression context of predicting the gamma passing rates as well as in the classification context of predicting the PSQA plan failures.

In the regression context we use mean absolute error and root mean squared error metrics as well as Pearson's and Spearman's correlation coefficients between the predictions and the ground truth gamma passing rate values.

In the classification context, we use ROC AUC to evaluate the model performance. We report the beam-level ROC AUC and patient-level or plan-level ROC AUC. The patient-level predictions and labels are obtained by converting the beam-level predictions and labels such that a plan is failed if at least one beam in the plan is failed.

6.2.4 Hyperparameter Tuning

For hyperparameter tuning we use the Optuna Bayesian optimization toolkit [5]. The hyperparameter search spaces for each model are reported in Appendix E.1. Additionally, to avoid overfitting, for each model we use early stopping with patience, i.e. we stop training the models if no improvement in the validation score is observed for 30 epochs with FT-Transformer or for 50 boosting rounds with CatBoost and XGBoost.

Table 6.1: **Regression results.** Rows correspond to models and columns correspond to regression metrics.

	MAE (%)	RMSE (%)	Pearson's r	Spearman's r
XGBoost	1.53	1.89	0.58	0.59
CatBoost	1.40	1.84	0.6	0.59
FT-Transformer	1.44	1.95	0.51	0.51

6.3 Results

In this section we present the results of our machine learning approaches. We investigate the model performance in both regression and classification contexts and compare the machine learning models against a non-learned baseline based on a simple plan complexity metric currently used in clinical practice at UWMC.

6.3.1 Regression Results

We first present the performance of our models in predicting the gamma passing rates in Table 6.1. While all models are competitive and achieve good results of for example mean absolute errors of the gamma rate predictions of $1.4\% - 1.53\%$, CatBoost appears to be the strongest regressor outperforming both XGBoost and FT-Transformer across all the metrics. We note that these MAE, RMSE, Pearson's r and Spearman's r values are consistent and are on the same order with the results of other studies in the literature [121, 219, 84, 111, 205] even though they are not directly comparable given the differences in the experimental setups due to the varying hospital equipment and PSQA processes.

6.3.2 Classification Results

The ultimate clinical utility of our models is predicting the PSQA patient treatment plan failures to reduce the patient treatment delays and the load on the hospital resources. This

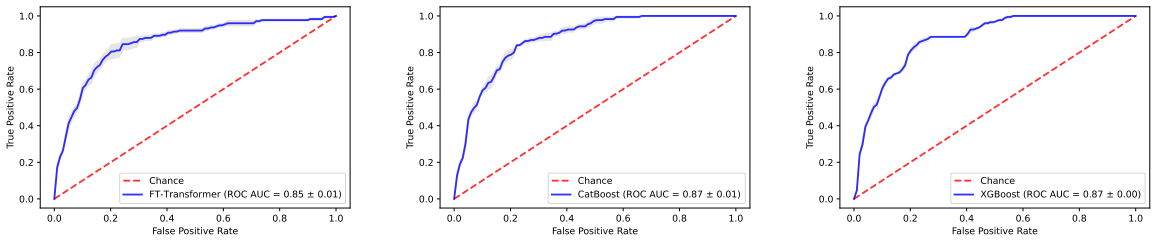
practical setup is best emulated by considering our models in the classification context. However, training the models using the regression labels instead of the classification labels directly allows us to leverage more fine-grained target information and avoid the challenges of severe class imbalance in the classification labels. Yet, the predictions of our regression models could be evaluated in the classification context and we present these results in Table 6.2. We highlight that Table 6.2 shows two types of ROC AUC metrics: beam-level and patient-level. As mentioned in section 6.2.3, the patient-level predictions are formed from the beam-level predictions by considering a patient plan to be failed if at least one of the beams in the plan is failed.

Table 6.2: **Classification results.** Rows correspond to models and columns correspond to classification metrics.

	Beam-level ROC AUC	Patient-level ROC AUC
Mean MLC Gap Baseline	0.71	0.72
XGBoost	0.87	0.87
CatBoost	0.86	0.87
FT-Transformer	0.82	0.85

As the main takeaways of Table 6.2, we observe that the patient-level ROC AUC classification performance of FT-Transformer is very close to that of CatBoost and XGBoost and that all of the machine learning approaches significantly outperform the Mean-MLC-Gap baseline currently used at UWMC.

While ROC AUC summarizes the classification performance for all of the prediction thresholds, a particular threshold has to be selected in practice. To investigate this, we further report the patient-level ROC curves for each of the machine learning models in Figure 6.1. Since missing a failed plan results in patient rescheduling, it is more costly than sending a successful plan for replanning. Therefore, in our clinical scenario it is beneficial to maximize the true positive rate of PSQA failure identification while keeping the false positive



(a) FT-Transformer (b) CatBoost (c) XGBoost

Figure 6.1: **Patient-level ROC Curves.** (a) FT-Transformer (b) CatBoost (c) XGBoost. The error bars represent the standard error across 5 seeds. The positive label corresponds to plan failure.

rate at a reasonable value. From the shape of the ROC curves in Figure 6.1, we observe that FT-Transformer and CatBoost serve this purpose well and allow to achieve 80% true positive rate while keeping the false positive rate under 20%, while XGBoost falls behind and achieves only approximately 70% true positive rate at 20% false positive rate.

6.4 Discussion

We leverage the dominant tabular machine learning models and show that accurate PSQA failure prediction is possible using just the MLC leaf position data. We evaluate our models in both regression and classification contexts and find that CatBoost produces the strongest regression model for gamma passing rate prediction and that FT-Transformer performs on par with CatBoost in the classification of PSQA plan failure.

Both CatBoost and FT-Transformer provide a substantial improvement over the complexity-metric-based baseline currently used in our clinic. While due to the lack of existing benchmark datasets and every institution using different linear accelerators, treatment planning systems and variations of the QA procedure, it is hard to compare models across studies directly, we note that our results are consistent with the literature. Our models achieve classification performance of 0.85-0.87 ROC AUC and are able to identify 80% of treatment

plans that would have failed the PSQA while sending for replanning only 20% of successful plans. Using these models in clinical practice will significantly reduce the need for patient rescheduling caused by QA failure as well as the workload for the radiation oncology staff members.

The FT-Transformer neural network model comes with an additional benefit of being end-to-end differentiable providing a differentiable map from MLC positions to the probability of PSQA failure. Therefore, this model could be leveraged as a differentiable regularizer to improve the gradient-based leaf sequencing optimization algorithms encouraging them to produce a deliverable treatment plan.

6.5 Conclusion

In this work we apply the leading tabular machine learning approaches to the problem of PSQA failure prediction based solely on MLC leaf positions, and obtain effective models which have both direct clinical practice impact as well as the potential to improve MLC leaf sequencing methods.

BIBLIOGRAPHY

[1] Reza Abbasi-Asl and Bin Yu. Interpreting convolutional neural networks through compression. *arXiv preprint arXiv:1711.02329*, 2017.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

[3] Bruno Afonso, Luckeciano Melo, Willian Oliveira, Samuel Sousa, and Lilian Berton. Housing prices prediction with a deep learning and random forest ensemble. In *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*, pages 389–400. SBC, 2019.

[4] 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.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[6] Samir Al-Stouhi and Chandan K Reddy. Adaptive boosting for transfer learning using dynamic updates. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 60–75. Springer, 2011.

[7] Hunt Allcott, Levi Boxell, Jacob Conway, Matthew Gentzkow, Michael Thaler, and David Yang. Polarization and public health: Partisan differences in social distancing during the coronavirus pandemic. *J Public Econ*, 191:104254, Nov 2020.

[8] Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 275–285, 2020.

[9] Laith Alzubaidi, Muthana Al-Amidie, Ahmed Al-Asadi, Amjad J Humaidi, Omran Al-Shamma, Mohammed A Fadhel, Jinglan Zhang, J Santamaría, and Ye Duan. Novel transfer learning approach for medical imaging with limited labeled data. *Cancers*, 13(7):1590, 2021.

[10] Laith Alzubaidi, Mohammed A Fadhel, Omran Al-Shamma, Jinglan Zhang, and Ye Duan. Deep learning models for classification of red blood cells in microscopy images to aid in sickle cell anemia diagnosis. *Electronics*, 9(3):427, 2020.

[11] Mikaël Antoine, Flavien Ralite, Charles Soustiel, Thomas Marsac, Paul Sargos, Audrey Cugny, and Jérôme Caron. Use of metrics to quantify imrt and vmat treatment plan complexity: A systematic review and perspectives. *Physica Medica*, 64:98–108, 2019.

[12] Sercan O Arık and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, volume 35, pages 6679–6687, 2021.

[13] Kumar Arun, Garg Ishan, and Kaur Sanmeet. Loan approval prediction based on machine learning approach. *IOSR J. Comput. Eng*, 18(3):18–21, 2016.

[14] Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, Khoa Doan, and Sathiya S Keerthi. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.

[15] Gene Balk. Nearly half of Seattle-area adults working from home because of COVID – here's who is and isn't hitting the road. *The Seattle Times*, September 28 2020. https://www.seattletimes.com/seattle-news/data/nearly-half-of-seattle-area-adults-working-from-home-because-of-pandemic/.

[16] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[17] Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Identifying and controlling important neurons in neural machine translation. *arXiv preprint arXiv:1811.01157*, 2018.

[18] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[19] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[20] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54, 2018.

[21] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

[22] R A Britten, L J Peters, and D Murray. Biological factors influencing the RBE of neutrons: implications for their past, present and future use in radiotherapy. *Radiation Research*, 156(2):125–131, 2001.

[23] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 11(5):393–401, 1997.

[24] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[25] Caroline O Buckee, Satchit Balsari, Jennifer Chan, Merce Crosas, Francesca Dominici, Urs Gasser, Yonatan H Grad, Bryan Grenfell, M Elizabeth Halloran, Moritz U G Kraemer, Marc Lipsitch, C Jessica E Metcalf, Lauren Ancel Meyers, T Alex Perkins, Mauricio Santillana, Samuel V Scarpino, Cecile Viboud, Amy Wesolowski, and Andrew Schroeder. Aggregated mobility data could help fight COVID-19. *Science*, 368(6487):145–146, Apr 10 2020.

[26] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[27] A Cassioli and J Unkelbach. Aperture shape optimization for imrt treatment planning. *Physics in Medicine & Biology*, 58(2):301, 2012.

[28] Gordon H Chan, Lee CL Chin, Ady Abdellatif, Jean-Pierre Bissonnette, Lesley Buckley, Daria Comsa, Dal Granville, Jenna King, Patrick L Rapley, and Aaron Vandermeer. Survey of patient-specific quality assurance practice for imrt and vmat. *Journal of Applied Clinical Medical Physics*, 22(7):155–164, 2021.

[29] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[30] Shupeng Chen, An Qin, Dingyi Zhou, and Di Yan. U-net-generated synthetic ct images for magnetic resonance imaging-only prostate intensity-modulated radiation therapy treatment planning. *Medical physics*, 45(12):5659–5665, 2018.

[31] Sihong Chen, Kai Ma, and Yefeng Zheng. Med3d: Transfer learning for 3d medical image analysis. *arXiv preprint arXiv:1904.00625*, 2019.

[32] 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.

[33] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[34] A Chi, H Chen, S Wen, H Yan, and Z Liao. Comparison of particle beam therapy and stereotactic body radiotherapy for early stage non-small cell lung cancer: A systematic review and hypothesis-generating meta-analysis. *Radiotherapy and Oncology*, 123(3):346–354, 2017.

[35] Byungchul Cho. Intensity-modulated radiation therapy: a review with a physics perspective. *Radiation oncology journal*, 36(1):1, 2018.

[36] D'vera Cohn. About a fifth of U.S. adults moved due to COVID-19 or know someone who did, 2020. https://www.pewresearch.org/fact-tank/2020/07/06/about-a-fifth-of-u-s-adults-moved-due-to-covid-19-or-know-someone-who-did/.

[37] Ronald R Coifman, Ioannis G Kevrekidis, Stéphane Lafon, Mauro Maggioni, and Boaz Nadler. Diffusion maps, reduction coordinates, and low dimensional representation of stochastic systems. *Multiscale Modeling & Simulation*, 7(2):842–864, 2008.

[38] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[39] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust region methods*, volume 1, page 19. Siam, 2000.

[40] SB Crowe, Tanya Kairn, John Kenny, RT Knight, Brendan Hill, Christian M Langton, and JV Trapp. Treatment plan complexity metrics for predicting imrt pre-treatment quality assurance results. *Australasian physical & engineering sciences in medicine*, 37(3):475–482, 2014.

[41] SB Crowe, Tanya Kairn, Nigel Middlebrook, Bess Sutherland, Brendan Hill, John Kenny, Christian M Langton, and JV Trapp. Examination of the properties of imrt and vmat beams and evaluation against pre-treatment quality assurance results. *Physics in Medicine & Biology*, 60(6):2587, 2015.

[42] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *arXiv preprint arXiv:2106.04803*, 2021.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[45] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[46] 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.

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[48] 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.

[49] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*, 2021.

[50] MA Earl, MKN Afghan, CX Yu, Z Jiang, and DM Shepard. Jaws-only imrt using direct aperture optimization. *Medical physics*, 34(1):307–314, 2007.

[51] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[52] SCM Ten Eikelder, D den Hertog, T Bortfeld, and Z Perko. Optimal combined proton-photon therapy schemes based on the standard bed model. *Phys Med Biol*, 64(6):065011, 2019.

[53] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[54] Esri. World topographic map, 2020. Downloaded on October 9, 2020.

[55] European Commission. *Recital 71 EU General Data Protection Regulation*. 2018.

[56] Gary A Ezzell, James M Galvin, Daniel Low, Jatinder R Palta, Isaac Rosen, Michael B Sharpe, Ping Xia, Ying Xiao, Lei Xing, and Cedric X Yu. Guidance document on delivery, treatment planning, and clinical implementation of imrt: report of the imrt subcommittee of the aapm radiation therapy committee. *Medical physics*, 30(8):2089–2115, 2003.

[57] Wenjing Fang, Chaochao Chen, Bowen Song, Li Wang, Jun Zhou, and Kenny Q Zhu. Adapted tree boosting for transfer learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 741–750. IEEE, 2019.

[58] Ian Fellows and using the JMapViewer library by Jan Peter Stotz. *OpenStreetMap: Access to Open Street Map Raster Images*, 2019. R package version 0.3.4.

[59] Nuno Fernandes. Economic effects of coronavirus outbreak (COVID-19) on the world economy, 2020. https://ssrn.com/abstract=3557504.

[60] James Fiedler. Simple modifications to improve tabular neural networks. *arXiv preprint arXiv:2108.03214*, 2021.

[61] A Fiorentino, R Laudicella, E Ciurlia, S Annunziata, V Lancellotta, P Mapelli, C Tuscano, F Caobelli, L Evangelista, L Marino, N Quartuccio, M Fiore, P Borghetti, A Chiaravalloti, M Ricci, I Desideri, P Alongi, AIRO Giovani Italian Association of Radiation Oncology-Young Members, and AIMN Italian Association of Nuclear Medicine-Young Members Working Group. Positron emission tomography with computed tomography imaging (pet/ct) for the radiotherapy planning definition of the biological target volume: Part 2. *Crit Rev Oncol Hematol*, 139:117–124, 2019.

[62] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.

[63] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[64] Chris Fraley and Adrian E Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[65] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[66] Richard Fry, Jeffrey S. Passel, and D'vera Cohn. A majority of young adults in the U.S. live with their parents for the first time since the Great Depression, 2020. https://www.pewresearch.org/fact-tank/2020/09/04/a-majority-of-young-adults-in-the-u-s-live-with-their-parents-for-the-first-time-since-the-great-depression/.

[67] H Gao. Hybrid proton-photon inverse optimization with uniformity-regularized proton and photon target dose. *Phys Med Biol*, 64(10):105003, 2019.

[68] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

[69] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[70] Yura Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *arXiv preprint arXiv:2203.05556*, 2022.

[71] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *arXiv preprint arXiv:2106.11959*, 2021.

[72] Kyra H Grantz, Hannah R Meredith, Derek A T Cummings, C Jessica E Metcalf, Bryan T Grenfell, John R Giles, Shruti Mehta, Sunil Solomon, Alain Labrique, Nishant Kishore, Caroline O Buckee, and Amy Wesolowski. The use of mobile phone data to inform analysis of COVID-19 pandemic epidemiology. *Nat Commun*, 11(1):4961, Sep 30 2020.

[73] Dal A Granville, Justin G Sutherland, Jason G Belec, and Daniel J La Russa. Predicting vmat patient-specific qa results using a support vector classifier trained on treatment plan characteristics and linac qc metrics. *Physics in Medicine & Biology*, 64(9):095017, 2019.

[74] Mateusz Grzyb, Zuzanna Trafas, Katarzyna Woźnica, and Przemysław Biecek. metamimic: analysis of hyperparameter transferability for tabular data using mimic-iv database, 2021.

[75] J. Guckenheimer and P. Holmes. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, volume 42 of *Applied Mathematical Sciences*. Springer, 1983.

[76] M F Haefner, V Verma, N Bougatf, T Mielke, E Tonndorf-Martini, L König, J M Rwigema, C B Simone, L Uhlmann, F Eichhorn, H Winter, H Grosch, T Haberer, K Herfarth, J Debus, and S Rieken. Dosimetric comparison of advanced radiotherapy approaches using photon techniques and particle therapy in the postoperative management of thymoma. *Acta Oncologica*, 2018.

[77] E J Hall and A J Giaccia. *Radiobiology for the Radiologist*. Lippincott Williams & Wilkins, Philadelphia, Pennsylvania, USA, 2006.

[78] Bjorn Hardemark, A Liander, H Rehbinder, and J Löf. Direct machine parameter optimization with raymachine in pinnacle. *Ray-Search White Paper*, 2003.

[79] Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

[80] 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.

[81] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[83] Michal Heker and Hayit Greenspan. Joint liver lesion segmentation and classification via transfer learning. *arXiv preprint arXiv:2004.12352*, 2020.

[84] Hideaki Hirashima, Tomohiro Ono, Mitsuhiro Nakamura, Yuki Miyabe, Nobutaka Mukumoto, Hiraku Iramina, and Takashi Mizowaki. Improvement of prediction and classification performance for gamma passing rate by using plan complexity and dosiomics features. *Radiotherapy and Oncology*, 153:250–257, 2020.

[85] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[86] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[87] Xiao Huang, Zhenlong Li, Junyu Lu, Sicheng Wang, Hanxue Wei, and Baixu Chen. Time-series clustering for home dwell time during COVID-19: what can we learn from it? *medRxiv*, 2020.

[88] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[89] Ying Huang, Yifei Pi, Kui Ma, Xiaojuan Miao, Sichao Fu, Hua Chen, Hao Wang, Hengle Gu, Yan Shao, Yanhua Duan, et al. Virtual patient-specific quality assurance of imrt using unet++: Classification, gamma passing rates prediction, and dose difference prediction. *Frontiers in Oncology*, page 2798, 2021.

[90] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 590–597, 2019.

[91] Vanshika Jain, Meghansh Goel, and Kshitiz Shah. Deep learning on small tabular dataset: Using transfer learning and image classification. In *International Conference on Artificial Intelligence and Speech Technology*, pages 555–568. Springer, 2021.

[92] Jonathan Jay, Jacob Bor, Elaine O. Nsoesie, Sarah K. Lipson, David K. Jones ADN Sandro Galea, and Julia Raifman. Neighbourhood income and physical distancing during the COVID-19 pandemic in the United States. *Nature Human Behaviour*. In press.

[93] Benjamin Jeffrey, Caroline E Walters, Kylie E C Ainslie, Oliver Eales, Constanze Ciavarella, Sangeeta Bhatia, Sarah Hayes, Marc Baguelin, Adhiratha Boonyasiri, Nicholas F Brazeau, Gina Cuomo-Dannenburg, Richard G FitzJohn, Katy Gaythorpe, William Green, Natsuko Imai, Thomas A Mellan, Swapnil Mishra, Pierre Nouvellet, H Juliette T Unwin, Robert Verity, Michaela Vollmer, Charles Whittaker, Neil M Ferguson, Christl A Donnelly, and Steven Riley. Anonymised and aggregated crowd level mobility data from mobile phones suggests that initial compliance with COVID-19 social distancing interventions was high and geographically consistent across the UK. *Wellcome Open Res*, 5:170, 2020.

[94] Leonid Joffe. Transfer learning for tabular data. 2021.

[95] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv, 2021.

[96] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[97] Tanya Kairn, SB Crowe, John Kenny, RT Knight, and JV Trapp. Predicting the likelihood of qa failure using treatment plan accuracy metrics. In *Journal of Physics: Conference Series*, volume 489, page 012051. IOP Publishing, 2014.

[98] Eunhee Kang, Junhong Min, and Jong Chul Ye. A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction. *Medical physics*, 44(10):e360–e375, 2017.

[99] 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. *Advances in neural information processing systems*, 30, 2017.

[100] M Kim, R D Stewart, and M H Phillips. A feasibility study: selection of a personalized radiotherapy fractionation schedule using spatiotemporal optimization. *Medical Physics*, 42:6671–6678, 2015.

[101] Yuto Kimura, Noriyuki Kadoya, Yohei Oku, Tomohiro Kajikawa, Seiji Tomori, and Keiichi Jingu. Error detection model developed using a multi-task convolutional neural network in patient-specific quality assurance for volumetric-modulated arc therapy. *Medical Physics*, 48(9):4769–4783, 2021.

[102] Nishant Kishore, Mathew V Kiang, Kenth Engo-Monsen, Navin Vembar, Andrew Schroeder, Satchit Balsari, and Caroline O Buckee. Measuring mobility to monitor travel and physical distancing interventions: a common framework for mobile phone data analysis. *Lancet Digit Health*, 2(11):e622–e628, Nov 2020.

[103] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[104] Andrey Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Inst. Ital. Attuari, Giorn.*, 4:83–91, 1933.

[105] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[106] Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Thomas Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[107] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[108] Terufumi Kusunoki, Shogo Hatanaka, Masatsugu Hariu, Yohsuke Kusano, Daisaku Yoshida, Hiroyuki Katoh, Munefumi Shimbo, and Takeo Takahashi. Evaluation of prediction and classification performances in different machine learning models for patient-specific quality assurance of head-and-neck vmat plans. *Medical physics*, 49(1):727–741, 2022.

[109] J. N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press, 2013.

[110] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. *arXiv preprint arXiv:2102.11600*, 2021.

[111] Dao Lam, Xizhe Zhang, Harold Li, Yang Deshan, Brayden Schott, Tianyu Zhao, Weixiong Zhang, Sasa Mutic, and Baozhou Sun. Predicting gamma passing rates for portal dosimetry-based imrt qa using machine learning. *Medical physics*, 46(10):4666–4675, 2019.

[112] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[113] Maria YY Law and Brent Liu. Dicom-rt and its utilization in radiation therapy. *Radiographics*, 29(3):655–667, 2009.

[114] R. Levin. Covid mobility and behavior. `https://github.com/InstituteforDiseaseModeling/covid-mobility-and-behavior`, 2020.

[115] Roman Levin, Aleksandr Y Aravkin, and Minsun Kim. A proof of principle: Multimodality radiotherapy optimization. *arXiv preprint arXiv:1911.05182*, 2019.

[116] Roman Levin, Dennis L Chao, Edward A Wenger, and Joshua L Proctor. Cell phone mobility data and manifold learning: Insights into population behavior during the covid-19 pandemic. *medRxiv*, 2020.

[117] Roman Levin, Dennis L Chao, Edward A Wenger, and Joshua L Proctor. Insights into population behavior during the covid-19 pandemic from cell phone mobility data and manifold learning. *Nature Computational Science*, 1(9):588–597, 2021.

[118] Roman Levin, Manli Shu, Eitan Borgnia, Furong Huang, Micah Goldblum, and Tom Goldstein. Where do models go wrong? parameter-space saliency maps for explainability. *arXiv preprint arXiv:2108.01335*, 2021.

[119] Eryk Lewinson. *Python for Finance Cookbook: Over 50 recipes for applying modern Python libraries to financial data analysis*. Packt Publishing Limited, 2020.

[120] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[121] Jiaqi Li, LE Wang, Xile Zhang, Lu Liu, Jun Li, Maria F Chan, Jing Sui, and Ruijie Yang. Machine learning for patient-specific quality assurance of vmat: prediction and classification accuracy. *International Journal of Radiation Oncology* Biology* Physics*, 105(4):893–902, 2019.

[122] Zhao Li, Donghu Ding, Xuanwu Liu, Peng Zhang, Youxi Wu, and Lingzhou Ma. Ttnet: Tabular transfer network for few-samples prediction. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 293–301, 2021.

[123] Congcong Liu and Huaming Wu. Channel pruning based on mean gradient for accelerating convolutional neural networks. *Signal Processing*, 156:84–91, 2019.

[124] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[125] Jéssica Caroline Lizar, Carolina Cariolatto Yaly, Alexandre Colello Bruno, Gustavo Arruda Viani, and Juliana Fernandes Pavoni. Patient-specific imrt qa verification using machine learning and gamma radiomics. *Physica Medica*, 82:100–108, 2021.

[126] Troy Long, Mingli Chen, Steve Jiang, and Weiguo Lu. Continuous leaf optimization for imrt leaf sequencing. *Medical Physics*, 43(10):5403–5411, 2016.

[127] Thomas LoSasso, Chen-Shou Chui, and C Clifton Ling. Comprehensive quality assurance for the delivery of intensity modulated radiotherapy with a multileaf collimator used in the dynamic mode. *Medical physics*, 28(11):2209–2219, 2001.

[128] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[129] Daniel A Low and James F Dempsey. Evaluation of the gamma dose distribution comparison method. *Medical physics*, 30(9):2455–2464, 2003.

[130] Daniel A Low, William B Harms, Sasa Mutic, and James A Purdy. A technique for the quantitative evaluation of dose distributions. *Medical physics*, 25(5):656–661, 1998.

[131] Daniel A Low, Jean M Moran, James F Dempsey, Lei Dong, and Mark Oldham. Dosimetry tools and techniques for imrt. *Medical physics*, 38(3):1313–1338, 2011.

[132] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. Mode: automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 175–186, 2018.

[133] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[134] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

[135] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[136] L. B. Marks, E. D. Yorke, A. Jackson, R. K. Ten Haken, L. S. Constine, A. Eisbruch, S. M. Bentzen, J. Nam, and J. O. Deasy. Use of normal tissue complication probability models in the clinic. *International Journal of Radiation Oncology Biology Physics*, 76(3):S10–S19, 2010.

[137] Laura Masi, Raffaela Doro, Virginia Favuzza, Samantha Cipressi, and Lorenzo Livi. Impact of plan parameters on the dosimetric accuracy of volumetric modulated arc therapy. *Medical physics*, 40(7):071718, 2013.

[138] Takaaki Matsuura, Daisuke Kawahara, Akito Saito, Hideharu Miura, Kiyoshi Yamada, Shuichi Ozawa, and Yasushi Nagata. Predictive gamma passing rate of 3d detector array-based volumetric modulated arc therapy quality assurance for prostate cancer via deep learning. 2022.

[139] MM Matuszak, R Kashani, M Green, D Owen, S Jolly, and M Mierzwa. Functional adaptation in radiation therapy. *Semin Radiat Oncol*, 29(3):236–244, 2019.

[140] Hunter Mehrens, Paige Taylor, David S Followill, and Stephen F Kry. Survey results of 3d-crt and imrt quality assurance practice. *Journal of applied clinical medical physics*, 21(7):70–76, 2020.

[141] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[142] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

[143] Amanda Moreland, Christine Herlihy, Michael A Tynan, Gregory Sunshine, Russell F McCord, Charity Hilton, Jason Poovey, Angela K Werner, Christopher D Jones, Erika B Fulmer, Adi V Gundlapalli, Heather Strosnider, Aaron Potvien, Macarena C Garcia, Sally Honeycutt, and Grant Baldwin. Timing of state and territorial COVID-19 stay-at-home orders and changes in population movement – United States, March 1–May 31, 2020. *MMWR Morb Mortal Wkly Rep*, 69(35):1198–1203, Sep 4 2020.

[144] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[145] Maria Nicola, Zaid Alsafi, Catrin Sohrabi, Ahmed Kerwan, Ahmed Al-Jabir, Christos Iosifidis, Maliha Agha, and Riaz Agha. The socio-economic implications of the coronavirus and COVID-19 pandemic: A review. *Int J Surg*, 78:185–193, Jun 2020.

[146] S Nourollahi, A Ghate, and M Kim. Optimal modality selection in external beam radiotherapy. *Mathematical Medicine and Biology*, page dqy013, 2018.

[147] S Nourollahi, A Ghate, and M Kim. Robust modality selection in external beam radiotherapy. In H Yang and R Qui, editors, *Advances in Service Science*. Springer, 2018.

[148] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[149] Jatinder R Palta, T Rockwell Mackie, and Rena Lee. Intensity-modulated radiation therapy state of the art. In *Proceedings of the Korean Society of Medical Physics Conference*, pages 4–4. Korean Society of Medical Physics, 2006.

[150] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[151] Yuxi Pan, Ruijie Yang, Shuming Zhang, Jiaqi Li, Jianrong Dai, Junjie Wang, and Jing Cai. National survey of patient specific imrt quality assurance in china. *Radiation Oncology*, 14(1):1–10, 2019.

[152] JM Park, HG Wu, JH Kim, JNK Carlson, and K Kim. The effect of mlc speed and acceleration on the plan delivery accuracy of vmat. *The British journal of radiology*, 88(1049):20140698, 2015.

[153] Jong Min Park, So-Yeon Park, Hyoungnyoun Kim, Jin Ho Kim, Joel Carlson, and Sung-Joon Ye. Modulation indices for volumetric modulated arc therapy. *Physics in Medicine & Biology*, 59(23):7315, 2014.

[154] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[155] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[156] A Peeters, J P.C. Grutters, M Pijls-Johannesma, S Reimoser, D De Ruysscher, J L Severens, M A Joore, and P Lambin. How costly is particle therapy? cost analysis of external beam radiotherapy with carbon-ions, protons and photons. *Radiotherapy and Oncology*, 95:45–53, 2010.

[157] F. Petroni, T. Rocktäschel, A. H. Miller, P. Lewis, A. Bakhtin, Y. Wu, and S. Riedel. Language models as knowledge bases? In *In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019*, 2019.

[158] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*, 2020.

[159] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[160] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

[161] Joshua L Proctor, Steven L Brunton, Bingni W Brunton, and JN Kutz. Exploiting sparsity and equation-free architectures in complex systems. *The European Physical Journal Special Topics*, 223(13):2665–2684, 2014.

[162] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[163] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[164] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[165] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[166] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[167] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[168] F Saberian, A Ghate, and M Kim. Optimal fractionation in radiotherapy with multiple normal tissues. *Mathematical Medicine and Biology*, 32(2), 2015.

[169] F Saberian, A Ghate, and M Kim. Spatiotemporally optimal fractionation in radiotherapy. *INFORMS J on Computing*, 29(3):422–438, 2017.

[170] Muhammad Sabih, Frank Hannig, and Juergen Teich. Utilizing explainable ai for quantization and pruning of deep neural networks. *arXiv preprint arXiv:2008.09072*, 2020.

[171] SafeGraph. Data analysis methodology for the SafeGraph stay-at-home index, 2020. https://docs.google.com/document/d/1k_9LGQn95P5gHsSeuBdzgtEWGGCmzXdcOkcphWi0Cas/e

[172] SafeGraph. June-2020 release notes, 2020. Accessed on August 1, 2020.

[173] SafeGraph. Safegraph common nighttime location algorithm, 2020. Accessed on October 1, 2020.

[174] SafeGraph. Social distancing metrics, 2020. Accessed on October 1, 2020.

[175] Beatriz Garcia Santa Cruz, Matías Nicolás Bossa, Jan Sölter, and Andreas Dominik Husch. Public covid-19 x-ray datasets and their impact on model bias–a systematic review of a significant problem. *Medical image analysis*, 74:102225, 2021.

[176] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[177] Bernhard Schäfl, Lukas Gruber, Angela Bitto-Nemling, and Sepp Hochreiter. Hopular: Modern hopfield networks for tabular data. 2021.

[178] Gideon Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[179] Luca Scrucca, Michael Fop, T Brendan Murphy, and Adrian E Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289–317, 2016.

[180] 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 *ICCV*, 2017.

[181] Venkatraman E. Seshan. *clinfun: Clinical Trial Design and Data Analysis Functions*, 2018. R package version 1.0.15.

[182] Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A Boroevich, and Tatsuhiko Tsunoda. Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific reports*, 9(1):1–7, 2019.

[183] David M Shepard, Matthew A Earl, X Allen Li, S Naqvi, and C Yu. Direct aperture optimization: a turnkey solution for step-and-shoot imrt. *Medical physics*, 29(6):1007–1018, 2002.

[184] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017.

[185] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[186] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

[187] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[188] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[189] Nikolai V Smirnov. Estimate of deviation between empirical distribution functions in two independent samples. *Bulletin Moscow University*, 2(2):3–16, 1939.

[190] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

[191] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170, 2019.

[192] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR*, 2015.

[193] Ryan Fox Squire. What about bias in the SafeGraph dataset?, 2019. https://www.safegraph.com/blog/what-about-bias-in-the-safegraph-dataset.

[194] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. *arXiv preprint arXiv:1905.00780*, 2019.

[195] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[196] Baohua Sun, Lin Yang, Wenhan Zhang, Michael Lin, Patrick Dong, Charles Young, and Jason Dong. Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[197] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *International Conference on Machine Learning*, pages 3299–3308. PMLR, 2017.

[198] Xu Sun, Zhiyuan Zhang, Xuancheng Ren, Ruixuan Luo, and Liangyou Li. Exploring the vulnerability of deep neural networks: A study of parameter corruption. *arXiv preprint arXiv:2006.05620*, 2020.

[199] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

[200] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[201] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[202] May Teoh, CH Clark, K Wood, S Whitaker, and A Nisbet. Volumetric modulated arc therapy: a review of current literature and clinical use in practice. *The British journal of radiology*, 84(1007):967–996, 2011.

[203] Sangutid Thongsawad, Somyot Srisatit, and Todsaporn Fuangrod. Predicting gamma evaluation results of patient-specific head and neck volumetric-modulated arc therapy quality assurance based on multileaf collimator patterns and fluence map features: A feasibility study. *Journal of Applied Clinical Medical Physics*, page e13622, 2022.

[204] Kai Ming Ting and Ian H Witten. Stacked generalization: when does it work? 1997.

[205] Seiji Tomori, Noriyuki Kadoya, Tomohiro Kajikawa, Yuto Kimura, Kakutarou Narazaki, Takahiro Ochi, and Keiichi Jingu. Systematic method for a deep learning-based prediction model for gamma evaluation in patient-specific quality assurance of volumetric modulated arc therapy. *Medical Physics*, 48(3):1003–1018, 2021.

[206] Seiji Tomori, Noriyuki Kadoya, Yoshiki Takayama, Tomohiro Kajikawa, Katsumi Shima, Kakutarou Narazaki, and Keiichi Jingu. A deep learning-based prediction model for gamma evaluation in patient-specific quality assurance. *Medical physics*, 45(9):4055–4065, 2018.

[207] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[208] Valérian Turbé, Carina Herbst, Thobeka Mngomezulu, Sepehr Meshkinfamfard, Nondumiso Dlamini, Thembani Mhlongo, Theresa Smit, Valeriia Cherepanova, Koki Shimada, Jobie Budd, et al. Deep learning of hiv field-based rapid tests. *Nature Medicine*, 27(7):1165–1170, 2021.

[209] Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[210] United States Congress Senate Committee on Banking and Housing and Urban Affairs. *Equal Credit Opportunity Act. [electronic resource].* S. Rpt. 94-685. Washington, 1976.

[211] J Unkelbach, M Bangert, B De Amorim, N Andratschke, and M Guckenberger. Optimization of combined proton-photon treatments. *Radiotheraoy Oncology*, 128(1):133–138, 2018.

[212] Jarkko Venna and Samuel Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In *International Conference on Artificial Neural Networks*, pages 485–491. Springer, 2001.

[213] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, CJ Carey, İlhan Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, page arXiv:1907.10121, Jul 2019.

[214] J Vogel, L Lin, L A Litzky, A T Berman, and C B Simone. Predicted rate of secondary malignancies following adjuvant proton versus photon radiation therapy for thymoma. *International Journal of Radiation Oncology Biology Physics*, 99(2):427–433, 2017.

[215] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

[216] Kyle Walker. *tidycensus: Load US Census Boundary and Attribute Data as 'tidyverse' and 'sf'-Ready Data Frames*, 2020. R package version 0.9.9.2.

[217] A Wambersie. RBE, reference RBE and clinical RBE: applications of these concepts in hadron therapy. *Strahlenther Onkol*, 175(Suppl 2):39–43, 1999.

[218] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *CVPR*, 2020.

[219] Le Wang, Jiaqi Li, Shuming Zhang, Xile Zhang, Qilin Zhang, Maria F Chan, Ruijie Yang, and Jing Sui. Multi-task autoencoder based classification-regression model for patient-specific vmat qa. *Physics in Medicine & Biology*, 65(23):235023, 2020.

[220] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7. 2017.

[221] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*, pages 1785–1797, 2021.

[222] Steve Webb. The physical basis of imrt and inverse planning. *The British journal of radiology*, 76(910):678–689, 2003.

[223] Joakim A Weill, Matthieu Stigler, Olivier Deschenes, and Michael R Springborn. Social distancing responses to COVID-19 emergency declarations strongly differentiated by income. *Proc Natl Acad Sci U S A*, 117(33):19658–19660, Aug 18 2020.

[224] Hannah K Weir, Trevor D Thompson, Sherri L Stewart, and Mary C White. Peer reviewed: Cancer incidence projections in the united states between 2015 and 2050. *Preventing chronic disease*, 18, 2021.

[225] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[226] David West. Neural network credit scoring models. *Computers & Operations Research*, 27(11-12):1131–1152, 2000.

[227] Johannes Wienke, Dennis Wigand, Norman Koster, and Sebastian Wrede. Model-based performance testing for robotics software components. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 25–32. IEEE, 2018.

[228] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[229] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[230] World Health Organization and others. Weekly operational update on COVID-19 9 October 2020, 2020. https://www.who.int/publications/m/item/weekly-update-on-covid-19—2-october-2020.

[231] Katarzyna Woźnica, Mateusz Grzyb, Zuzanna Trafas, and Przemysław Biecek. Consolidated learning–a domain-specific model-free optimization strategy with examples for xgboost and mimic-iv. *arXiv preprint arXiv:2201.11815*, 2022.

[232] Or Yair, Ronen Talmon, Ronald R Coifman, and Ioannis G Kevrekidis. Reconstruction of normal forms by learning informed observation geometries from data. *Proceedings of the National Academy of Sciences*, 114(38):E7865–E7874, 2017.

[233] Ruijie Yang, Xueying Yang, Le Wang, Dingjie Li, Yuexin Guo, Ying Li, Yumin Guan, Xiangyang Wu, Shouping Xu, Shuming Zhang, et al. Commissioning and clinical implementation of an autoencoder based classification-regression model for vmat patient-specific qa in a multi-institution scenario. *Radiotherapy and Oncology*, 161:230–240, 2021.

[234] Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

[235] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021.

[236] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.

[237] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

[238] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[239] Kelly C Younge, Don Roberts, Lindsay A Janes, Carlos Anderson, Jean M Moran, and Martha M Matuszak. Predicting deliverability of volumetric-modulated arc therapy (vmat) plans using aperture complexity analysis. *Journal of applied clinical medical physics*, 17(4):124–131, 2016.

[240] CX Yu, D Yan, MN Du, S Zhou, and LJ Verhey. Optimization of leaf positions when shaping a radiation field with a multileaf collimator. *Physics in Medicine & Biology*, 40(2):305, 1995.

[241] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[242] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[243] Hao Zhang and WK Chan. Apricot: A weight-adaptation approach to fixing deep learning models. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 376–387. IEEE, 2019.

[244] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[245] P. Zheng and A. Aravkin. Relax-and-split method for nonsmooth nonconvex problems. *arXiv preprint arXiv:1802.02654*, 2018.

[246] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2131–2145, 2018.

[247] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Comparing the interpretability of deep networks via network dissection. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 243–252. Springer, 2019.

[248] 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.

[249] J Zhou, B Yang, X Wang, and Z Jing. Comparison of the effectiveness of radiotherapy with photons and particles for chordoma after surgery: A meta-analysis. *World Neurosurgery*, 117:46–53, 2018.

[250] Yitan Zhu, Thomas Brettin, Fangfang Xia, Alexander Partin, Maulik Shukla, Hyunseung Yoo, Yvonne A Evrard, James H Doroshow, and Rick L Stevens. Converting tabular data into images for deep learning with convolutional neural networks. *Scientific reports*, 11(1):1–11, 2021.

[251] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

Appendix A

MULTI-MODALITY RADIOTHERAPY OPTIMIZATION

A.1 Proximal Operator Calculation

$$\text{prox}_{-\eta_0 f}(y) = \arg\min_x \left\{ -f(x) + \frac{1}{2\eta_0}\|x-y\|^2 \right\}. \tag{A.1}$$

Equation (2.17) gives us an update w_0^+ for w_0. We note that the proximal operator is defined for convex functions whereas our $-f(x)$ is concave. Therefore, the minimization problem in (A.1) may be unbounded. We will find the proximal operator by definition, that is, we will directly solve the minimization problem (A.1). Recall that B is diagonal.

$$\min_x -x^T Bx + \frac{1}{2\eta_0}\|x-y\|^2 \tag{A.2}$$

$$\Leftrightarrow \min_x -x^T Bx + \frac{1}{2\eta_0}x^T x - \frac{1}{\eta_0}\langle y, x\rangle \tag{A.3}$$

$$\Leftrightarrow \min_x -\sum_i B_{ii}x_i^2 + \frac{1}{2\eta_0}\sum_i x_i^2 - \frac{1}{\eta_0}\sum_i y_i x_i. \tag{A.4}$$

Finally, it is equivalent to

$$\min_x -\sum_i (B_{ii} - \frac{1}{2\eta_0})x_i^2 - \frac{1}{\eta_0}\sum_i y_i x_i. \tag{A.5}$$

Consider possible cases:

- If $\exists i : (B_{ii} - \frac{1}{2\eta_0}) > 0$, then take $x : x_j = 0\ \forall j \neq i, x_i \to \infty$, the objective is unbounded below since the quadratic term dominates over the linear term.

- If $\max_i(B_{ii} - \frac{1}{2\eta_0}) \leq 0$, then we have a convex objective and therefore we can set

derivative equal to zero to find the optimum. We have:

$$-2Bx + \frac{1}{2\eta_0}2(y-x)(-1) = 0 \Rightarrow (-2B + \frac{1}{\eta_0}I)x = \frac{1}{\eta_0}y.$$

Thus,

$$(\text{prox}_{-\eta_0 f}(y))_j = \begin{cases} \infty, & \max_i(B_{ii} - \frac{1}{2\eta_0}) > 0 \ \& \ j = \arg\max_i(B_{ii} - \frac{1}{2\eta_0}) \\ 0, & \max_i(B_{ii} - \frac{1}{2\eta_0}) > 0 \ \& \ j \neq \arg\max_i(B_{ii} - \frac{1}{2\eta_0}) \\ \frac{1}{\eta_0}y_j/(-2B_{jj} + \frac{1}{\eta_0}), & \max_i(B_{ii} - \frac{1}{2\eta_0}) \leq 0. \end{cases}$$

A.2 Projection Calculation

To find the projection, let us reformulate the problem as follows:

$$\min_w \|w - v\|^2$$
$$\text{s.t. } \gamma_1 w_1 + \gamma_2 w_2 + \cdots + \gamma_n w_n + D_1 w_1^2 + D_2 w_2^2 + \cdots + D_n w_n^2 \leq C_{\text{mean/max}}$$

where D_i stands for the i-th diagonal element of D. Now, let us complete the square:

$$\gamma_1 w_1 + \gamma_2 w_2 + \cdots + \gamma_n w_n + D_1 w_1^2 + D_2 w_2^2 + \cdots + D_n w_n^2 \leq C_{\text{mean/max}} \Leftrightarrow$$
$$(\frac{\gamma_1}{2\sqrt{D_1}})^2 + 2(\sqrt{D_1}w_1)(\frac{\gamma_1}{2\sqrt{D_1}}) + (\sqrt{D_1}w_1)^2 + \cdots \leq C_{\text{mean/max}} + (\frac{\gamma_1}{2\sqrt{D_1}})^2 + \cdots \Leftrightarrow$$
$$\textstyle\sum_i(\frac{\gamma_i}{2\sqrt{D_i}} + \sqrt{D_i}w_i)^2 \leq C_{\text{mean/max}} + \sum_i(\frac{\gamma_i}{2\sqrt{D_i}})^2.$$

Now, denote

$$K = C_{\text{mean/max}} + \textstyle\sum_i(\frac{\gamma_i}{2\sqrt{D_i}})^2, \ z_i = \frac{\gamma_i}{2\sqrt{D_i}} + \sqrt{D_i}w_i, \text{ then}$$
$$w_i = \frac{1}{\sqrt{D_i}}(z_i - \frac{\gamma_i}{2\sqrt{D_i}}).$$

We finally get:

$$\min_z \sum_i (\frac{1}{\sqrt{D_i}} z_i - \frac{\gamma_i}{2D_i} - v_i)^2$$
$$\text{s.t. } \|z\|^2 \leq K$$

Take the substitution $\hat{D} = \text{diag}(\frac{1}{\sqrt{D_1}}, \frac{1}{\sqrt{D_2}}, \ldots, \frac{1}{\sqrt{D_n}})$, $l_i = \frac{\gamma_i}{2D_i} + v_i$ and consider the following (using KKT):

$$\min_z \frac{1}{2}\|\hat{D}z - l\|_2^2 + \lambda(\|z\|^2 - K)$$

In fact, we have:

$$L(z, \lambda) = \|\hat{D}z - l\|_2^2 + \lambda(\|z\|^2 - K)$$
$$\text{KKT:}$$
$$\nabla L(z, \lambda) = \hat{D}^T(\hat{D}z - l) + 2\lambda z = 0$$
$$\lambda(z^T z - K) = 0$$
$$z^T z \leq K$$
$$\lambda \leq 0$$

From the first condition, we have

$$z^* = (\hat{D}^T\hat{D} + \lambda I)^{-1}\hat{D}^T l.$$

If $\lambda = 0$, then we check the $z^* = (\hat{D}^T\hat{D})^{-1}\hat{D}^T l$ to satisfy $z^{*T} z^* \leq K$. Otherwise, we look for the root λ^* of

$$\|(\hat{D}^T\hat{D} + \lambda I)^{-1}\hat{D}^T l\|_2^2 = K,$$

which we can find numerically by a root-finder routine. Then the projection is given by:

$$z^* = (\hat{D}^T \hat{D} + \lambda^* I)^{-1} \hat{D}^T l.$$

Going back to the original variables,

$$w_i = \frac{1}{\sqrt{D_i}}(z_i^* - \frac{\gamma_i}{2\sqrt{D_i}}).$$

Appendix B

CELL PHONE MOBILITY DATA AND MANIFOLD LEARNING: INSIGHTS INTO POPULATION BEHAVIOR DURING THE COVID-19 PANDEMIC

B.1 Linear dimensionality reduction and clustering

We used the singular value decomposition (SVD), closely related to principal component analysis, as a linear method to reduce the dimensionality of the mobility data matrix. Figure B.1 illustrates that more than 60 singular vectors (principal components) are required to reach 90% of the explained variance. This singular value distribution suggests a marginal benefit from linear dimensionality reduction. Moreover, applying a GMM to the data in the transformed space, we obtained highly uncertain GMM cluster assignments (see Table B.1 and Supplement B.4 for uncertainty quantification). Figure B.1 provides a 3D visualization of clustering in the first three singular vectors. This provides an illustration for why cluster assignment is uncertain and changes depending on the model initialization. Linear dimensionality reduction and GMM clustering is not an effective method for the mobility dataset motivating the use of manifold learning methods.

Table B.1: **Quartiles of the cluster assignment uncertainty based on a GMM with 11 clusters and the SVD with 8 modes.**

0%	25%	50%	75%	100%
3.38e-08	1.91e-01	3.53e-01	4.97e-01	7.46e-01

Table B.2: **Quartiles of uncertainty of the cluster assignment based on GMM with 5 clusters.**

Method	25%	50%	75%	100%
Laplacian Eigenmaps	0	0	1.11e-15	4.99e-01
Locally Linear Embedding	1.65e-06	6.16e-04	4.30e-02	6.18e-01
Isomap	4.79e-08	5.90e-05	4.32e-03	4.97e-01

B.2 Nonlinear Dimensionality Reduction Methods

We investigated a variety of nonlinear dimensionality reduction methods: t-SNE, locally linear embedding (standard, modified, and Hessian), Isomap, Spectral Embedding, local Tangent Space alignment, and multi-dimensional scaling. We found that Laplacian Eigenmaps [19], Locally Linear Embedding [167], and Isomaps [201] reduced the dimensionality of the data and identified a consistent tubular dense structure in the data (Figure B.2). Figure B.2 presents 3D representations of low-dimensional embeddings for each of the methods on the example of Washington State.

As described in Methods of the main manuscript, we chose the optimal embedding dimensionality based on the knee-point of the trustworthiness metric as a function of number of dimensions. All three methods qualitatively agree on the intrinsic dimensionality of the mobility data with the optimal dimensionality identified to be between 14 and 18 dimensions (see Figure B.3 for the trustworthiness metric plots for Washington State). All three methods capture the structure of the data well and produce cluster assignments with significantly lower associated uncertainty than linear dimensionality reduction (see Table B.2).

We selected Laplacian Eigenmaps as the primary methodology because it produced the least uncertain GMM cluster assignment and the clustering was robust to perturbations in the method's single hyperparameter, `n_neighbors` (Figure B.4). Trustworthiness was computed as a function of the Laplacian Eigenmap embedding dimensionality; a knee-point detection algorithm was then used to identify the optimal number of dimensions. Figure B.5 shows the optimal Laplacian eigenmaps dimensionality is 14 for every state. Note that California is

different in that there are two possible knee points: one that is consistent with other states at 14 and another at 44 dimensions. We compared clustering results for these two knee points (Figure B.6) and identify that the cluster assignments are similar. Therefore, in our main analysis we used 14D Laplacian eigenmap embedding for California.

B.3 Robustness of GMM fitting

The underlying objective function for the standard implementation of a Gaussian mixture model is not convex. We ensured that the GMM clustering produced consistent and robust results by re-initializing the fitting algorithm many times. As an example, Figure B.7 illustrates the GMM fit for six different initializations. We observe that the results are qualitatively similar with the main difference that the sparse region sometimes is identified as a separate cluster or divided into two clusters.

B.4 GMM Model and Uncertainty Quantification

Gaussian mixture model (Section 11.2 in [144]) is a latent variable model which assumes that the data has K sub-populations which follow Gaussian distributions with parameters μ_k, Σ_k respectively and that a latent discrete variable $z_i \in \{1, \ldots, K\}$ controls which sub-population a data point x_i comes from. If π corresponds to the probability mass function of z_i, then the GMM model has the form:

$$p(x_i|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k),$$

where θ stands for the set of all parameters of the model and $\mathcal{N}(x_i|\mu_k, \Sigma_k)$ is the probability density function of normal distribution. We note that GMM could be seen as a generalization of the famous K-means clustering algorithm [133].

The probabilistic formulation of the GMM model provides a natural way to quantify uncertainty of the cluster assignment. Using Bayes' Theorem, the posterior probability

$p(z_i = k|x_i, \theta)$ that point x_i belongs to cluster k can be computed as follows:

$$p(z_i = k|x_i, \theta) = \frac{p(z_i = k|\theta)p(x_i|z_i = k, \theta)}{\sum_{k'=1}^{K} p(z_i = k'|\theta)p(x_i|z_i = k', \theta)} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(x_i|\mu'_k, \Sigma'_k)}.$$

Then, the amount of uncertainty ϵ_i in the cluster assignment of point x_i could be computed as

$$\epsilon_i = 1 - \max_k p(z_i = k|x_i, \theta).$$

We note that the above formula assumes that the cluster assignment is computed as

$$z_i^* = \arg\max_k p(z_i = k|x_i, \theta).$$

B.5 GMM Model Selection

We used Bayesian Information Criterion (BIC) to identify the optimal number of GMM components [178, 64, 179]. BIC is based on a penalized form of the log-likelihood. As the likelihood increases with the addition of more components, a penalty term for the number of estimated parameters is subtracted from the log-likelihood [179]. To find the optimal number of GMM components, we applied knee-point detection to the BIC curve (Figure B.8). Note that in the `mclust` implementation, higher BIC values correspond to better models. The optimal number of clusters turned out to be 4 for Washington, Texas, and California and 5 for Georgia. We decided to use 5 clusters for every state in the main manuscript for consistency across the states and to leverage optimal results for Georgia.

B.6 Altering the Number of Clusters and Continuous Colormap

While the optimal number of clusters for Washington, Texas, and California was 4 (based on knee-point detection in BIC, see §B.5), we chose 5 as the number of clusters for every state in our main analysis. Allowing for more clusters provides more granular information within urban areas while maintaining consistency with the 4 cluster model. Figure B.9 presents the clustering results with the optimal number of clusters for every state. Note

that Figure 2 in the main manuscript provides more granular information for Washington, Texas, and California. Increasing the number of clusters beyond the optimal results in a finer partitioning of the embedding (Figure B.10).

We also demonstrate that by modeling the data with a single dimensional parameter in the nonlinear embedding along the dense tubular manifold matches the intuition provided by increasing the number of clusters for the GMM. For example, we constructed a single dimensional phase variable along the manifold based on the cosine similarity of the data points in the 2D nonlinear embedding space. The result is an even smoother transition across urban, periurban, suburban, and rural areas consistent across all four states, see Figure B.11.

B.7 Clustering in metropolitan areas: Georgia and California

Figures B.12 and B.13 present the clustering for metropolitan areas in Georgia and California, respectively.

B.8 Response Speed Distributions

We quantified the speed at which CBGs increased their stay-at-home behavior in response to the pandemic during a transition period between March and April (more specifically, March 10 – March 31) by the slope of a linear fit of the CBG mobility time series during the transition period (Figure B.14). Figure B.14 shows that the response speed distributions are directly correlated with CBG cluster assignment.

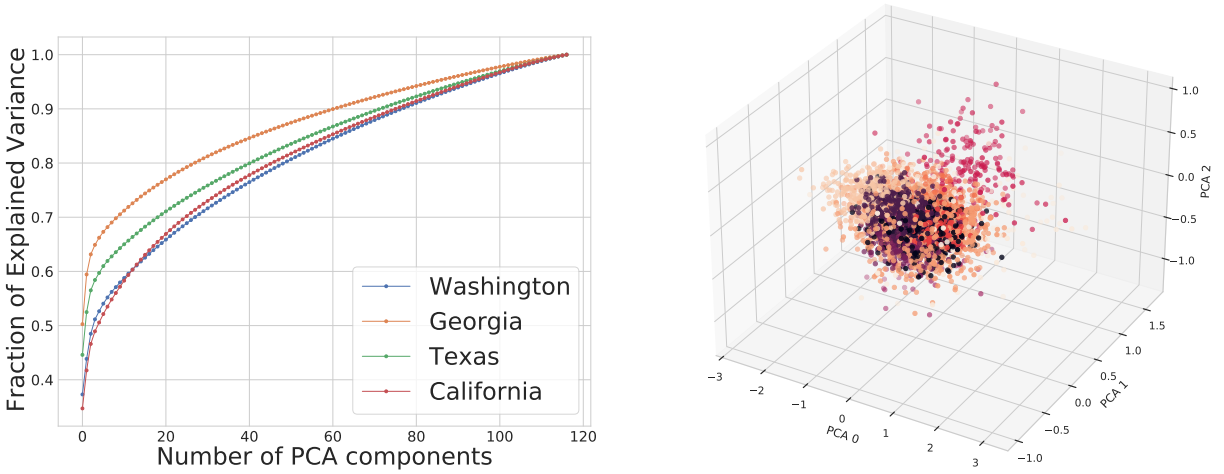


Figure B.1: **Linear Dimensionality Reduction Performance.** Left: fraction of explained variance vs. number of PCA components for every state. Right: 3D visualization of PCA projection of the mobility time series data for Washington State with 11 clusters highlighted in color. Clustering was done in 8D PCA space.

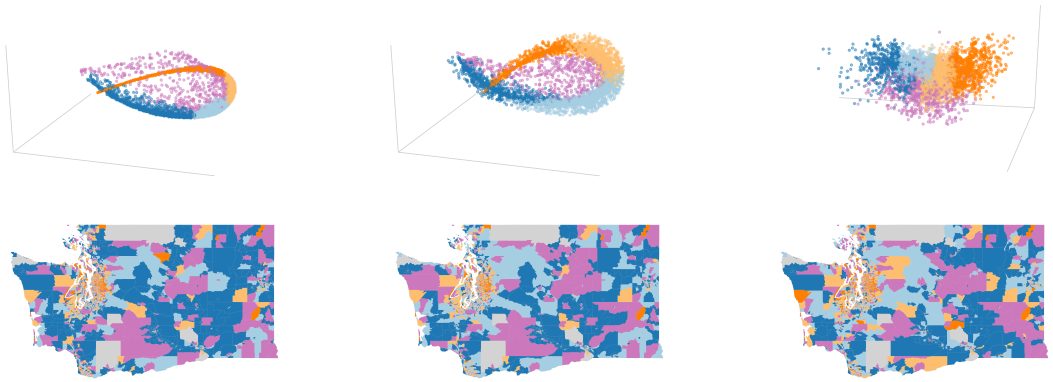


Figure B.2: **Nonlinear Dimensionality Reduction Embeddings and Corresponding Maps for Washington State.** Top Left: Laplacian Eigenmaps, Top Middle: Locally Linear Embedding, Top Right: Isomap. Bottom Left: Laplacian Eigenmaps clustering map, Bottom Middle: Locally Linear Embedding clustering map, Bottom Right: Isomap clustering map.

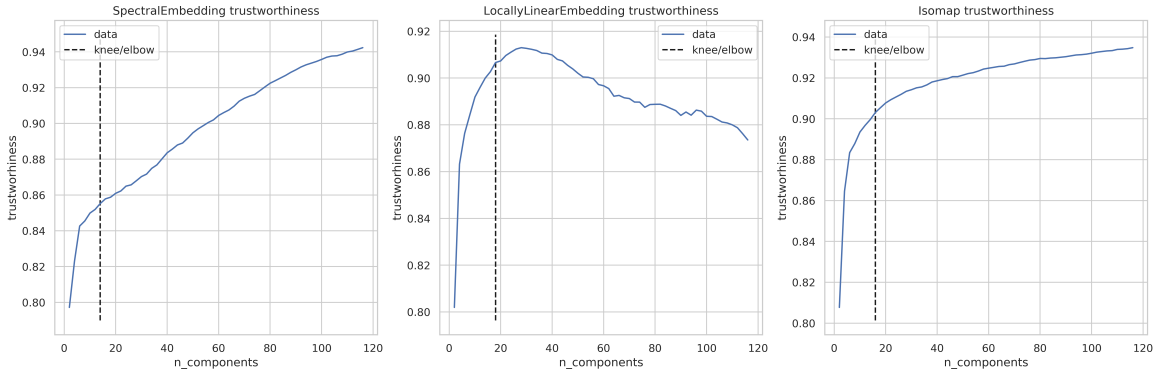


Figure B.3: **Optimal Embedding Dimensionality for Washington State.** Left: Laplacian Eigenmaps (optimal dimensionality 14), Middle: Locally Linear Embedding (optimal dimensionality 12), Right: Isomap (optimal dimensionality 16).

Figure B.4: **Laplacian Eigenmaps Hyperparameter Robustness.** Each panel presents a 3D Laplacian Eigenmaps embedding of Washington state mobility data for `n_neighbors` in $\{20, 30, 40, 50\}$ respectively.

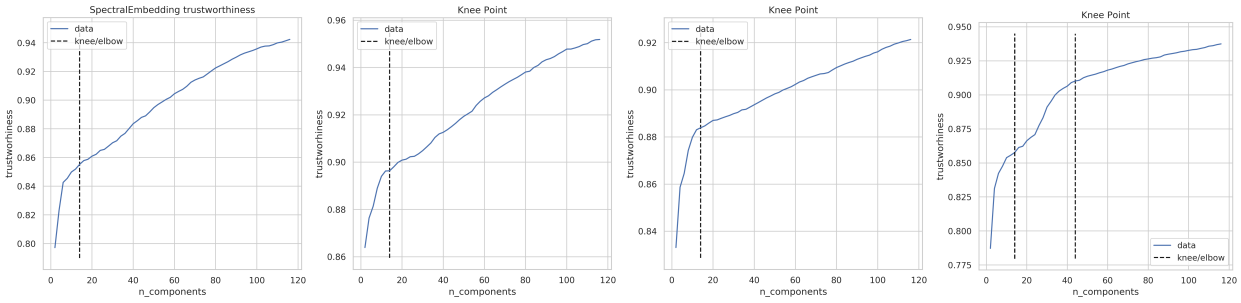


Figure B.5: **Optimal Laplacian Eigenmap Embedding Dimensionality for Every State.** Each panel presents trustworthiness vs number of Laplacian Eigenmap components for Washington (optimal dimensionality 14), Georgia (optimal dimensionality 14), Texas (optimal dimensionality 14), and California (optimal dimensionality 14 or 44) respectively.

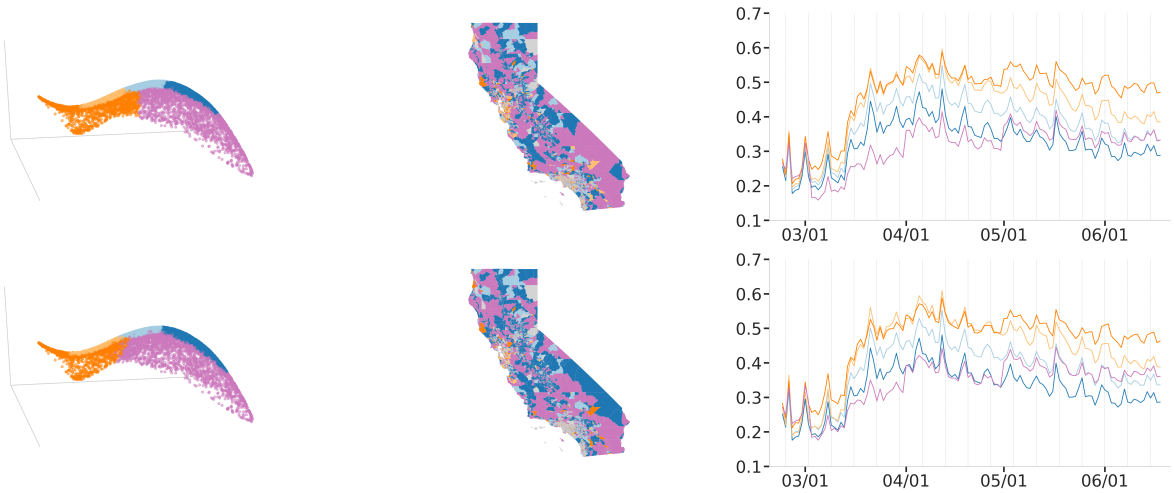


Figure B.6: **Comparison of two trustworthiness knee points for California.** Top: Clustering results using 14D Laplacian Eigenmap embedding (3D illustration of the embedding, geographic map and average mobility time series per cluster with clusters highlighted in color). Bottom: Clustering results using 44D Laplacian Eigenmap embedding (3D illustration of the embedding, geographic map and average mobility time series per cluster with clusters highlighted in color).

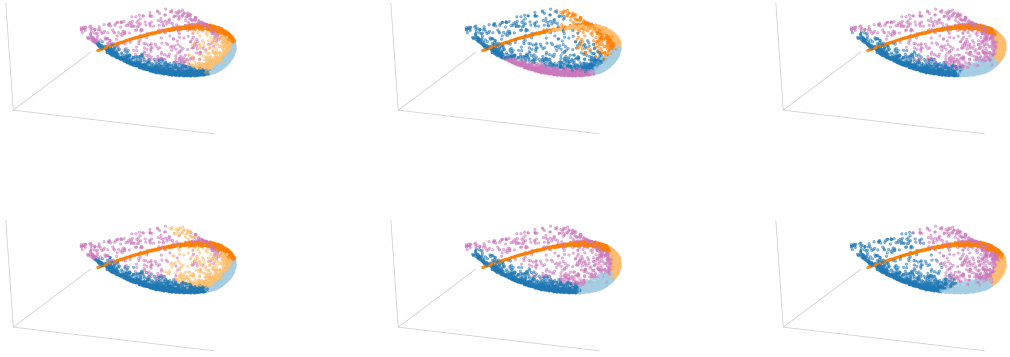


Figure B.7: **Nonconvexity robustness.** Different types of GMM clustering results obtained by refitting GMM several times.

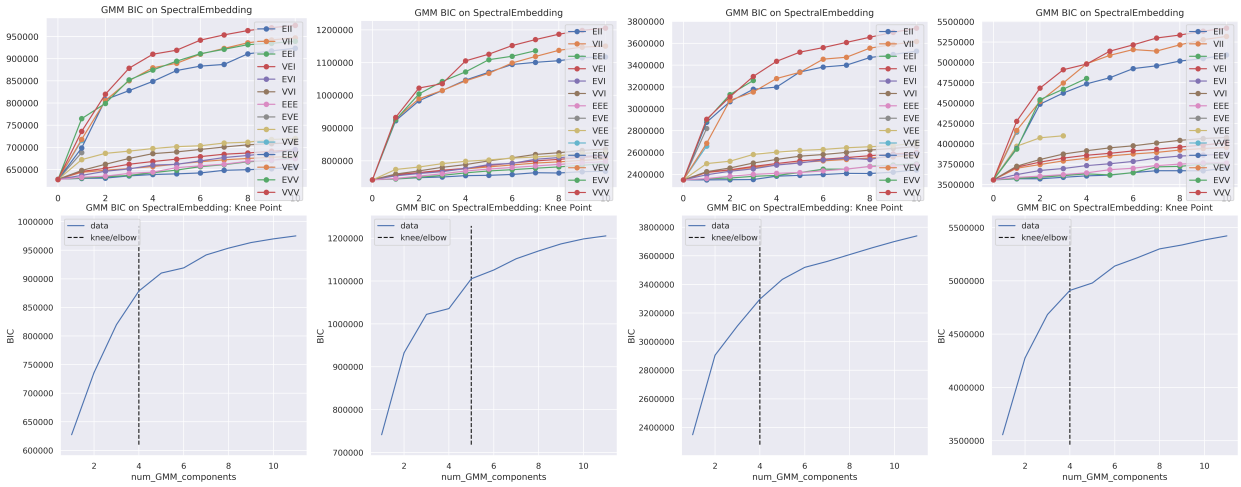


Figure B.8: **GMM model selection for every state based on BIC.** Top: BIC curves for different parametrizations of the GMM model as described in [179]. Bottom: Optimal number of GMM components identified using knee-point detection on the best BIC curve for Washington (optimal number of clusters 4), Georgia (optimal number of clusters 5), Texas (optimal number of clusters 4), and California (optimal number of clusters 4) respectively.

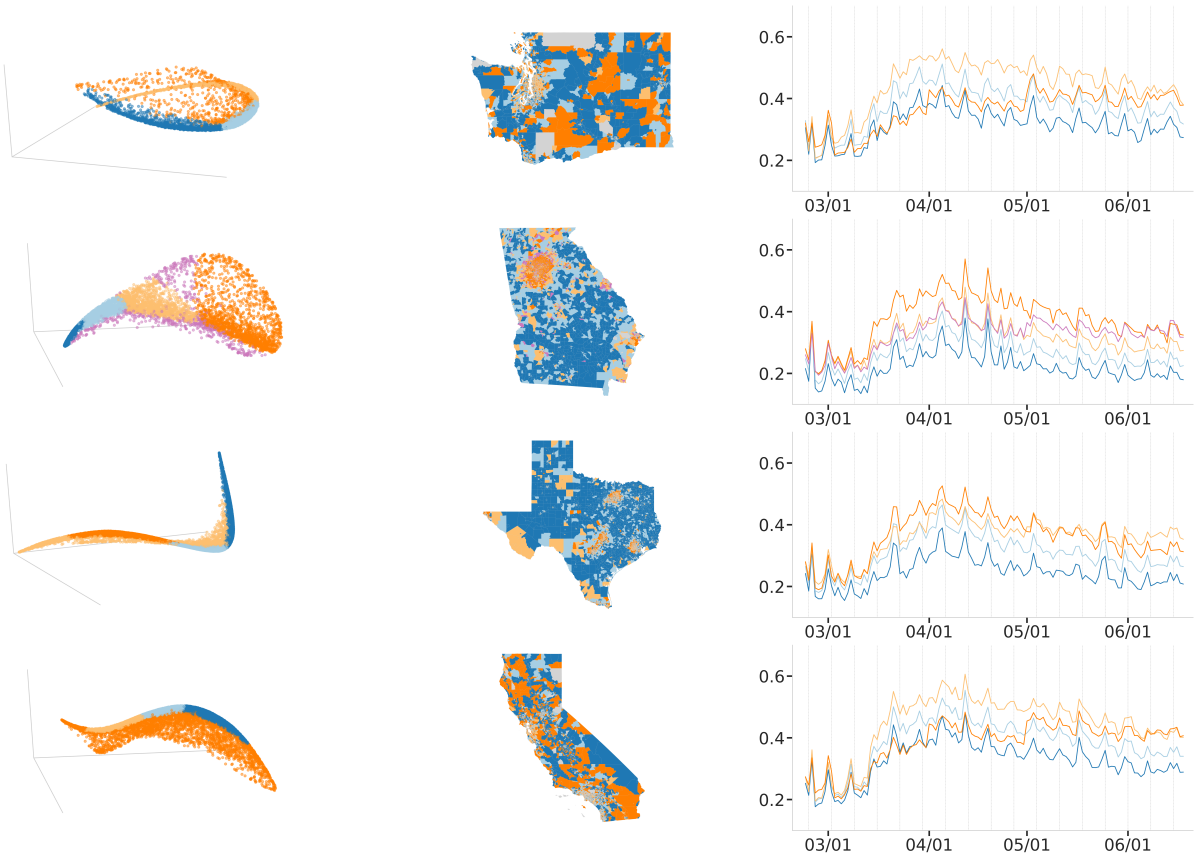


Figure B.9: **Clustering with the optimal number of clusters for every state**

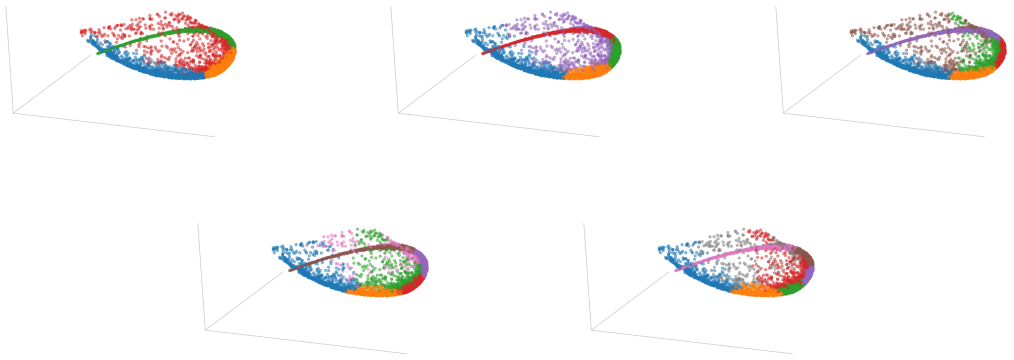


Figure B.10: **Bigger number of clusters results in finer partitioning of the embedding.** Panels present clustering for the number of clusters in $\{4, 5, 6, 7, 8\}$ respectively.

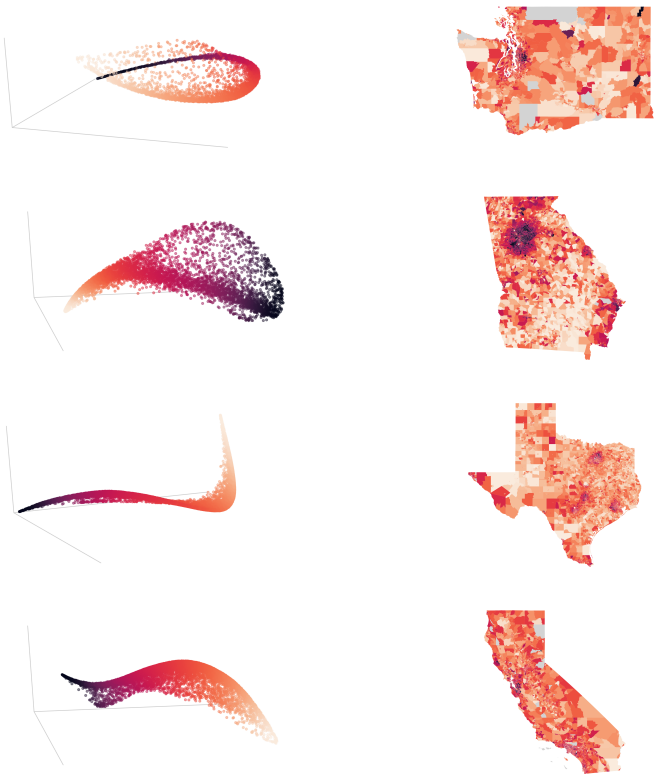


Figure B.11: **Continuous Colormap.** Smooth transition across urban, periurban, suburban, and rural areas in Washington, Georgia, Texas, and California.

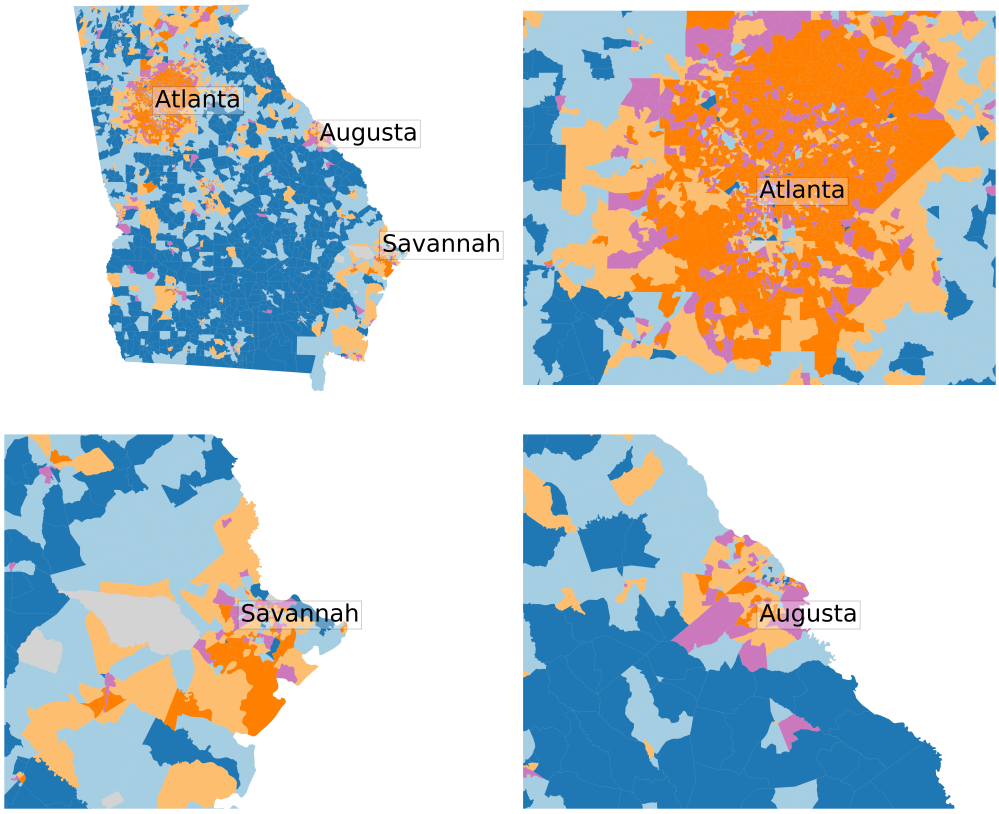


Figure B.12: **Clustering in metropolitan areas in Georgia**

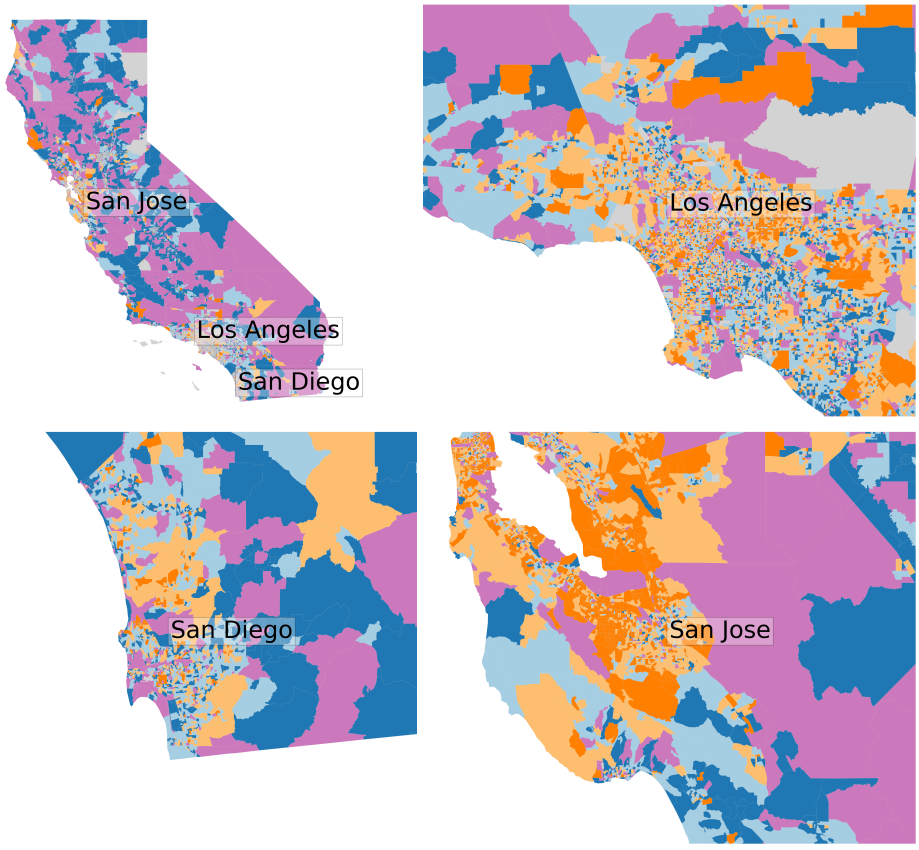


Figure B.13: **Clustering in metropolitan areas in California**

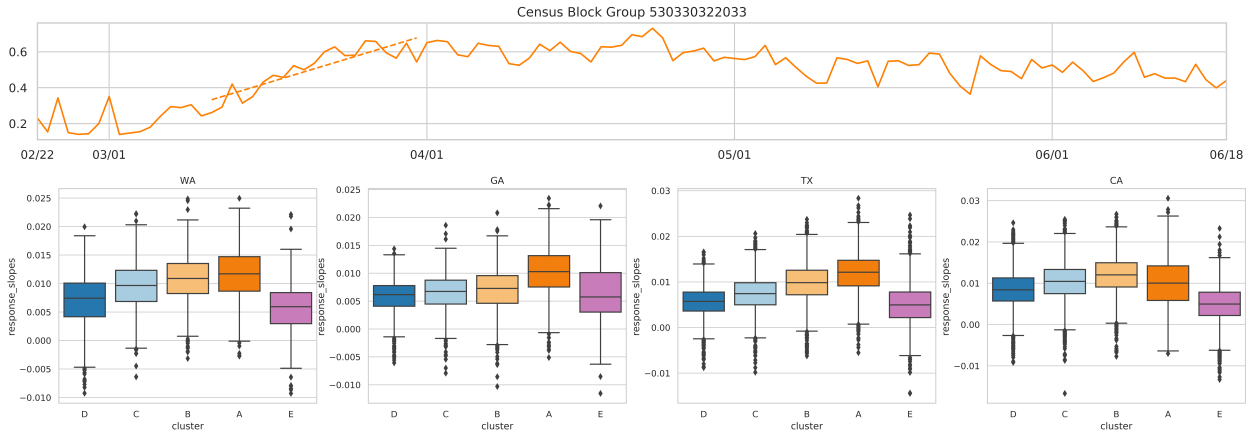


Figure B.14: **Response Speed Distributions.** Top: The response speed is quantified by the slope of a linear fit of the CBG mobility time series during the transition period of March 10 – March 31, dashed line represents that linear fit for an example CBG. Bottom: Response speed distributions for every state.

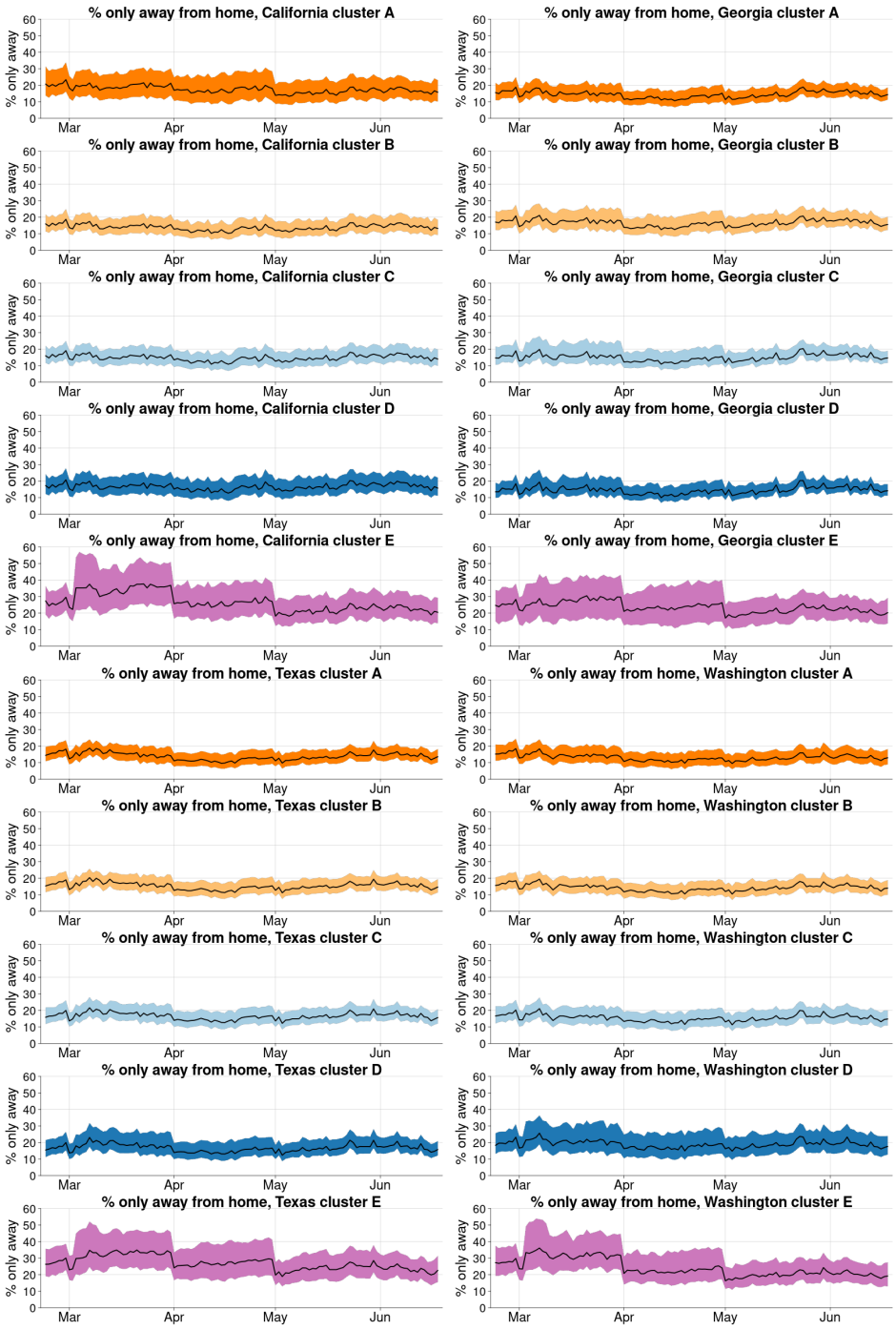


Figure B.15: **The fraction of devices that are *only* away from their homes each day.** The medians and inter-quartile range are shown for each cluster.

Appendix C

WHERE DO MODELS GO WRONG? PARAMETER-SPACE SALIENCY MAPS FOR EXPLAINABILITY

C.1 Additional Experiments

C.1.1 Parameter saliency profiles for other network architectures

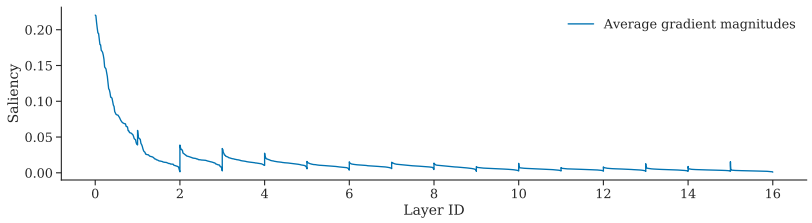
In this section, we present average saliency profiles for several popular network architectures other than ResNet-50 [82]. Analogously to Figure 4.1, Figure C.1 presents average gradient magnitudes for VGG-19 [187], Inception v3 [200], and DenseNet [86]. Similarly to Figure 4.2, we also present in Figure C.2 standardized filter-wise saliency profiles for those architectures averaged across correctly and incorrectly classified ImageNet [43] samples.

C.1.2 More examples of nearest neighbors

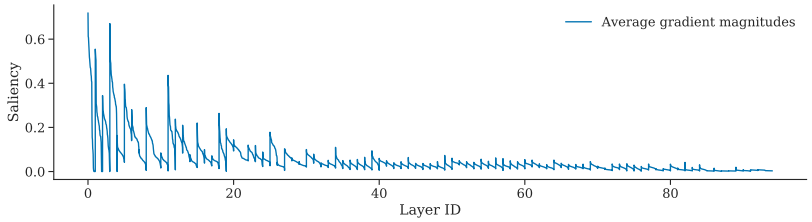
We present more examples of nearest neighbors in our parameter saliency space. Figure C.4 are nearest neighbors in CIFAR-10 [107] dataset, where reference images are chosen from samples misclassified by our classifier. Figure C.5 are examples from ImageNet, where images are captioned with the true label of the reference images.

In addition, in comparison with the nearest neighbor in our parameter saliency space, we also conduct the nearest neighbor search in the feature representation space. We take the feature representation from the `conv5_3` layer of a ResNet-50, and run the nearest neighbor search using the same reference images and candidate pool as in Figure 4.4. Results are shown in Figure C.3. We find that nearest neighbors in the feature space bear more resemblance in image structures, but it fails to identify samples that share the same mistakes. In fact, most of the nearest neighbors in Figure C.3 are correctly classified samples.

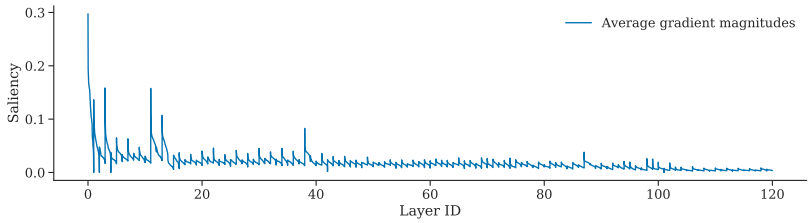
Furthermore, we present preliminary results on applying our method to language models.



(a) VGG-19



(b) Inception v3



(c) DenseNet

Figure C.1: **Filter-wise saliency profiles for other architectures.** (a) VGG-19 saliency profile (without standardization). (b) Inception v3 saliency profile (without standardization). (c) DenseNet saliency profiles (without standardization). In each panel the filter-wise saliency profile is averaged over the ImageNet validation set. In every panel, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis.

We use a BERT [46] model, pre-trained on the task of predicting the word at a masked position. We consider an independent dataset for evaluation in our experiments, which consists of short sentences with masked words, provided by LAMA [157, 158], an open-source language model analysis framework. Similar to the filter-wise aggregation in convolutional networks, we adopt column-wise aggregation for obtaining our saliency profiles in the transformer-based architecture. We conduct the nearest neighbor search by comparing sentences from

the dataset in saliency space and analyzing the top-5 nearest neighbors.

We present four examples below, where the numbering indicates n-th nearest neighbor sentence, and the italic word is the ground truth. Each example is accompanied with a description of similarities between the neighbors:

Reference: "Cany Ash and Robert Sakula are both Architects." incorrectly predicted as: actors

1. "David Castlles-Quintana and Vicente Royuela are *economists*." incorrectly predicted as: actors

2. "Raghuram Rajan is an *economist*." incorrectly predicted as: actor

3. "Richard G. Wilkinson and Kate Pickett are *British*." incorrectly predicted as: actors

4. "Nathan Alterman was a *poet*." correctly predicted: poet

5. "Zbigniew Badowski is an *architect*." incorrectly predicted as: author

Note that all 5 neighboring sentences here share the common structure of being declarations of profession for specifically named people. Interestingly, the first three closest sentences to the reference incorrectly predict the profession to be an actor as well. Moreover, the ground truth of the reference and its closest neighbor is economist/s.

Reference: "D'Olier Street is in Dublin." incorrectly predicted as: Paris

1. "A group who call themselves Huguenots lives in *Australia*." incorrectly predicted as: France

2. "Huguenots and Walloons settled in *Canterbury*." incorrectly predicted as: France

3. "In the Treaty of Lisbon 2007 *Ireland* refused to consent to changes." incorrectly predicted as: it

4. "Samuel Marsden Collegiate School is located in *Wellington*." incorrectly predicted as: Melbourne

5. "Konstantin Mereschkowski has *Russian* nationality." correctly predicted: Russian

Again, we see all five nearest neighbor sentences are semantically similar to the reference, this time relating to national affiliations and geography. In the first two closest sentences, the model incorrectly fills in a location with France, which is similar to the reference which incorrectly predicts Paris.

Reference: "The Super Bowl sponsor was the *Gap* clothing company." incorrectly predicted as Nike

1. "During Super Bowl 50 the *Nintendo* gaming company debuted their ad for the first time." incorrectly predicted as: video

2. "Experimental measurements on a model steam engine was made by *Watt*." incorrectly predicted as: Siemens

3. "ABC's programming strategy was criticized in May 1961 by *Life* magazine." incorrectly predicted as: Time

4. "In 2009, Doctor Who started to be shown on Canadian cable station *Space*." incorrectly predicted as: CBC

5. "To emphasize the 50th anniversary of the Super Bowl the *gold* color was used." incorrectly predicted as: blue

All but one of the nearest neighbors in this example relate to some kind of TV programming, and two also mention the Super Bowl. Much like the reference sentence, which incorrectly predicts the name of a corporation, the first four neighbors have a ground truth or incorrect prediction that is also a corporation.

Reference sample: "Tetzel's collections of money to free souls from purgatory was objected by *Luther*." incorrectly predicted as: some

1. "Newcastle was granted a new charter in 1589 by *Elizabeth*." incorrectly predicted as: Parliament

2. "The suggestion that imperialism was the "highest" form of capitalism is from *Lenin*." incorrectly predicted as: Aristotle

3. "Fritschel said the man's sleep was disturbed by *dreams*." incorrectly predicted as: lightning

4. "The concept that falling objects fell at the same speed regardless of weight was introduced by *Galileo*." incorrectly predicted as: NASA

5. "One of the earliest examples of Civil Disobedience was brought forward by the *Egyptians*." incorrectly predicted as: government

This is an example where there is a weak relation between the incorrect classifications (three of five involve some kind of government or government organization) of the nearest neighbors, but the sentences are still highly semantically similar. All of the neighbors except the third are declarations of historical actions performed by a specific person or group of people.

C.1.3 Fine-tuning salient filters of a VGG-19

In this section, we conduct the fine-tuning experiment introduced in Section 4.4.3 on a VGG-19 network trained on ImageNet, which has a total of 5504 filters. The learning rate for training our VGG network is 1/10 of that for the ResNet, so we decrease the fine-tuning step size by 10 in this experiment. Figure C.6 shows the effect of updating salient filters of a VGG-19. Note that we use the same range for the number of tunable filters in this

experiment; 300 filters correspond to 5.5% of total filters in a VGG-19, while it is 1.2% for a ResNet-50.

C.1.4 Random perturbation of salient filters

As an alternative to the pruning approach described in Section 4.4.1, random perturbations could be used to show that the most salient filters are indeed responsible for misclassification. We perturbed the filters using small Gaussian noise $\mathcal{N}(0, 0.001)$. Figure C.7 presents the effect of randomly perturbing salient filters, we observe similar trends to our pruning experiments in Section 4.4.1.

C.1.5 Connection to adversarial attacks in parameter space

Adversarial attacks in parameter space have been used for optimizers which find flat loss minima [63, 110, 49] and for improving model robustness through parameter-corruption-resistant training [198].

One could instead apply adversarial attacks to construct parameter saliency profiles – choose a constraint space and perturb parameters in order to minimize loss, subject to the constraint, using the perturbation to parameters as a saliency profile (perhaps standardizing afterwards). We notice several advantages and disadvantages of this alternative. On the one hand, the "adversarial" approach requires a choice of constraint space and may require more compute (our method results in the exact same saliency profile as a single-step adversary). On the other hand, the choice of constraint space and optimizer in the adversarial approach yields more flexibility.

In this section, we compare our method with the adversarial-attack-based method. More specifically, for a given sample (x, y), we perturb parameters either to maximize or minimize

the loss subject to an L2-norm constraint on the parameter change:

$$\min_{\theta} / \max_{\theta} \mathcal{L}_{\theta}(x, y) \quad \text{s.t. } \|\theta - \theta_0\| < \epsilon \tag{C.1}$$

Then, given the adversarially perturbed parameters θ^*, we define the adversarial-attack-based parameter saliency profile of the sample (x, y) as the absolute difference from the initial parameters θ_0: $s(x, y) = |\theta^* - \theta_0|$.

We compare the resulting adversarial saliency profiles with our original method on a random sample of 100 images from the ImageNet validation set. We observe that for a reasonably small constraint ($\epsilon = 10^{-4}$), the resulting adversarial saliency profiles are similar to our original parameter saliency profiles (as one would expect for smooth loss) with the average cosine similarity between the saliency profiles generated by each method for the same images reaching 0.99 (the average is taken over the random sample of 100 images). We also see that both methods agree on the top-k (we tried k=100) most salient filters: on average, 95% of filters identified by our original method as top-k salient filters were also identified as top-k salient filters by the adversarial parameter saliency. The differences were gradually more distinct with larger constraints, however, we note that the smaller epsilons are of greater interest since they reflect the intuition of perturbing only the most important parameters.

We also tried L1-regularized adversarial attacks. We can similarly enforce sparsity in our original method by simply adding the L1 regularizer to our loss before computing the gradient:

$$s(x, y)_i := |(1 - \alpha)\nabla_{\theta_i}(\mathcal{L}_{\theta}(x, y) + \alpha\|\theta - \theta_0\|_1)| \tag{C.2}$$

This modification can be seen as one step of the L1-regularized adversarial attack, and we experimentally checked that it produces very similar results with the cosine similarity between the resulting saliency profiles of 0.97 on average (for sufficiently large $\alpha = 0.99$).

C.1.6 Additional case study figures

Masking non-salient parts of the image. As noted in section 4.4.4 and presented in Figure C.8, masking the non-salient parts of the image results in even worse misclassification confidence with the incorrect class confidence increasing compared to the reference image.

Pasting the salient region from the reference image onto other "great white shark" images. As mentioned in section 4.4.4, in order to further investigate the effect of the salient region, we pasted it (i.e., the seal and the boat) from the original image onto other images with "great white shark" ground truth label that were correctly classified by ResNet-50 (see Figure C.9 for examples). We observed that this increased the probability of "killer whale" for 39 out of 40 examples of great white sharks from the ImageNet validation set with an average increase of 3.75%.

Examples of images with "killer whale" label. As we discussed in section 4.4.4, the model might have learned to correlate a combination of sea creatures (e.g. a seal) and man-made structures (e.g. a boat) with the "killer whale" label. Images of killer whales in ImageNet often have man-made structures which look similar to the boat, we provide examples of that in Figure C.10.

C.1.7 Exploring neural network mistakes

In Section 4.4.4, we apply our parameter-space saliency method as an explainability tool using it alongside our input-space technique to study how image features cause specific filters to malfunction. In our case study, the salient pixels that confuse the network are focused less on the target object than on other image features, and masking the salient regions which are not on the target object improves the network behavior. Such examples expose the network's reliance on spurious correlations and constitute an interesting type of model mistake.

The masking approach can be adopted to explore network mistakes and find other interesting cases (e.g., cases where the salient pixels are not concentrated on the target object). Investigating examples where masking the most salient pixels improves performance may

provide insights into the model's misbehavior as well as expose dataset noise and biases. We select misclassified samples where masking the top 5% of salient pixels leads to an increase of at least 25% in confidence corresponding to the correct class. We showcase representative examples of different types of mistakes that we observe in those samples in Figure 4.8. Many of the neural network misclassifications stem from classifying a non-target object in images with multiple objects (see Figure 4.8 (a), (b)). However, other mistakes are triggered by background features (Figure 4.8 (d)), dataset biases (as our case study experiments in Figure 4.7 suggest), and label noise (Figure 4.8 (e)).

Interestingly, in some of the selected cases the salient regions still focus on the target object features (see Figure 4.8 (c)), and masking them improves the model's behavior. Masking salient target object features that confuse the network seems to be particularly beneficial for images of animals; for example, masking dog ears helps the network identify the correct breed (see Figure C.11(a)) or masking spot patterns helps distinguish different types of rays (see Figure C.11(b)).

While masking the top 5 % of salient pixels results in correct classification for all samples in Figure 4.8 and in Figure C.11(a), (b), this is, of course, not always the case. Sometimes, pixels which cause large filter saliency values are focused densely on the target object, and masking them results in decreased confidence corresponding to the correct class. Selecting such samples can be used to find situations where the network is genuinely confused by the target object rather than background features (Figure C.11 (c)) or samples with multiple objects present and with salient pixels more focused on the target object (Figure C.11 (d)).

C.1.8 Comparison to GradCAM

Existing input saliency maps used with the predicted label can highlight features which are related to misclassification. However, they are not specifically geared towards that goal. Our input saliency technique highlights image features that cause specific filters to malfunction and those features, while they might in some cases coincide with the features that explain a high class confidence score corresponding to the predicted label, may not be the same.

In this section, we will compare our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network) to the visual explanations produced by GradCAM with the predicted label [1] [180] – one of the most popular and high quality input-saliency methods.

Figures C.12 and C.13 present panels of images comparing our method to GradCAM explanations computed with the predicted label. From the perspective of highlighting pixels responsible for neural network mistakes and for driving high filter saliency values, we note the following:

- GradCAM highlights the object that corresponds to the incorrect label, and the entire target object is highlighted in the images where only the target object is present (see Figure C.12 (c)-(e), Figure C.13 (a)-(c)). However, when our method focuses on the target object, it highlights specific features of that object. Those are the features that confuse the network, and masking them can correct the misclassification.

- In cases where the network classifies the non-target object in the image (see Figure C.12 (a)-(b), Figure C.13 (d)), both methods highlight the non-target objects. However, GradCAM is more localized to the non-target object. This is expected since GradCAM produces visual explanations for the predicted label (and has been shown to produce highly localized saliency maps [180]) while our method highlights all pixels that drive the filter saliency, and these pixels may be located on the target object as well.

- In cases where the misclassification does not stem from confusion by the target object or classifying the non-target object (see Figure C.12 (d)-(e), Figure C.13 (e)), our method highlights background features and/or a combination of non-target object features, while GradCAM still highlights the target object. For example, in Figure C.13 (e), our method highlights the boat and the sky much more than GradCAM, and our case

[1] Implementation from https://github.com/kazuto1011/grad-cam-pytorch under MIT license

study masking experiments in section 4.4.4 show that those regions indeed confuse the network.

- In addition, we emphasize that our input saliency technique is specific to the chosen filter set F and is introduced to study the interaction between the image features and the malfunctioning filters. In contrast, GradCAM is not able to relate image-space mistakes to an arbitrary set of model parameters or filters chosen by the user.

To summarize, GradCAM (as well as many other input-space saliency methods) was designed to be highly localized to the object correponding to the label of interest, while our method highlights sparse fine-grained features of images which we believe is a desirable property for our specific application. Therefore, we opt for using input-gradient information similar to the original Vanilla Gradient [186] method. However, instead of class confidence scores, we use a different loss – cosine distance to the boosted parameter-saliency profile (as described in Section 4.3.2) which allows us to explore how image features cause specific filters to malfunction.

C.1.9 Input-saliency sanity check

To assure that our input-space saliency technique is model dependent, we performed the model randomization test from [2]. We can see that the input saliency map is model dependent. We note that the data randomization test is not applicable in our case because our input-space saliency map is based on the parameter-saliency profile and parameter-saliency is designed to investigate a pretrained model with particular weights.

C.2 Implementation details

C.2.1 Hyper-parameter setting for fine-tuning salient filters

When tuning a small number of random or least salient filters, we re-normalize the gradient magnitude of these parameters to be the same as the salient filters for a fair comparison;

otherwise the gradients for these parameters are always smaller than the salient ones by the definition of our saliency profile, and updating them would make less change to a model than updating the salient ones. In addition to re-normalizing the gradients, we also multiply them with a step size, similar to the concept of learning rate in stochastic gradient descent. For ResNet-50, we set the step size to be 0.001, which equals to the learning rate of the last epoch when training the ResNet-50 on ImageNet from scratch. For VGG-19, we also set the fine-tuning step size to be the learning rate from the last training epoch, which is 0.0001. We also note that the batch normalization layers were set to the test mode for our fine-tuning experiments.

C.2.2 Input saliency visualization

The number of top salient filters to boost was chosen to be 10 in all input-space saliency experiments. The filters were boosted by multiplying by 100. For visualization, absolute input gradients were thresholded at 90-th percentile and Gaussian Blur with $(3, 3)$ kernel was applied.

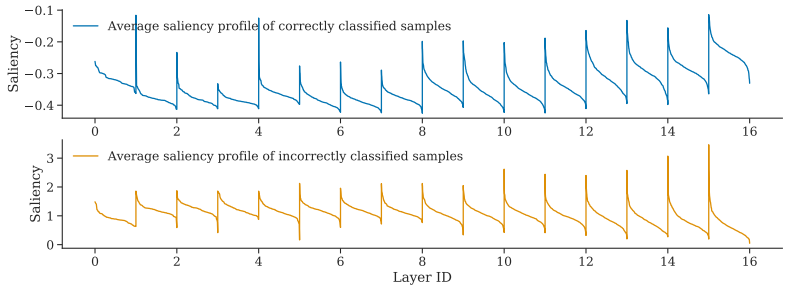
C.2.3 Hardware

The experiments were run on Nvidia GeForce RTX 2080Ti GPUs with 11Gb GPU memory on a machine with 4 cpu cores and 64Gb RAM. The input-space and parameter-saliency profiles take seconds to compute for a single sample.

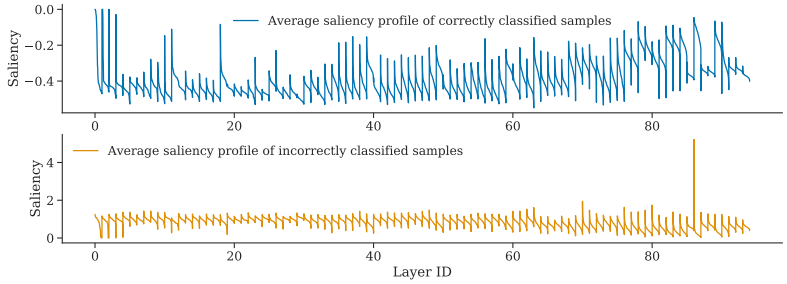
C.3 Limitations and Impact

Although our formulation of parameter saliency is not restricted to image datasets and CNNs, we only conduct experiments in these settings. In contrast, real-world data and models come in many forms. Explainability methods which shed light in some settings may fail to do so in others. Moreover, we emphasize that some erroneous model behaviors are simply difficult to understand through existing methods, and the capabilities of parameter

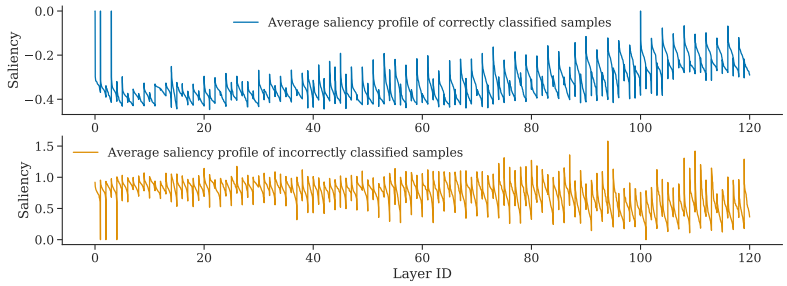
saliency are limited. In many applications, it is imperative that practitioners understand why their models behave as they do and that they are able to diagnose problems when they arise. We hope that our work helps to enable solutions to real-world problems. However, we caution against a false sense of security. Our visual interpretations of model behavior should be viewed as approximations as neural networks are incredibly complex.



(a) VGG-19



(b) Inception v3



(c) DenseNet

Figure C.2: **Standardized saliency profiles averaged over correctly vs incorrectly classified samples.** (a) VGG-19 saliency profiles. (b) Inception v3 saliency profiles. (c) DenseNet saliency profiles. In each panel, the top row presents the standardized saliency profiles averaged over correctly classified samples and the bottom row shows standardized saliency profiles averaged over incorrectly classified samples. On every panel, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis.

(a) (b)

Figure C.3: **Examples of nearest neighbors in the feature representation space (from ImageNet)**.

(a) bird ↔ cat (b) frog ↔ cat (c) airplane ↔ ship (d) horse ↔ deer

Figure C.4: **CIFAR-10 examples of nearest neighbors in parameter saliency space**. On CIFAR-10 images that cause similar filters to malfunction are often misclassified in a similar way.

(a) coyote (b) goldfish (c) terrapin (d) great white shark

Figure C.5: **ImageNet examples of nearest neighbors in parameter saliency space**. In every panel, the reference image is in the left column and its nearest neighbors are in the right column. Panels are captioned by the true label of their reference image.

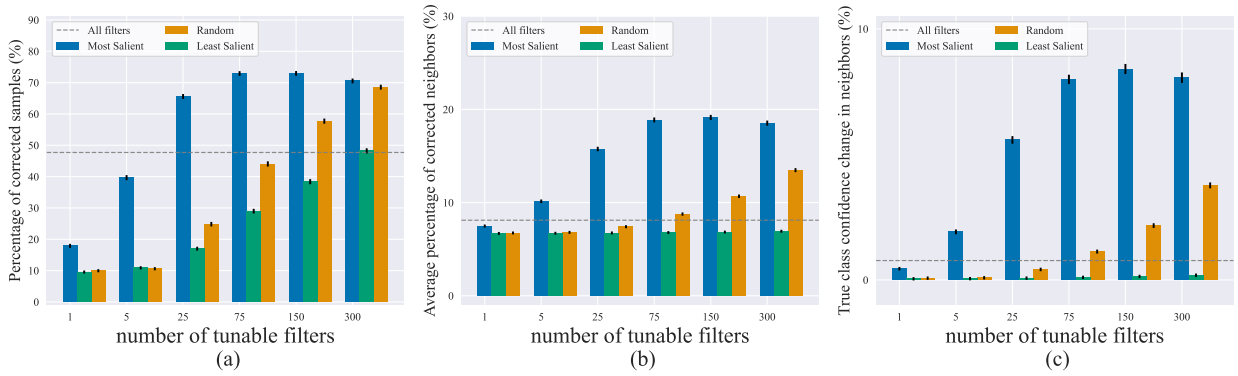


Figure C.6: **Effect of updating a small number of filters on VGG-19.** (a) Percentage of samples that are corrected after fine-tuning. (b) Average percentage of nearest neighbors that are also corrected after fine-tuning. (c) Average change in the confidence score of the true class among nearest neighbors. The horizontal line in each plot is the effect of updating the entire network.

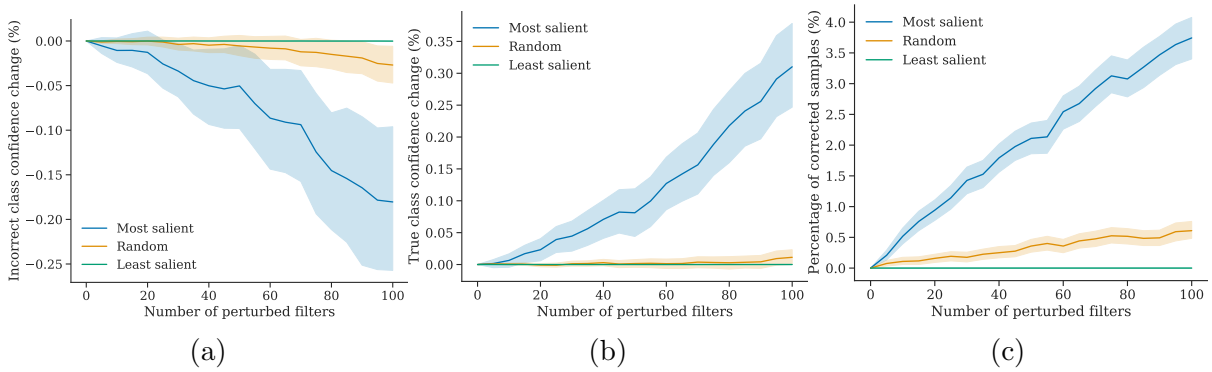


Figure C.7: **Effect of randomly perturbing filters.** (a) Change in incorrect class confidence score. (b) Change in true class confidence score. (c) Percentage of samples that were corrected as the result of pruning filters. These trends are averaged across all images misclassified by ResNet-50 in the ImageNet validation set. The error bars represent 95% bootstrap confidence intervals.

(a) Reference image
4.07% great white shark
78.36 % **killer whale**

(b) Input-space saliency for top 10 salient filters (pixels most responsible for misclassification)

(c) Masked human
3.84% great white shark
86.04% **killer whale**

(d) Masked non-salient water
5.15% great white shark
79.38% **killer whale**

Figure C.8: **Masking non-salient parts of the image.** (a) Reference image of "great white shark" misclassified by the model as "killer whale" and the corresponding confidence scores. (b) Pixels that cause the top 10 most salient filters to have high saliency. (c) Masked (non-salient) human. (d) Masked non-salient water region.

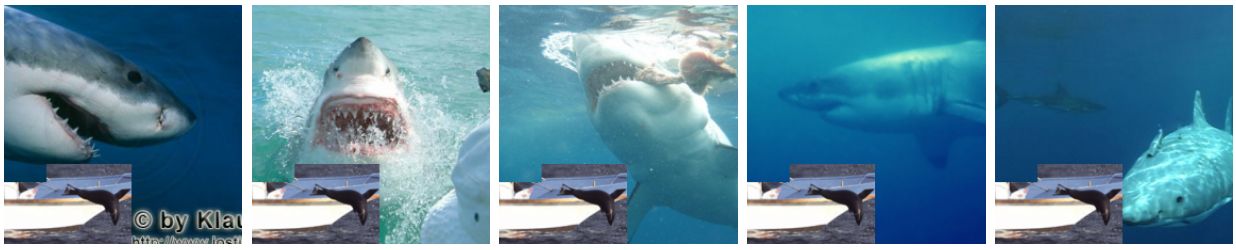


Figure C.9: **Sample "great white shark" images with boat and seal.** The salient region from the case study image pasted onto other "great white shark" images.

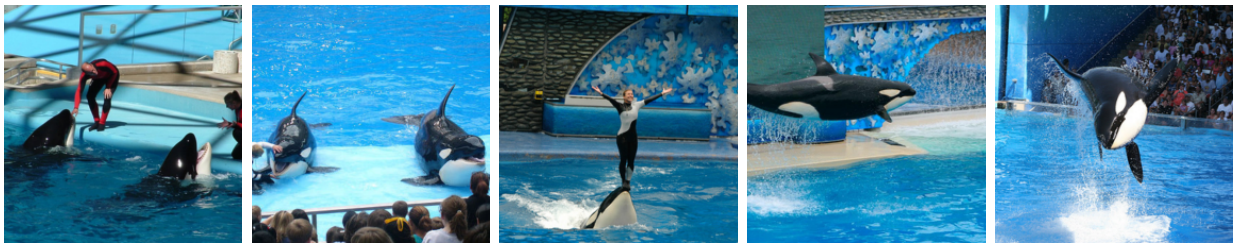


Figure C.10: **ImageNet examples of "killer whale".**

Figure C.11: **Pixels responsible for mistakes focused on the target object.** (a)-(b) Masking the salient pixels corrects the misclassification where masking confusing features (e.g. dog ears or spot patterns) helps distinguish animals. (c)-(d) Masking the salient pixels results in correct class confidence decrease, when the salient pixels are densely focused on the target object. The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel.

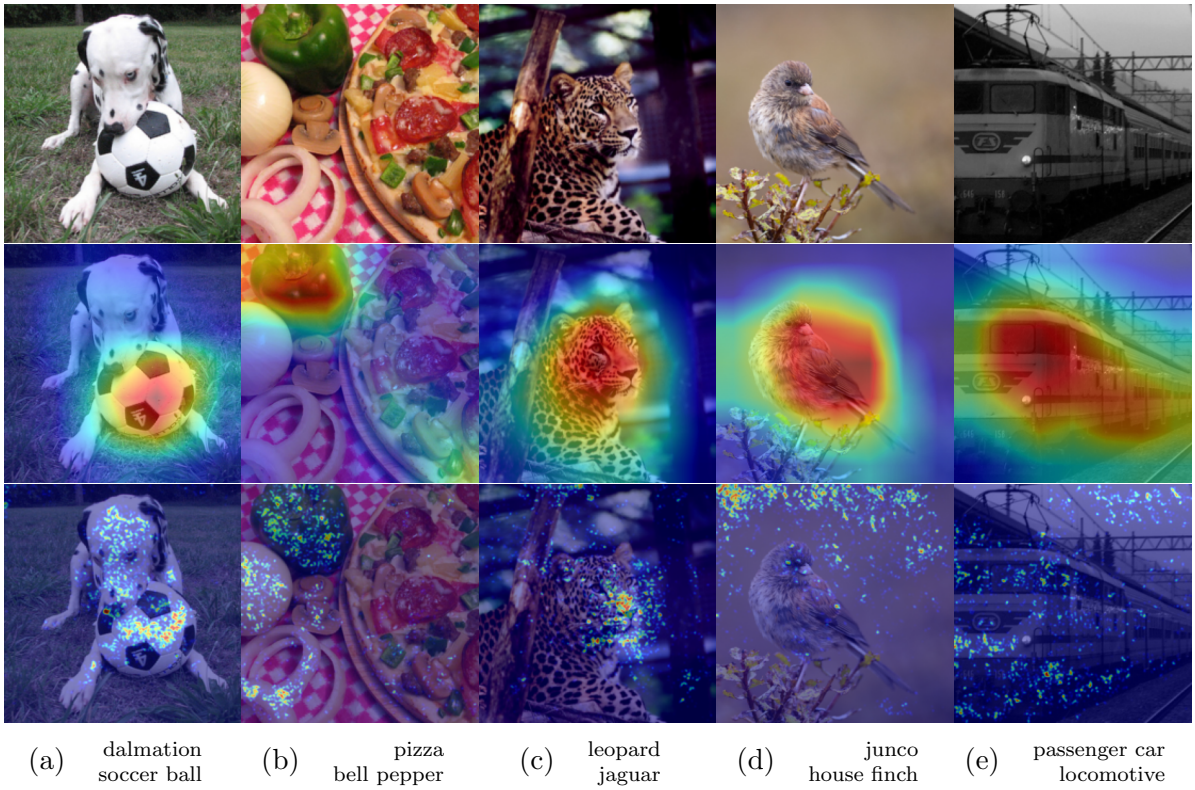


Figure C.12: **Comparison to GradCAM.** Top row: original image. Middle row: GradCAM input-space saliency map for the predicted label. Bottom row: our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network). The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel.

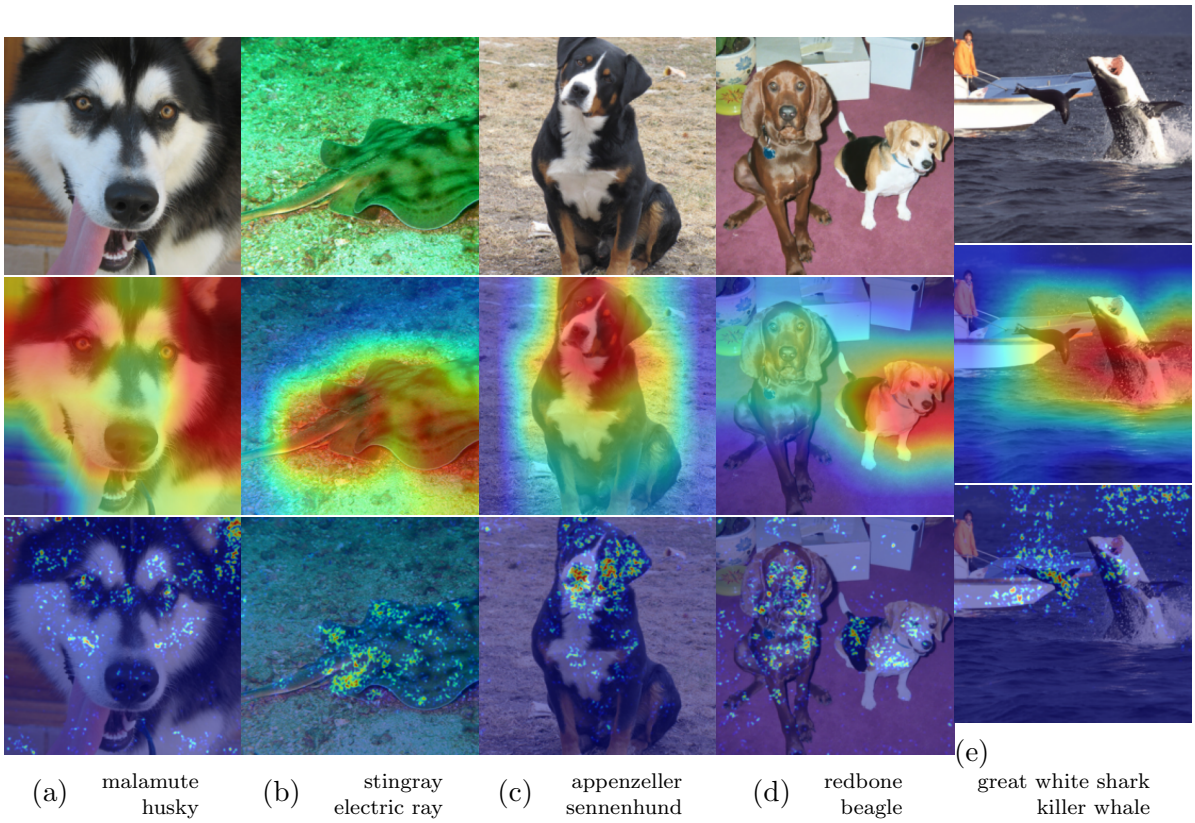


Figure C.13: **Comparison to GradCAM.** Top row: original image. Middle row: GradCAM input-space saliency map for the predicted label. Bottom row: our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network). The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel.

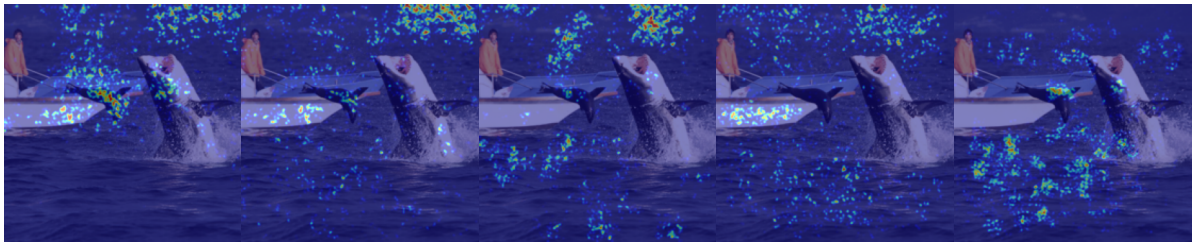


Figure C.14: **Sanity checks.** (a) No randomization of ResNet-50. (b) Only stage 4 of ResNet-50 is randomized. (c) Stages 3-4 of ResNet-50 are randomized. (d) Stages 2-4 of ResNet-50 are randomized. (e) The entire ResNet-50 is randomized.

Appendix D

TRANSFER LEARNING WITH DEEP TABULAR MODELS

D.1 Limitations and Impact

In this section we discuss limitations of our work. We note that transfer learning with deep tabular models requires more computational resources than training XGBoost or CatBoost, especially when the hyperparameter tuning is used thus incurring additional costs on practitioners adopting our approach. In addition, while we handle the cases of differing upstream and downstream feature sets thus allowing for transfer learning with heterogeneous data, our approach relies on having reasonably similar (but not identical) upstream and downstream feature sets and related upstream and downstream tasks and we do not propose a foundation model for tabular data.

D.2 Experimental Details.

D.2.1 Hardware

We ran our experiments on NVIDIA GeForce RTX 2080 Ti machines.

D.2.2 Additional Details on MetaMimic Repository

The MetaMIMIC Repository [74, 231] is based on the publicly accessible MIMIC database [69, 96, 95]. Regarding the patient consent to collect this data, as stated in [96]: "The project was approved by the Institutional Review Boards of Beth Israel Deaconess Medical Center (Boston, MA) and the Massachusetts Institute of Technology (Cambridge, MA). Requirement for individual patient consent was waived because the project did not impact clinical care and all protected health information was deidentified". The MIMIC database

is freely available to any person upon completion of the credentialing process found at the following link: `https://mimic-iv.mit.edu/docs/access/` and is distributed under Open Data Commons Open Database License v1.0, please see the following link for details: `https://physionet.org/content/mimic-iv-demo-omop/view-license/0.9/`.

D.2.3 Implementation Licenses

For the model implementations and training routines we adapt the code from the following publicly available repositories and libraries:

- RTDL [1] under MIT License
- TabTransformer [2] under MIT License
- Catboost [3] under Apache License
- XGBoost [4] under Apache License

D.2.4 Data Preprocessing

We preprocess numerical features with quantile transformation with standard output distribution for neural networks and we use original features for GBDT models. Quantile transformer is first fit on upstream data and then applied to downstream data to preserve similar feature distribution in upstream and downstream data. We note, that using the same normalization for upstream and downstream data is a very important step for transfer learning. We impute missing values with mean values for numerical features and with a new category for categorical features.

---

[1] `https://github.com/Yura52/tabular-dl-revisiting-models`

[2] `https://github.com/lucidrains/tab-transformer-pytorch`

[3] `https://catboost.ai/`

[4] `https://xgboost.ai/`

D.2.5 Training Details

Supervised Pre-training

All deep models are trained with AdamW optimizer [128]. We pre-train models on upstream data for 500 epochs with patience set to 30, meaning that we continue training until there are 30 consecutive epochs without improvement on the validation set, but no longer than 500 epochs. Since we assume limited data availability for downstream task, we do not sample a validation set for early stopping. Instead, we fine-tune/train models from scratch for 200 epochs and choose an "optimal" epoch as discussed in Section D.2.5. We use learning rate $1e-4$ for training from scratch on downstream data and learning rate $5e-5$ for fine-tuning pre-trained models. For pre-training, learning rate and weight decay are tunable hyperparameters for each model and each pre-training dataset (collection of $n-1$ upstream tasks). Batch size was set to 256 in all transfer-learning experiments.

Self-supervised Pre-training

MLM Pre-training

To implement MLM pre-training for tabular data, for each training sample in a batch we randomly select a feature to replace with a masking token (in the embedding space), which is unique for every feature. We also initialize a distinct fully connected layer, or head, for each column which is used to predict the masked values of that feature. We compute the appropriate loss, cross-entropy for categorical features and MSE for numerical ones, using the output from the head corresponding to the masked feature.

Our masked pre-training routine has very no additional hyperparameters. The only deviations form the pre-training hyperparameters listed above are that we use larger batch sizes of 512 and we do not employ patience. In this setting we pre-training for a full 500 epochs.

Since MLM pre-training requires training additional classification heads for every feature

in the data, it dramatically increases the memory requirements. Because of that we limit the experiments with self-supervised pre-training to using the default FT-transformer configuration, including the setups we compare to, that is training from scratch and using supervised pre-training.

Contrastive Learning

We adapt the implementation of contrastive learning for tabular data from [190]. The pre-training loss consists of two components, the first is the InfoNCE contrastive loss, which minimizes the distance between the representation of two views of the same data point (original and augmented samples), while maximizing the distance in the feature space between different samples. The second component is denoising loss, which is used to train an MLP head to predict the original sample from a noisy view of that sample. For a detailed explanation of contrastive pre-training for tabular data we refer the reader to [190]. To construct positive pairs for contrastive loss and noisy samples for denoising loss we use two data aumentations: CutMix in the input space and Mixup in the embedding space [241, 244].

Formally, let a batch of data be represented by $X = \{x_i\}_{i=0}^{n}$, where each x_i has q features. Let m denote a binary mask (over the features of any x_i) with each entry being a one with probability p. Then CutMix augmentation of a sample in the input space is computed as

$$x_i^{\text{CutMix}} = m \times x_i + (1 - m) \times x_j$$

where j is a random index chosen from $[0, n]$. To define Mixup in the embedding space, let $\hat{X}^{\text{CutMix}}$ be the set of the embeddings of the entire CutMix-ed batch. Then, Mixup augmentation is computed as a convex combination

$$\hat{x_i}^{\text{Mixup}} = \mu \hat{x}_i^{\text{CutMix}} + (1 - \mu)\hat{x}_k^{\text{CutMix}}$$

where k is again a random index chosen from $[0, n]$.

The hyperparameters are similar to supervised pre-training with only several modifications. First, we use smaller batch sized of 200 and we train for a full 500 epochs (no patience). Also, contrastive learning has additional hyperparameters m and μ, both of which we set to 0.9.

Epoch Selection

Since we work with limited downstream data, sampling a validation set for early stopping is not always possible. Therefore, we select the number of fine-tuning epochs for deep models as follows. For the data levels of 4 and 10 samples, we simply select 30 fine-tuning epochs. For more data of 20 samples, we select 60 fine-tuning epochs. In the larger data levels of 100 and 200 samples, we sample 20% of the data as a validation set to perform early stopping with the flexible end-to-end fine-tuned transfer learning setups prone to overfitting. For early stopping, we terminates training if no improvement in the validation score is observed for more than 30 epochs. In the less flexible transfer learning setups with a frozen feature extractor, we select 100 fine-tuning epochs for the MLP head atop a frozen feature extractor and 200 fine-tuning epochs for the linear head atop a frozen feature extractor. Finally, for the deep baselines with the hyperparameters tuned on a small subsample of the upstream data, we select the best epoch from the small upstream subsample.

Stacking for GBDT Models

For XGBoost models we incorporate stacking by training 11 additional XGBoost classifiers on upstream tasks and stack their predictions as additional features for downstream data. For Catboost we apply the same strategy, but we train a single multi-label Catboost classifier predicting all 11 upstream labels.

Statistical Significance

We compute ranks taking into account statistical significance of the performance differences between the models; on a given task, top models without statistically significant performance difference all receive rank 1. To compute statistical significance we use the one-sided Wilcoxon Rank-Sum test [228, 135] with $p = 0.05$. We run each experiment with 10 random seeds for the experiments in Section 5.4 and 5 random seeds for the experiments in Sections 5.5, 5.6.

D.3 Hyperparameter Tuning

In this section we provide hyperparameter search spaces and distributions for Optuna for each model, which are adapted from the original papers. For Catboost and XGBoost models we use search spaces proposed in [71]. We run 50 Optuna trials to tune hyperparameters for each model. In our experiments we tune the hyperparameters on full upstream data for deep tabular models with transfer learning, and on upstream data of the same size as downstream data for deep baselines trained from scratch and for GBDT models.

FT-Transformer

The hyperparameter search space and distributions as well as the default configuration are presented in Table E.1. The number of heads is always set to 8 as recommended in the original paper.

ResNet

The hyperparameter search space and distributions as well as the default configuration are presented in Table D.2

Table D.1: **Optuna hyperparameter search space and default configuration for FT-Transformer**

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 4]$	3
Feature embedding size	UniformInt$[64, 512]$	192
Residual dropout	$\{0$, Uniform$[0, 0.2]\}$	0.0
Attention dropout	Uniform$[0, 0.5]$	0.2
FFN dropout	Uniform$[0, 0.5]$	0.1
FFN factor	Uniform$[2/3, 8/3]$	$4/3$
Learning rate	LogUniform$[1e-5, 1e-3]$	$1e-4$
Weight decay	LogUniform$[1e-6, 1e-3]$	$1e-5$

Table D.2: **Optuna hyperparameter search space and default configuration for ResNet**

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 8]$	5
Category embedding size	UniformInt$[64, 512]$	128
Layer size	UniformInt$[64, 512]$	200
Hidden factor	Uniform$[1, 4]$	3
Hidden dropout	Uniform$[0, 0.5]$	0.2
Residual dropout	$\{0$, Uniform$[0, 0.5]\}$	0.2
Learning rate	LogUniform$[1e-5, 1e-2]$	$1e-4$
Weight decay	$\{0$, LogUniform$[1e-6, 1e-3]\}$	0.0

MLP

The hyperparameter search space and distributions as well as the default configuration are presented in Table D.3

TabTransformer

The hyperparameter search space and distributions as well as the default configuration are presented in Table D.4.

Table D.3: **Optuna hyperparameter search space and default configuration for MLP**

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 8]$	3
Category embedding size	UniformInt$[64, 512]$	128
Layer size	UniformInt$[1, 512]$	$[300, 200, 300]$
Dropout	$\{0$, Uniform$[0, 0.5]\}$	0.2
Learning rate	LogUniform$[1e-5, 1e-2]$	$1e-4$
Weight decay	$\{0$, LogUniform$[1e-6, 1e-3]\}$	$1e-5$

Table D.4: **Optuna hyperparameter search space and default configuration for TabTransformer**

Parameter	Search Space	Default
Number of heads	UniformInt$[2, 8]$	8
Number of layers	UniformInt$[1, 12]$	6
Category embedding size	UniformInt$[8, 128]$	32
Attention Dropout	Uniform$[0.0, 0.5]$	0.0
FF Dropout	Uniform$[0.0, 0.5]$	0.0
Learning rate	LogUniform$[1e-6, 1e-3]$	$1e-4$
Weight decay	LogUniform$[1e-6, 1e-3]$	$1e-5$

Catboost

The hyperparameter search space and distributions are presented in Table E.2. For default configuration we use default parameters from the Catboost library [162].

XGBoost

The hyperparameter search space and distributions as well as the default configuration are presented in Table E.3. For default configuration we use default parameters from the XGBoost library [32].

Table D.5: **Optuna hyperparameter search space for Catboost**

Parameter	Search Space
Iterations	UniformInt$[2, 1000]$
Max depth	UniformInt$[3, 10]$
Learning rate	LogUniform$[1e-5, 1]$
Bagging temperature	Uniform$[0, 1]$
L2 leaf reg	LogUniform$[1, 10]$
Leaf estimation iterations	UniformInt$[1, 10]$

Table D.6: **Optuna hyperparameter search space for XGBoost**

Parameter	Search Space
Num estimators	UniformInt$[2, 1000]$
Max depth	UniformInt$[3, 10]$
Min child weight	LogUniform$[1e-8, 1e5]$
Subsample	Uniform$[0.5, 1]$
Learning rate	LogUniform$[1e-5, 1]$
Col sample by level	Uniform$[0.5, 1]$
Col sample by tree	Uniform$[0.5, 1]$
Gamma	$\{0$, LogUniform$[1e-8, 1e2]\}$
Lambda	$\{0$, LogUniform$[1e-8, 1e2]\}$
Alpha	$\{0$, LogUniform$[1e-8, 1e2]\}$

D.3.1 Tuning GBDT Models

In Table D.7 we explore the effectiveness of our hyperparameter tuning strategy for GBDT models. In particular, we compute average ranks for models with tuned and default configurations. Recall, because of the limited data availability in downstream tasks, we tune the hyperparameters on upstream data of the same size as the downstream data using full-size validation set. We find that using upstream data for tuning hyperparameters is effective for XGBoost model while for catboost tuned configuration outperforms default configuration at three out of five data availability levels.

Table D.7: **Comparison of tuned and default configurations of GBDT models.** The table displays ranks computed pair-wise for default/tuned configurations of XGBoost and Catboost models.

Num samples	4	10	20	100	200
XGBoost Tuned	1.17	1.17	1.25	1.33	1.42
XGBoost Default	1.42	1.83	1.67	1.50	1.58
Catboost Tuned	1.33	1.08	1.17	1.25	1.42
Catboost Default	1.25	1.25	1.42	1.33	1.33

D.3.2 Tuning Deep Baselines

In Table D.8 we evaluate hyperparameter tuning routine for deep baselines, i.e. deep neural networks trained from scratch. We find that unlike GBDT models, hyperparameters tuned on upstream data do not transfer to downstream tasks at lower data regimes (i.e. 4-20 samples) for deep models. However, tuning helps deep baselines in higher data regimes.

Table D.8: **Comparison of tuned and default configurations of deep tabular baselines.** The table displays ranks computed pair-wise for default/tuned configurations of deep tabular neural networks.

Num samples	4	10	20	100	200
FT-Transformer Tuned	1.33	1.25	1.00	1.33	1.33
FT-Transformer Default	1.00	1.08	1.58	1.417	1.50
ResNet Tuned	1.25	1.17	1.33	1.25	1.75
ResNet Default	1.00	1.08	1.25	1.00	1.00
MLP Tuned	1.42	1.67	1.58	1.17	1.33
MLP Default	1.00	1.08	1.33	1.67	1.50
TabTransformer Tuned	1.25	1.33	1.17	1.08	1.17
TabTransformer Default	1.17	1.25	1.33	1.50	1.67

D.3.3 Tuning Deep Models for Transfer Learning

In Table D.9 we evaluate the hyperparameter tuning strategy for deep tabular neural networks with transfer learning. Recall, we tune the hyperparameters on the full upstream data which is then used for pre-training. We find this strategy helpful for most models, especially for FT-Transformer and TabTransformer, and for most transfer learning setups.

Table D.9: **Comparison of tuned and default configurations of deep tabular neural networks with transfer learning.** The table displays ranks computed pair-wise for default/tuned configurations of deep tabular models fine-tuned with 4 different transfer-learning setups.

Num Samples	4	10	20	100	200
FT-Transformer + LH-E2E Tuned	1.00	1.00	1.00	1.08	1.17
FT-Transformer + LH-E2E Default	1.25	1.25	1.17	1.42	1.17
FT-Transformer + MLP-E2E Tuned	1.00	1.00	1.08	1.00	1.08
FT-Transformer + MLP-E2E Default	1.08	1.00	1.25	1.33	1.58
FT-Transformer + LH Tuned	1.00	1.00	1.00	1.00	1.00
FT-Transformer + LH Default	1.17	1.25	1.33	1.58	1.67
FT-Transformer + MLP Tuned	1.08	1.00	1.00	1.00	1.00
FT-Transformer + MLP Default	1.17	1.08	1.33	1.58	1.75
ResNet + LH-E2E Tuned	1.00	1.17	1.33	1.25	1.33
ResNet + LH-E2E Default	1.08	1.00	1.00	1.17	1.25
ResNet + MLP-E2E Tuned	1.00	1.00	1.08	1.08	1.00
ResNet + MLP-E2E Default	1.17	1.00	1.08	1.58	1.42
ResNet + LH Tuned	1.00	1.00	1.08	1.00	1.00
ResNet + LH Default	1.17	1.25	1.25	1.50	1.50
ResNet + MLP Tuned	1.00	1.00	1.00	1.00	1.08
ResNet + MLP Default	1.17	1.17	1.33	1.58	1.75
MLP + LH-E2E Tuned	1.17	1.25	1.42	1.42	1.17
MLP + LH-E2E Default	1.08	1.08	1.08	1.33	1.50
MLP + MLP-E2E Tuned	1.00	1.08	1.33	1.25	1.42
MLP + MLP-E2E Default	1.08	1.00	1.25	1.25	1.33
MLP + LH Tuned	1.00	1.25	1.17	1.17	1.08
MLP + LH Default	1.08	1.08	1.17	1.17	1.25
MLP + MLP Tuned	1.08	1.08	1.08	1.00	1.00
MLP + MLP Default	1.00	1.00	1.17	1.17	1.42
TabTransformer + LH-E2E Tuned	1.00	1.00	1.08	1.08	1.00
TabTransformer + LH-E2E Default	1.17	1.25	1.25	1.50	1.25
TabTransformer + MLP-E2E Tuned	1.00	1.00	1.08	1.08	1.00
TabTransformer + MLP-E2E Default	1.17	1.25	1.25	1.42	1.25
TabTransformer + LH Tuned	1.00	1.00	1.00	1.00	1.00
TabTransformer + LH Default	1.25	1.17	1.25	1.25	1.17
TabTransformer + MLP Tuned	1.00	1.00	1.00	1.00	1.00
TabTransformer + MLP Default	1.25	1.33	1.25	1.25	1.25

D.4 Additional Results

D.4.1 Results for TabTransformer

Figure D.1 is equivalent to Figure 5.2, but also includes the results for TabTransformer model, a tabular neural network consisting of transformer block for categorical features and an MLP block on top for numerical features. We find that TabTransformer performs poorly compared to other deep tabular neural networks, which might be explained by the fact that the original paper huang2020tabtransformer only suggests to tune the hyperparameters of transformer block, but not the MLP part, while Meta-MIMIC data has only one categorical features and 171 numerical features.

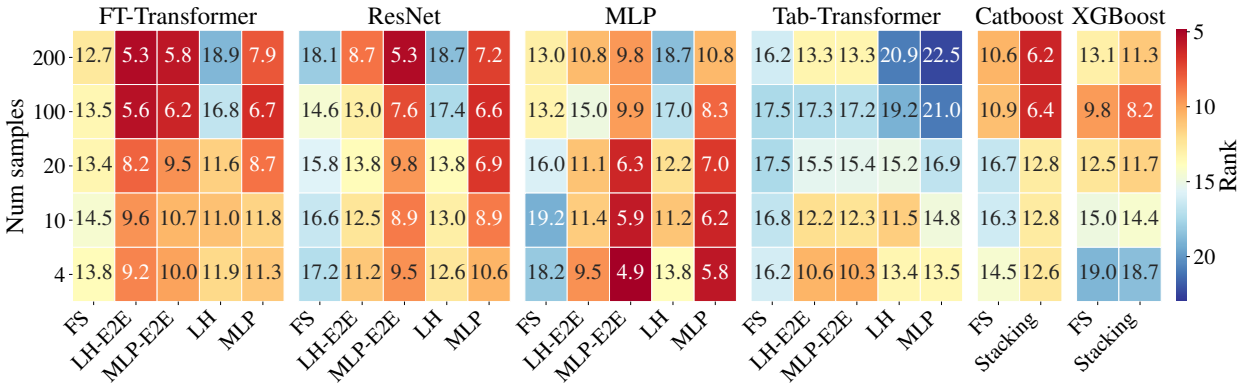


Figure D.1: **Average model ranks including TabTransformer.** Deep tabular models and GBDT performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks.

D.4.2 Model-Wise Ranks

In Figure D.2 we compare different transfer learning setups for each deep tabular model.

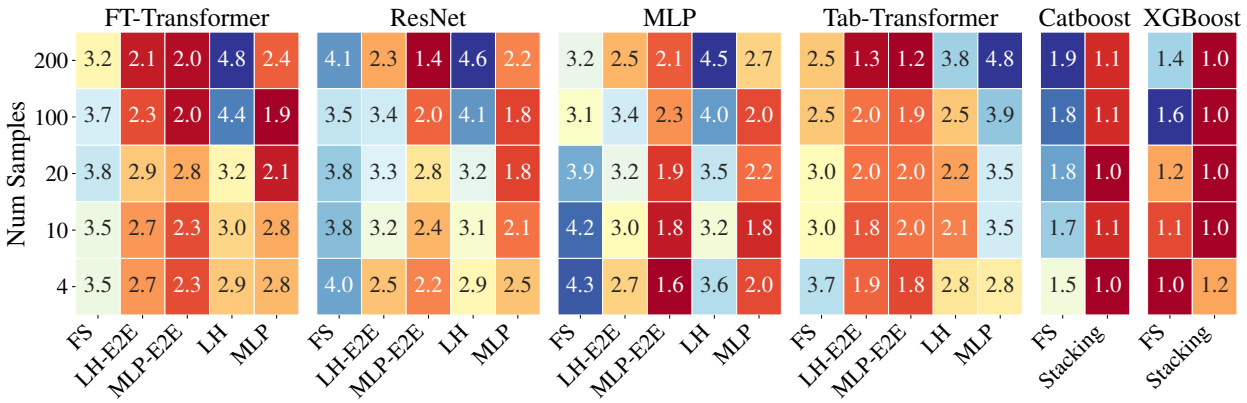


Figure D.2: **Model-wise ranks for GBDT and neural networks.** Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is computed across training setups for each model and is averaged across all downstream tasks.

D.4.3 Average ROC-AUC Improvement over Catboost

In Figure D.3 we display ROC-AUC improvements over Catboost baseline averaged across all downstream tasks. We observe trends similar to ones with ranks; MLP model performs best in low data regimes, while FT-Transformer offers consistent gains across all data levels.

D.4.4 Dataset-level Results

In Tables D.10, D.11, D.12, D.13 we report ROC-AUC measurements for each model on each downstream task. We include the results for GBDT models, neural baselines and neural networks with transfer learning.

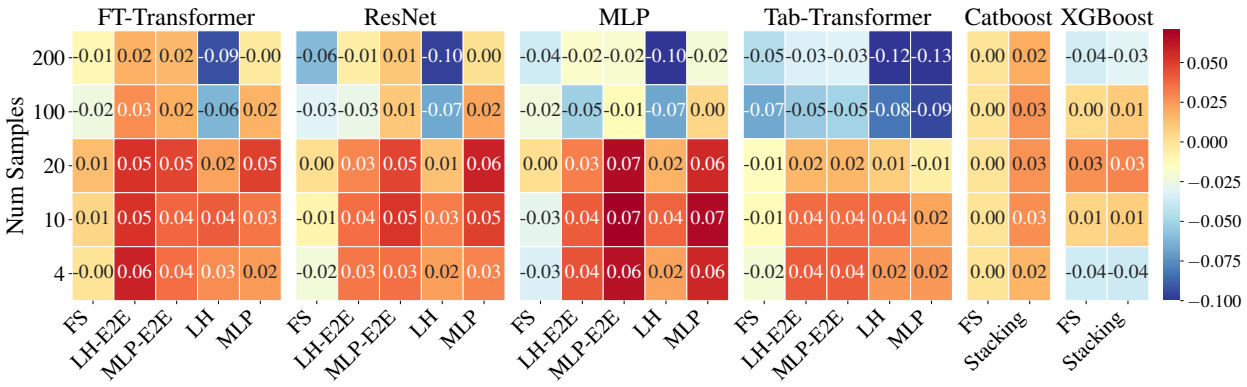


Figure D.3: **Average ROC-AUC improvements over Catboost baseline.** Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. ROC-AUC improvement is computed as difference in ROC-AUC with Catboost model and is averaged across all downstream tasks.

Dataset	**Diabetes**					**Hypertensive**				
Num Samples	4	10	20	100	200	4	10	20	100	200
FT-FS	0.511±0.008	0.540±0.004	0.567±0.004	0.596±0.009	0.802±0.010	0.453±0.010	0.477±0.007	0.528±0.006	0.658±0.008	0.722±0.005
FT-FS-2	0.514±0.008	0.550±0.006	0.570±0.003	0.666±0.006	0.779±0.013	0.464±0.006	0.482±0.007	0.525±0.004	0.666±0.005	0.701±0.004
FT-LH-E2E	0.661±0.007	**0.692**±0.005	**0.710**±0.004	0.770±0.006	0.821±0.003	0.669±0.005	0.713±0.003	0.707±0.002	0.746±0.002	0.760±0.002
FT-MLP-E2EE	0.524±0.011	0.606±0.009	0.654±0.012	0.781±0.007	**0.830**±0.004	0.648±0.009	0.721±0.003	0.709±0.003	**0.759**±0.003	**0.768**±0.002
FT-LH	**0.690**±0.005	**0.689**±0.004	**0.704**±0.005	0.700±0.005	0.702±0.005	0.658±0.008	0.713±0.005	0.708±0.004	0.714±0.003	0.716±0.003
	0.534±0.012	0.598±0.011	0.662±0.011	0.736±0.002	0.755±0.001	0.648±0.012	0.724±0.004	0.725±0.003	**0.760**±0.002	**0.767**±0.001
ResNet-FS	0.498±0.014	0.481±0.011	0.501±0.006	0.627±0.009	0.677±0.006	0.464±0.011	0.492±0.012	0.568±0.004	0.666±0.003	0.689±0.003
ResNet-FS-2	0.484±0.008	0.514±0.005	0.539±0.006	0.677±0.003	0.776±0.002	0.500±0.007	0.494±0.008	0.538±0.006	0.644±0.007	0.724±0.002
ResNet-LH-E2E	0.643±0.003	0.660±0.003	0.686±0.002	0.705±0.002	0.789±0.001	0.684±0.004	0.683±0.002	0.697±0.001	0.681±0.002	0.762±0.001
ResNet-MLP-E2E	0.445±0.008	0.542±0.009	0.616±0.007	0.633±0.009	0.775±0.002	**0.704**±0.009	0.719±0.003	0.716±0.002	0.737±0.003	**0.767**±0.001
ResNet-LH	0.672±0.003	0.649±0.003	0.692±0.002	0.700±0.002	0.709±0.002	0.686±0.004	0.719±0.002	0.719±0.001	0.729±0.001	0.738±0.001
ResNet-MLP	0.457±0.007	0.546±0.009	0.635±0.009	0.720±0.002	0.752±0.001	**0.704**±0.009	0.730±0.002	**0.739**±0.002	0.757±0.001	**0.768**±0.001
MLP-FS	0.503±0.007	0.491±0.005	0.515±0.002	0.652±0.003	0.706±0.002	0.442±0.003	0.470±0.004	0.578±0.001	0.670±0.002	0.699±0.002
MLP-FS-2	0.507±0.005	0.529±0.006	0.543±0.004	0.550±0.012	0.671±0.005	0.500±0.010	0.492±0.011	0.518±0.005	0.638±0.004	0.692±0.002
MLP-LH-E2E	0.675±0.002	0.682±0.002	0.689±0.002	0.698±0.002	0.766±0.002	0.661±0.003	0.674±0.002	0.682±0.001	0.692±0.002	0.753±0.001
MLP-MLP-E2E	0.536±0.005	0.589±0.004	0.626±0.004	0.618±0.008	0.623±0.006	**0.706**±0.006	**0.736**±0.002	0.732±0.002	0.753±0.002	0.753±0.002
MLP-LH	0.685±0.002	0.665±0.002	0.697±0.002	0.701±0.002	0.707±0.002	0.658±0.003	0.711±0.001	0.704±0.002	0.718±0.001	0.724±0.001
MLP-MLP	0.533±0.005	0.590±0.004	0.627±0.003	0.719±0.001	0.744±0.001	**0.706**±0.006	**0.738**±0.002	**0.739**±0.002	**0.762**±0.001	0.764±0.001
Tab-FS	0.488±0.007	0.487±0.002	0.517±0.003	0.616±0.004	0.691±0.004	0.496±0.007	0.465±0.004	0.570±0.004	0.667±0.004	0.701±0.003
Tab-FS-2	0.495±0.007	0.507±0.004	0.541±0.004	0.563±0.012	0.713±0.003	0.513±0.005	0.496±0.011	0.538±0.003	0.564±0.007	0.659±0.004
Tab-LH-E2E	**0.693**±0.006	**0.692**±0.006	0.694±0.005	0.704±0.006	0.727±0.005	0.658±0.004	0.665±0.004	0.685±0.003	0.659±0.004	0.727±0.003
Tab-MLP-E2E	**0.693**±0.006	**0.692**±0.006	0.694±0.005	0.704±0.006	0.725±0.005	0.658±0.004	0.665±0.004	0.685±0.003	0.659±0.004	0.727±0.003
Tab-LH	**0.698**±0.006	**0.691**±0.005	**0.703**±0.005	0.705±0.005	0.705±0.006	0.653±0.005	0.684±0.004	0.678±0.004	0.666±0.004	0.664±0.004
Tab-MLP	**0.698**±0.006	**0.698**±0.006	**0.703**±0.006	0.703±0.006	0.702±0.006	0.653±0.005	0.658±0.004	0.657±0.004	0.656±0.004	0.654±0.004
Catboost	0.495±0.004	0.514±0.008	0.585±0.003	0.782±0.001	0.797±0.001	0.524±0.005	0.501±0.004	0.491±0.001	0.721±0.001	0.738±0.001
Catboost+stacking	0.507±0.004	0.555±0.006	0.623±0.004	**0.807**±0.002	**0.826**±0.001	0.604±0.012	0.602±0.008	0.576±0.014	0.732±0.001	0.756±0.001
XGBoost	0.532±0.006	0.525±0.003	0.593±0.006	**0.805**±0.001	0.824±0.000	0.466±0.003	0.514±0.005	0.514±0.004	0.733±0.001	0.756±0.001
XGBoost+stacking	0.524±0.011	0.529±0.004	0.599±0.005	**0.806**±0.000	**0.826**±0.001	0.458±0.003	0.510±0.004	0.517±0.005	0.740±0.001	0.759±0.000

Table D.10: **ROC-AUC scores for all models for "Diabetes" and "Hypertensive" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Ischematic					Heart				
Num Samples	4	10	20	100	200	4	10	20	100	200
FT-FS	0.620±0.009	0.656±0.004	0.644±0.011	0.678±0.005	0.740±0.006	0.532±0.013	0.557±0.009	0.653±0.005	0.687±0.008	0.746±0.003
FT-FS-2	0.655±0.002	0.656±0.005	0.596±0.011	0.740±0.003	0.759±0.002	0.517±0.005	0.589±0.004	0.571±0.011	0.762±0.002	0.748±0.003
FT-LH-E2E	0.675±0.004	0.664±0.005	0.701±0.002	0.771±0.002	0.786±0.001	0.589±0.002	0.642±0.004	0.667±0.002	0.770±0.002	0.782±0.002
FT-MLP-E2EE	0.702±0.006	0.704±0.008	0.708±0.004	0.781±0.002	0.807±0.001	0.542±0.008	0.642±0.007	0.658±0.002	0.768±0.005	0.784±0.003
FT-LH	0.637±0.003	0.679±0.002	0.688±0.001	0.716±0.002	0.726±0.002	**0.626**±0.003	0.664±0.002	0.672±0.002	0.689±0.003	0.713±0.002
FT-MLP	0.691±0.006	0.707±0.007	0.740±0.003	0.792±0.001	0.813±0.001	0.516±0.016	0.640±0.014	0.667±0.005	**0.777**±0.001	**0.790**±0.001
ResNet-FS	0.608±0.008	0.622±0.006	0.631±0.009	0.707±0.003	0.726±0.003	0.489±0.028	0.532±0.012	0.644±0.006	0.683±0.004	0.730±0.005
ResNet-FS-2	0.627±0.006	0.646±0.005	0.631±0.003	0.736±0.002	0.752±0.002	0.497±0.020	0.551±0.010	0.650±0.004	0.750±0.004	0.772±0.002
ResNet-LH-E2E	0.631±0.003	0.663±0.002	0.687±0.003	0.767±0.001	0.778±0.001	0.587±0.005	0.609±0.004	0.644±0.002	0.656±0.002	0.760±0.002
ResNet-MLP-E2E	0.677±0.008	0.724±0.006	0.715±0.005	0.786±0.003	0.814±0.001	0.497±0.011	0.618±0.008	0.641±0.004	0.761±0.002	**0.789**±0.001
ResNet-LH	0.581±0.003	0.636±0.003	0.678±0.002	0.707±0.002	0.712±0.003	0.612±0.005	0.657±0.004	0.668±0.002	0.661±0.003	0.677±0.003
ResNet-MLP	0.663±0.009	0.715±0.007	0.752±0.004	**0.799**±0.001	**0.818**±0.001	0.488±0.012	0.627±0.008	**0.683**±0.003	0.768±0.002	0.783±0.001
MLP-FS	0.648±0.004	0.643±0.008	0.626±0.002	0.728±0.002	0.760±0.001	0.426±0.007	0.512±0.019	0.646±0.002	0.710±0.004	0.765±0.001
MLP-FS-2	0.639±0.004	0.667±0.005	0.668±0.003	0.729±0.003	0.761±0.003	0.570±0.010	0.643±0.006	0.669±0.003	0.756±0.002	0.751±0.001
MLP-LH-E2E	0.657±0.003	0.662±0.003	0.689±0.002	0.726±0.002	0.785±0.001	**0.626**±0.003	0.644±0.003	0.665±0.002	0.677±0.002	0.766±0.001
MLP-MLP-E2E	**0.713**±0.004	**0.745**±0.004	**0.768**±0.002	0.788±0.001	0.811±0.000	**0.629**±0.007	**0.692**±0.003	**0.685**±0.002	0.719±0.003	0.780±0.001
MLP-LH	0.626±0.004	0.671±0.002	0.675±0.002	0.696±0.003	0.699±0.003	0.610±0.003	0.650±0.002	0.656±0.002	0.665±0.003	0.679±0.003
MLP-MLP	**0.715**±0.005	**0.738**±0.004	0.758±0.002	0.789±0.001	0.807±0.001	**0.620**±0.006	0.681±0.003	**0.684**±0.002	0.755±0.002	0.767±0.001
Tab-FS	0.461±0.006	0.659±0.003	0.620±0.004	0.711±0.002	0.739±0.002	0.501±0.009	0.545±0.002	0.645±0.003	0.731±0.003	0.741±0.004
Tab-FS-2	0.472±0.003	0.632±0.011	0.571±0.006	0.721±0.001	0.746±0.001	0.517±0.006	0.565±0.006	0.614±0.004	0.709±0.006	0.736±0.004
Tab-LH-E2E	0.664±0.005	0.668±0.005	0.675±0.004	0.708±0.004	0.772±0.002	0.602±0.006	0.613±0.005	0.636±0.004	0.661±0.006	0.760±0.004
Tab-MLP-E2E	0.664±0.005	0.668±0.005	0.675±0.004	0.706±0.004	0.770±0.002	0.602±0.006	0.613±0.005	0.636±0.004	0.705±0.005	0.757±0.004
Tab-LH	0.625±0.006	0.652±0.007	0.655±0.006	0.668±0.006	0.664±0.006	0.599±0.007	0.619±0.006	0.621±0.006	0.631±0.006	0.636±0.007
Tab-MLP	0.625±0.006	0.624±0.006	0.628±0.006	0.642±0.006	0.639±0.006	0.599±0.007	0.595±0.008	0.600±0.007	0.616±0.007	0.617±0.007
Catboost	0.657±0.004	0.641±0.005	0.651±0.003	0.735±0.001	0.734±0.001	0.554±0.005	0.581±0.004	0.652±0.001	0.712±0.001	0.744±0.001
Catboost+stacking	0.671±0.004	0.688±0.010	0.734±0.004	0.768±0.002	0.766±0.002	0.555±0.005	0.624±0.009	0.674±0.001	0.756±0.001	0.777±0.001
XGBoost	0.619±0.006	0.656±0.003	0.699±0.003	0.747±0.001	0.752±0.000	0.470±0.016	0.564±0.004	0.646±0.005	0.735±0.001	0.757±0.001
XGBoost+stacking	0.628±0.007	0.659±0.003	0.722±0.004	0.770±0.001	0.776±0.001	0.458±0.014	0.567±0.004	0.649±0.004	0.742±0.001	0.767±0.001

Table D.11: **ROC-AUC scores for all models for "Ischematic" and "Heart" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Overweight					Anemia				
Num Samples	4	10	20	100	200	4	10	20	100	200
FT-FS	0.516±0.012	0.506±0.003	0.623±0.006	0.582±0.015	**0.822**±0.009	0.643±0.013	0.709±0.003	0.741±0.006	0.764±0.003	0.722±0.007
FT-FS-2	0.491±0.008	0.490±0.009	0.610±0.004	0.638±0.005	0.679±0.004	**0.728**±0.006	0.749±0.002	0.735±0.006	0.782±0.001	**0.794**±0.002
FT-LH-E2E	0.673±0.004	0.620±0.008	**0.728**±0.012	**0.759**±0.015	**0.818**±0.004	0.639±0.003	0.646±0.003	0.665±0.004	0.756±0.001	0.762±0.001
FT-MLP-E2EE	0.540±0.015	0.469±0.011	0.581±0.024	0.720±0.018	0.768±0.013	0.662±0.009	0.682±0.004	0.725±0.003	0.750±0.002	0.773±0.001
FT-LH	0.669±0.002	0.675±0.003	0.682±0.002	0.675±0.002	0.674±0.002	0.557±0.003	0.580±0.003	0.597±0.004	0.610±0.003	0.617±0.003
FT-MLP	0.532±0.015	0.469±0.016	0.507±0.016	0.656±0.006	0.718±0.004	0.610±0.013	0.642±0.006	0.687±0.003	0.705±0.003	0.712±0.003
ResNet-FS	0.515±0.015	0.502±0.011	0.642±0.006	0.642±0.008	0.665±0.007	0.553±0.019	0.653±0.009	0.698±0.004	0.750±0.003	0.767±0.002
ResNet-FS-2	0.495±0.009	0.504±0.008	0.615±0.004	0.666±0.004	0.739±0.003	0.644±0.019	0.752±0.004	0.747±0.002	0.757±0.002	0.780±0.001
ResNet-LH-E2E	**0.699**±0.003	**0.691**±0.003	0.700±0.002	0.714±0.001	0.770±0.002	0.607±0.003	0.652±0.003	0.665±0.002	0.736±0.001	0.748±0.001
ResNet-MLP-E2E	0.547±0.008	0.545±0.011	0.568±0.013	0.674±0.009	0.765±0.005	0.608±0.009	0.697±0.008	0.727±0.004	0.760±0.001	0.767±0.002
ResNet-LH	**0.698**±0.003	0.683±0.003	0.679±0.002	0.687±0.001	0.685±0.001	0.563±0.004	0.593±0.004	0.594±0.003	0.635±0.003	0.639±0.003
ResNet-MLP	0.548±0.009	0.544±0.012	0.525±0.010	0.669±0.004	0.726±0.002	0.582±0.009	0.666±0.011	0.710±0.004	0.739±0.002	0.739±0.003
MLP-FS	0.533±0.004	0.499±0.008	0.635±0.003	0.651±0.005	0.698±0.002	0.641±0.006	0.689±0.002	0.708±0.000	0.763±0.002	**0.796**±0.001
MLP-FS-2	0.528±0.008	0.489±0.005	0.573±0.003	0.566±0.006	0.646±0.003	**0.733**±0.005	0.753±0.003	0.738±0.001	0.768±0.001	0.790±0.001
MLP-LH-E2E	0.680±0.002	0.675±0.002	0.697±0.002	0.683±0.002	0.718±0.002	0.610±0.003	0.615±0.003	0.657±0.002	0.706±0.002	0.749±0.001
MLP-MLP-E2E	0.599±0.005	0.574±0.008	0.609±0.004	0.614±0.004	0.626±0.003	0.690±0.004	0.695±0.004	0.723±0.002	0.757±0.001	0.758±0.001
MLP-LH	0.675±0.002	0.688±0.001	0.685±0.001	0.685±0.002	0.686±0.002	0.551±0.004	0.567±0.004	0.592±0.004	0.597±0.003	0.600±0.004
MLP-MLP	0.597±0.005	0.576±0.008	0.582±0.004	0.646±0.003	0.669±0.003	0.655±0.005	0.666±0.004	0.694±0.003	0.703±0.003	0.699±0.002
Tab-FS	0.510±0.012	0.506±0.003	0.582±0.005	0.578±0.002	0.654±0.005	0.548±0.009	0.677±0.003	0.708±0.003	0.762±0.002	0.770±0.001
Tab-FS-2	0.518±0.009	0.499±0.007	0.547±0.004	0.564±0.012	0.612±0.008	0.602±0.004	0.689±0.006	0.729±0.002	0.727±0.005	0.779±0.001
Tab-LH-E2E	0.676±0.004	0.679±0.004	0.684±0.004	0.684±0.003	0.726±0.003	0.594±0.004	0.603±0.004	0.650±0.003	0.701±0.004	0.742±0.003
Tab-MLP-E2E	0.676±0.004	0.679±0.004	0.684±0.004	0.684±0.003	0.733±0.003	0.594±0.005	0.603±0.004	0.650±0.003	0.698±0.004	0.745±0.003
Tab-LH	0.673±0.004	0.681±0.004	0.681±0.004	0.679±0.004	0.679±0.004	0.561±0.005	0.567±0.006	0.591±0.004	0.592±0.004	0.590±0.003
Tab-MLP	0.673±0.004	0.674±0.004	0.674±0.004	0.676±0.004	0.676±0.004	0.561±0.005	0.561±0.006	0.572±0.004	0.576±0.004	0.574±0.004
Catboost	0.605±0.007	0.499±0.008	0.625±0.004	0.764±0.002	0.765±0.001	0.560±0.009	**0.765**±0.003	0.760±0.001	0.775±0.001	0.778±0.000
Catboost+stacking	0.614±0.006	0.539±0.007	0.672±0.005	**0.792**±0.002	0.797±0.001	0.557±0.009	0.754±0.003	0.763±0.001	0.781±0.001	0.783±0.000
XGBoost	0.500±0.005	0.540±0.004	**0.715**±0.010	0.789±0.001	0.766±0.001	0.527±0.015	0.754±0.001	**0.773**±0.002	**0.788**±0.000	0.783±0.000
XGBoost+stacking	0.511±0.008	0.538±0.005	**0.711**±0.009	**0.789**±0.001	0.767±0.002	0.504±0.009	0.754±0.002	**0.772**±0.002	**0.788**±0.000	0.783±0.000

5pt

Table D.12: **ROC-AUC scores for all models for "Overweight" and "Anemia" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Respiratory					Hypotension				
Num Samples	4	10	20	100	200	4	10	20	100	200
FT-FS	0.500±0.009	0.475±0.005	0.514±0.006	0.572±0.009	0.561±0.004	0.526±0.008	0.518±0.009	0.546±0.006	0.601±0.009	0.644±0.002
FT-FS-2	0.495±0.005	0.469±0.005	0.484±0.006	0.562±0.010	0.577±0.003	0.574±0.003	0.552±0.005	0.539±0.012	0.546±0.011	0.641±0.001
FT-LH-E2E	0.545±0.004	0.498±0.004	0.517±0.002	0.578±0.002	0.585±0.001	0.564±0.002	0.564±0.003	0.584±0.002	0.592±0.002	0.635±0.001
FT-MLP-E2EE	0.531±0.004	0.462±0.007	0.496±0.006	0.576±0.004	0.562±0.004	0.595±0.002	0.595±0.004	0.612±0.001	0.627±0.002	0.647±0.002
FT-LH	0.556±0.002	0.536±0.002	0.543±0.002	0.565±0.002	0.564±0.002	0.511±0.002	0.533±0.002	0.545±0.002	0.538±0.002	0.553±0.002
FT-MLP	0.547±0.004	0.490±0.008	0.518±0.004	0.575±0.002	0.569±0.001	0.590±0.003	0.593±0.007	0.604±0.002	0.612±0.001	0.637±0.001
ResNet-FS	0.499±0.007	0.500±0.004	0.520±0.010	0.564±0.004	0.550±0.004	0.516±0.008	0.502±0.011	0.506±0.011	0.572±0.006	0.578±0.011
ResNet-FS-2	0.479±0.008	0.460±0.005	0.465±0.004	0.565±0.003	0.586±0.003	0.552±0.010	0.535±0.006	0.539±0.004	0.589±0.004	0.638±0.002
ResNet-LH-E2E	0.536±0.003	0.530±0.002	0.517±0.002	0.577±0.001	0.581±0.001	0.521±0.005	0.535±0.004	0.531±0.003	0.547±0.002	0.590±0.001
ResNet-MLP-E2E	0.524±0.006	0.521±0.003	0.511±0.003	0.583±0.003	0.578±0.002	**0.614**±0.004	0.603±0.004	0.601±0.004	0.624±0.003	0.638±0.001
ResNet-LH	0.552±0.002	**0.551**±0.001	**0.550**±0.001	0.570±0.001	0.565±0.001	0.503±0.004	0.543±0.004	0.527±0.003	0.524±0.003	0.535±0.003
ResNet-MLP	0.520±0.006	0.532±0.003	0.530±0.003	0.573±0.001	0.566±0.001	**0.616**±0.004	0.603±0.004	0.621±0.002	0.626±0.001	0.649±0.001
MLP-FS	0.492±0.006	0.494±0.004	0.503±0.006	0.564±0.003	0.576±0.002	0.504±0.004	0.513±0.005	0.529±0.005	0.621±0.001	0.648±0.002
MLP-FS-2	0.500±0.003	0.447±0.004	0.458±0.003	0.542±0.004	0.574±0.002	0.568±0.003	0.545±0.005	0.587±0.002	0.571±0.007	**0.652**±0.001
MLP-LH-E2E	0.545±0.002	0.535±0.001	0.531±0.001	0.578±0.001	**0.591**±0.001	0.532±0.002	0.530±0.002	0.560±0.002	0.563±0.001	0.595±0.001
MLP-MLP-E2E	**0.564**±0.003	0.533±0.005	0.535±0.003	0.583±0.002	0.583±0.002	**0.608**±0.004	**0.628**±0.002	**0.633**±0.001	**0.635**±0.001	**0.654**±0.000
MLP-LH	0.551±0.002	0.544±0.002	**0.553**±0.001	0.567±0.002	0.567±0.002	0.513±0.003	0.550±0.002	0.557±0.002	0.548±0.002	0.555±0.002
MLP-MLP	**0.571**±0.003	**0.545**±0.005	**0.554**±0.003	0.577±0.001	0.575±0.002	**0.611**±0.004	**0.629**±0.002	**0.631**±0.001	0.628±0.001	0.645±0.001
Tab-FS	0.463±0.002	0.459±0.003	0.472±0.004	0.525±0.010	0.567±0.003	0.547±0.004	0.535±0.006	0.544±0.007	0.561±0.008	0.635±0.001
Tab-FS-2	0.461±0.004	0.454±0.003	0.458±0.002	0.494±0.008	0.526±0.003	0.541±0.004	0.525±0.006	0.540±0.002	0.566±0.008	0.632±0.001
Tab-LH-E2E	0.531±0.004	0.522±0.004	0.504±0.003	0.551±0.004	0.582±0.002	0.526±0.004	0.523±0.004	0.540±0.003	0.544±0.003	0.577±0.004
Tab-MLP-E2E	0.531±0.004	0.522±0.004	0.504±0.003	0.552±0.004	0.582±0.002	0.526±0.004	0.524±0.004	0.540±0.003	0.544±0.003	0.572±0.004
Tab-LH	0.550±0.005	**0.548**±0.004	**0.551**±0.004	0.562±0.004	0.562±0.004	0.511±0.004	0.534±0.004	0.537±0.004	0.533±0.004	0.537±0.004
Tab-MLP	0.550±0.005	**0.547**±0.005	**0.552**±0.004	0.558±0.004	0.558±0.004	0.511±0.004	0.512±0.004	0.520±0.004	0.523±0.004	0.526±0.004
Catboost	0.482±0.001	0.492±0.003	0.496±0.003	0.554±0.002	0.572±0.002	0.523±0.001	0.490±0.004	0.531±0.005	0.552±0.001	0.635±0.001
Catboost+stacking	0.488±0.003	0.498±0.002	0.503±0.003	**0.592**±0.002	0.584±0.001	0.568±0.002	0.501±0.004	0.545±0.004	0.583±0.001	0.643±0.001
XGBoost	0.493±0.004	0.493±0.003	0.494±0.001	0.571±0.001	0.496±0.003	0.507±0.004	0.509±0.002	0.553±0.001	0.539±0.001	0.517±0.009
XGBoost+stacking	0.499±0.005	0.495±0.004	0.497±0.002	0.578±0.001	0.499±0.006	0.507±0.006	0.511±0.003	0.553±0.002	0.543±0.001	0.516±0.008

Table D.13: **ROC-AUC scores for all models for "Respiratory" and "Hypotension" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Lipoid					Atrial				
Num Samples	4	10	20	100	200	4	10	20	100	200
FT-FS	0.502±0.006	0.552±0.011	0.564±0.006	0.640±0.012	0.675±0.004	0.603±0.008	0.594±0.015	0.617±0.007	0.702±0.008	0.690±0.008
FT-FS-2	0.520±0.004	0.530±0.002	0.574±0.003	0.598±0.003	0.660±0.004	0.606±0.003	0.628±0.003	0.382±0.005	0.665±0.005	0.703±0.001
FT-LH-E2E	0.584±0.006	0.565±0.008	0.630±0.004	0.727±0.002	**0.745**±0.002	0.579±0.005	0.603±0.005	0.604±0.004	0.725±0.002	**0.788**±0.002
FT-MLP-E2EE	0.429±0.009	0.475±0.012	0.633±0.003	0.714±0.004	0.737±0.001	0.605±0.009	0.627±0.006	0.636±0.005	0.753±0.002	0.752±0.002
FT-LH	**0.631**±0.004	0.582±0.006	0.655±0.003	0.707±0.001	0.724±0.001	0.543±0.003	0.568±0.003	0.577±0.002	0.610±0.002	0.612±0.002
FT-MLP	0.415±0.009	0.531±0.014	0.689±0.002	0.735±0.002	0.740±0.001	0.587±0.015	0.637±0.016	**0.667**±0.007	0.766±0.002	0.772±0.001
ResNet-FS	0.503±0.007	0.574±0.003	0.581±0.004	0.595±0.004	0.649±0.002	0.595±0.007	0.570±0.009	0.599±0.006	0.691±0.004	0.625±0.019
ResNet-FS-2	0.505±0.009	0.551±0.007	0.574±0.003	0.523±0.018	0.664±0.003	0.585±0.010	0.573±0.009	0.629±0.005	0.705±0.002	0.749±0.002
ResNet-LH-E2E	0.561±0.004	0.571±0.005	0.610±0.004	0.635±0.002	0.725±0.001	0.609±0.006	0.523±0.004	0.587±0.002	0.681±0.001	0.758±0.001
ResNet-MLP-E2E	0.442±0.006	0.545±0.010	0.657±0.005	0.720±0.003	**0.744**±0.001	**0.671**±0.013	0.627±0.010	0.650±0.005	0.732±0.005	0.748±0.005
ResNet-LH	0.563±0.005	0.552±0.007	0.620±0.004	0.697±0.002	0.714±0.002	0.582±0.007	0.557±0.003	0.570±0.002	0.587±0.002	0.595±0.002
ResNet-MLP	0.442±0.008	0.551±0.014	0.684±0.005	**0.742**±0.001	**0.746**±0.001	**0.669**±0.014	0.625±0.009	**0.677**±0.004	**0.770**±0.002	0.766±0.002
MLP-FS	0.507±0.006	0.531±0.015	0.574±0.005	0.598±0.004	0.559±0.034	0.535±0.015	0.529±0.010	0.603±0.005	0.704±0.001	0.707±0.001
MLP-FS-2	0.510±0.005	0.577±0.005	0.574±0.002	0.633±0.003	0.665±0.003	0.588±0.004	0.594±0.008	0.635±0.003	0.577±0.010	0.726±0.001
MLP-LH-E2E	0.602±0.003	**0.607**±0.003	0.624±0.002	0.624±0.003	0.713±0.001	0.563±0.004	0.584±0.004	0.606±0.003	0.646±0.003	0.733±0.002
MLP-MLP-E2E	0.430±0.009	0.592±0.007	0.684±0.004	0.708±0.003	0.741±0.001	**0.662**±0.007	**0.689**±0.003	**0.681**±0.001	0.714±0.001	0.753±0.001
MLP-LH	0.608±0.003	0.577±0.004	0.644±0.003	0.696±0.001	0.707±0.001	0.535±0.005	0.595±0.004	0.588±0.004	0.598±0.004	0.593±0.004
MLP-MLP	0.429±0.009	0.594±0.007	**0.701**±0.004	**0.740**±0.001	0.743±0.001	**0.653**±0.007	**0.682**±0.003	0.679±0.002	0.745±0.001	0.740±0.002
Tab-FS	0.536±0.002	0.510±0.005	0.505±0.003	0.538±0.007	0.656±0.002	0.593±0.007	0.563±0.007	0.608±0.009	0.608±0.011	0.696±0.003
Tab-FS-2	0.537±0.003	0.489±0.005	0.487±0.002	0.513±0.009	0.561±0.009	0.530±0.004	0.547±0.004	0.583±0.002	0.688±0.002	0.692±0.002
Tab-LH-E2E	0.615±0.005	**0.614**±0.006	0.611±0.006	0.625±0.006	0.670±0.007	0.561±0.008	0.573±0.007	0.591±0.005	0.607±0.008	0.716±0.005
Tab-MLP-E2E	0.615±0.005	**0.614**±0.006	0.611±0.006	0.625±0.006	0.668±0.007	0.561±0.008	0.573±0.007	0.591±0.005	0.607±0.008	0.720±0.005
Tab-LH	0.617±0.006	0.595±0.006	0.619±0.007	0.647±0.008	0.651±0.008	0.540±0.008	0.578±0.008	0.575±0.007	0.587±0.008	0.584±0.008
Tab-MLP	0.617±0.006	**0.618**±0.006	0.621±0.006	0.634±0.007	0.635±0.007	0.540±0.008	0.555±0.008	0.557±0.008	0.568±0.008	0.567±0.008
Catboost	0.523±0.014	0.536±0.002	0.547±0.005	0.618±0.006	0.693±0.001	0.598±0.001	0.588±0.007	0.541±0.004	0.624±0.001	0.705±0.002
Catboost+stacking	0.515±0.017	0.536±0.002	0.555±0.007	0.669±0.005	0.718±0.002	0.610±0.001	0.610±0.007	0.555±0.005	0.678±0.001	0.749±0.002
XGBoost	0.390±0.008	0.483±0.004	0.539±0.001	0.558±0.002	0.695±0.001	0.573±0.009	0.632±0.002	0.586±0.001	0.626±0.001	0.583±0.001
XGBoost+stacking	0.394±0.006	0.487±0.005	0.545±0.002	0.576±0.001	0.723±0.001	0.584±0.007	0.636±0.004	0.584±0.002	0.627±0.001	0.574±0.008

Table D.14: **ROC-AUC scores for all models for "Lipoid" and "Atrial" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Purpura					Alcohol				
Num Samples	4	10	20	100	200	4	10	20	100	200
FT-FS	0.711±0.007	0.703±0.010	0.737±0.005	**0.870**±0.007	0.851±0.009	0.593±0.008	0.570±0.010	0.600±0.005	0.666±0.006	0.640±0.005
FT-FS-2	**0.763**±0.003	**0.745**±0.003	0.451±0.012	0.778±0.004	0.852±0.006	0.603±0.004	0.584±0.003	0.588±0.007	0.635±0.006	0.633±0.005
FT-LH-E2E	0.643±0.002	0.641±0.002	0.664±0.004	0.784±0.005	0.777±0.005	0.585±0.005	0.573±0.004	0.616±0.005	0.667±0.003	0.700±0.005
FT-MLP-E2EE	0.705±0.007	0.672±0.006	0.695±0.005	0.629±0.014	0.798±0.003	0.658±0.007	0.589±0.006	0.622±0.003	0.677±0.004	0.700±0.002
FT-LH	0.511±0.002	0.538±0.002	0.543±0.002	0.561±0.004	0.552±0.004	0.462±0.003	0.520±0.003	0.513±0.004	0.477±0.004	0.475±0.004
FT-MLP	0.647±0.010	0.563±0.014	0.615±0.008	0.715±0.004	0.723±0.003	0.685±0.005	**0.606**±0.011	**0.666**±0.006	0.685±0.003	0.703±0.003
ResNet-FS	0.643±0.007	0.708±0.007	0.689±0.022	0.755±0.007	0.659±0.021	0.560±0.015	0.550±0.004	0.611±0.009	0.680±0.006	0.700±0.007
ResNet-FS-2	0.684±0.010	0.702±0.006	0.730±0.004	0.769±0.004	0.775±0.003	0.567±0.011	0.573±0.006	0.616±0.006	**0.702**±0.003	**0.745**±0.003
ResNet-LH-E2E	0.562±0.006	0.599±0.004	0.618±0.002	0.687±0.002	0.715±0.002	0.473±0.004	0.504±0.003	0.576±0.002	0.603±0.003	0.704±0.003
ResNet-MLP-E2E	0.687±0.009	0.652±0.011	0.697±0.006	0.729±0.005	0.761±0.004	**0.715**±0.007	**0.621**±0.009	0.630±0.005	0.689±0.003	0.720±0.002
ResNet-LH	0.533±0.008	0.560±0.005	0.535±0.003	0.542±0.004	0.546±0.004	0.436±0.005	0.471±0.005	0.488±0.005	0.464±0.003	0.463±0.003
ResNet-MLP	0.665±0.011	0.617±0.014	0.659±0.008	0.740±0.003	0.751±0.003	**0.719**±0.008	**0.617**±0.009	0.649±0.003	0.661±0.002	0.689±0.002
MLP-FS	0.559±0.026	0.570±0.022	0.689±0.007	0.754±0.002	0.749±0.003	0.518±0.013	0.504±0.011	0.584±0.009	0.618±0.002	0.632±0.003
MLP-FS-2	**0.764**±0.003	0.696±0.006	0.733±0.003	0.716±0.006	0.750±0.002	0.514±0.008	0.505±0.006	0.531±0.005	0.626±0.003	0.556±0.005
MLP-LH-E2E	0.594±0.003	0.595±0.002	0.642±0.003	0.595±0.003	0.723±0.002	0.483±0.003	0.473±0.002	0.496±0.002	0.492±0.002	0.639±0.002
MLP-MLP-E2E	0.699±0.004	0.688±0.003	0.725±0.002	0.692±0.002	0.772±0.001	0.639±0.008	0.517±0.006	0.555±0.004	0.581±0.004	0.658±0.003
MLP-LH	0.503±0.003	0.561±0.003	0.564±0.004	0.565±0.004	0.563±0.004	0.460±0.004	0.473±0.002	0.459±0.002	0.457±0.002	0.454±0.003
MLP-MLP	0.660±0.005	0.647±0.003	0.662±0.003	0.728±0.002	0.738±0.002	0.642±0.008	0.501±0.007	0.523±0.006	0.580±0.006	0.590±0.006
Tab-FS	0.688±0.007	0.697±0.007	0.672±0.014	0.673±0.016	0.672±0.006	0.626±0.004	0.536±0.006	0.552±0.016	0.536±0.012	0.672±0.003
Tab-FS-2	0.649±0.003	0.681±0.009	0.661±0.004	0.654±0.003	0.723±0.003	0.623±0.007	**0.573**±0.019	0.459±0.003	0.558±0.025	0.666±0.005
Tab-LH-E2E	0.572±0.005	0.579±0.005	0.617±0.004	0.599±0.006	0.680±0.005	0.506±0.007	0.495±0.006	0.476±0.005	0.614±0.006	0.666±0.006
Tab-MLP-E2E	0.572±0.005	0.579±0.005	0.617±0.004	0.599±0.006	0.687±0.005	0.506±0.007	0.495±0.006	0.476±0.005	0.613±0.006	0.668±0.006
Tab-LH	0.508±0.005	0.565±0.006	0.562±0.005	0.557±0.005	0.552±0.006	0.473±0.008	0.502±0.009	0.489±0.009	0.494±0.009	0.489±0.008
Tab-MLP	0.508±0.005	0.529±0.005	0.535±0.005	0.537±0.005	0.534±0.005	0.473±0.008	0.474±0.008	0.467±0.007	0.484±0.008	0.478±0.008
Catboost	0.651±0.002	0.618±0.005	0.677±0.006	0.834±0.003	0.875±0.001	0.541±0.002	0.568±0.004	0.611±0.004	0.640±0.002	0.706±0.001
Catboost+stacking	0.681±0.003	0.641±0.005	0.680±0.009	0.836±0.003	**0.878**±0.001	0.545±0.002	0.566±0.007	0.599±0.005	0.625±0.003	0.692±0.001
XGBoost	0.572±0.006	0.646±0.002	**0.762**±0.003	0.804±0.001	**0.877**±0.003	0.618±0.006	0.548±0.001	0.613±0.002	0.659±0.001	0.502±0.002
XGBoost+stacking	0.568±0.004	0.652±0.002	**0.763**±0.003	0.802±0.001	**0.878**±0.003	0.623±0.004	0.546±0.001	0.613±0.002	0.659±0.001	0.505±0.003

Table D.15: **ROC-AUC scores for all models for "Purpura" and "Alcohol" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Appendix E

IMPROVING PATIENT-SPECIFIC QUALITY ASSURANCE: RADIOTHERAPY PLAN FAILURE PREDICTION WITH DEEP TABULAR MODELS

E.1 Hyperparameter Search Spaces

FT-Transformer

The number of heads is always set to 8.

Table E.1: **Optuna hyperparameter search space and default configuration for FT-Transformer**

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 4]$	3
Feature embedding size	UniformInt$[64, 512]$	192
Residual dropout	$\{0$, Uniform$[0, 0.2]\}$	0.0
Attention dropout	Uniform$[0, 0.5]$	0.2
FFN dropout	Uniform$[0, 0.5]$	0.1
FFN factor	Uniform$[2/3, 8/3]$	$4/3$
Learning rate	LogUniform$[1e-5, 1e-3]$	$1e-4$
Weight decay	LogUniform$[1e-6, 1e-3]$	$1e-5$

Catboost

The hyperparameter search space and distributions are presented in Table E.2. For default configuration we use default parameters from the Catboost library [162].

Table E.2: **Optuna hyperparameter search space for Catboost**

Parameter	Search Space
Max depth	UniformInt$[3, 10]$
Learning rate	LogUniform$[1e-5, 1]$
Bagging temperature	Uniform$[0, 1]$
L2 leaf reg	LogUniform$[1, 10]$
Leaf estimation iterations	UniformInt$[1, 10]$

XGBoost

The hyperparameter search space and distributions as well as the default configuration are presented in Table E.3. For default configuration we use default parameters from the XGBoost library [32].

Table E.3: **Optuna hyperparameter search space for XGBoost**

Parameter	Search Space
Max depth	UniformInt$[3, 10]$
Min child weight	LogUniform$[1e-8, 1e5]$
Subsample	Uniform$[0.5, 1]$
Learning rate	LogUniform$[1e-5, 1]$
Col sample by level	Uniform$[0.5, 1]$
Col sample by tree	Uniform$[0.5, 1]$
Gamma	$\{0$, LogUniform$[1e-8, 1e2]\}$
Lambda	$\{0$, LogUniform$[1e-8, 1e2]\}$
Alpha	$\{0$, LogUniform$[1e-8, 1e2]\}$