Explainable Machine Learning and Applications in Protein-Ligand Complex Structure Prediction

Pascal Sturmfels

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

Doctor of Philosophy

University of Washington

2024

Reading Committee:

David Baker, Chair

Sheng Wang

Frank Dimaio

Program Authorized to Offer Degree:

Computer Science & Engineering

University of Washington

Abstract

Explainable Machine Learning and Applications in Protein-Ligand Complex Structure Prediction

Pascal Sturmfels

Chair of the Supervisory Committee:
David Baker
Biochemistry

This thesis touches upon two main topics: interpreting machine learning models, and the application of machine learning to protein sequences and structures. The first portion deals with feature attribution techniques, which attribute attribution scores on a per-instance basis to a machine learning model that represent that model locally around that instance as linear. Three methods are proposed: expected gradients, attribution priors, and integrated hessians, that extend interpretability beyond feature attribution towards feature interaction and training more interpretable models. The second portion deals with training protein language models and how best to design semi-supervised pre-training tasks. It takes inspiration from multiple sequence alignments to propose two tasks - profile prediction and seq2msa - that extend language modeling beyond autoregressive and masked language modeling. The third portion deals applications in protein structure prediction, chiefly with predicting the structure of proteins in concert with small molecules. Jointly determining both the structure of a protein from its sequence input and how small molecule binding partners dock to that structure remains an open and challenging problem, and has applications in biological discovery, virtual screening, and de-novo design. This thesis discusses the development of a structure prediction network, RoseTTAFold All-Atom, capable of simultaneous folding and docking, as well as some applications enabled by that existing network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

There are so many that contributed to the work in this thesis, and the following list is, at best, a small attempt to acknowledge direct contributors and co-authors, as well as mentor figures, that I encountered during my PhD.

I want to start by mentioning my advisor, David Baker. David had the grace to let me join his lab despite already being in my 4th year and knowing very little about protein design. Although I was only at the Institute for Protein design for two years - less, surely, than most every other PhD student - I've grown tremendously as both a researcher and as a scientist in my time here. Not only have I gained a wealth of structural biology knowledge through discussions with all of my peers, I have also learned more and more about what it means to ask good scientific questions. I attribute that fact directly to David's mentorship. David never minced his words with me, but at every step of the way he made sure I was becoming better and better equipped to ask meaningful scientific questions.

Two other members in my committee deserve mention: Frank Dimaio and Sheng Wang. I've never met a PI more in the trenches than Frank - all throughout the process of developing the work in Chapter 4 he was right alongside us, coding and making PRs late into the night. Although I never worked with Sheng directly, I had the fortune of TA'ing for him during a difficult period in my PhD, and thank him wholeheartedly for his help during that time.

The final member of my committee, and my graduate student representative, Rekha Thomas, deserves special mention. She has served as a both a mentor figure to me and a source of constant inspiration. I hope one day to be able to be the kind of mentor that she is - the kind of mentor that embodies kindness, patience, and understanding.

Before I joined the Baker Lab, I worked for several years with Su-In Lee. I was a difficult PhD

student to mentor at that time - constantly bouncing from one idea to the next with little focus, and plagued in my 3rd and 4th year with health problems that caused me to take most of those two years off. Nonetheless, Su-In was patient with me and supported me through developing the work I did in Chapter 2, for which I am grateful.

I did two internships during the course of my PhD, and had several wonderful mentors throughout those internships. Jesse Vig, Nazneen Rajani and Ali Madani made excellent mentors, despite it being the first year of pandemic and our interactions being completely virtual. Alex Rives and Adam Lehrer also made great mentors - along with all of the other researchers I met at FAIR, including Brian Hie, Halil Alkin, Robert Verkuil, Zeming Lin, Roshan Rao and many others.

I had many amazing peers and co-authors during my PhD. Thanks to Scott Lundberg, who was one of my first mentors when I joined the program. I remember many white board sessions with him leading to some of my earliest ideas about interpretability and subsequently my first few papers published as a PhD student. Nicasia Beebe-Wang, Gabe Erion, Hugh Chen and Ian Covert were all people I collaborated with and talked with in the early portions of my PhD and offered a wonderful social support structure during those years. Joe Janizek was one of my closest collaborators in my first two years and I thank him for the many research discussions, as well as occasional late nights spend writing before a deadline. Thanks to Ivan Evtimov, who was not only my collaborator for a brief foray into adversarial privacy, but also a close friend who gave me much life advice over the years.

A large thank you to everyone that I worked with and talked to in the Baker Lab, including Ivan Anishchenko, Jue Wang, Paul Kim, Andrew Kubaney, Declan Evans, Nate Corley, Magnus Bauer, Hannah Han, Gyu Rie Lee and many others. A particular thanks to Rohith Krishna, who was both a patient mentor and a wonderful colleague. Nao Hiranuma also deserves special mention - thank you for paving the way for what I did during my PhD, and advocating for my career at its most critical junctions.

I of course have to thank my family for so much of what I accomplished in this PhD: my father,

for his constant enthusiasm for research and mentorship that was an inspiration ever since I was young, my mother, for her unconditional support of every life path I embarked down upon, and my sister, who I've always looked up to but never had the courage to say so.

Last but not least, I want to thank all of the friends that I've made along this journey. From taking me on epic bike rides across the Olympic peninsula, to extended lunch breaks at the Seattle Central culinary school and impromptu nights cooking together or teaching me to swing dance, thank you for showing me what it means to truly live outside of work. Although you will remain nameless in this section, you know who you are.

Papers Included in This Dissertation.

The contents of Chapter 2 is primarily from three papers I co-authored early in my PhD: [42, 61, 125]. Chapter 3 references two workshop papers I wrote during two different internships: [126, 127]. Chapter 4 references a project that I had a small part in playing that ended up as a larger publication in Science: [71]. Some of the work in that last chapter was unpublished and was work done in the course of developing [71] that didn't make it into the paper.

Funding Acknowledgements.

Funding for the work in my PhD came from many sources. In particular, the early portion of my work in Chapter 2 was funded by the National Science Foundation (DBI-1552309, DBI-1759487), American Cancer Society (RSG-14-257-01-TBG), and National Institutes of Health (R01 NIA AG 061132).

The second portion of my work, in Chapter 3 was done as a series of two internships at Salesforce AI Research and Meta AI Research, respectively.

The final portion of my work in Chapter 4 was funded predominantly by the Microsoft Grant GF117374.

DEDICATION

To my sister Nina Sturmfels, who I have always said is the much cooler sibling in the family.

Chapter 1

INTRODUCTION

I had an unusually disjoint PhD. As a result, this will be an unusually disjoint thesis. I thought for a while about how to join the three main chapters in my thesis into one grand, unified story and ultimately decided to abandon the effort. What follows instead is a set of three, self-contained chapters that encompass the breadth of the work I did in the last six years.

1.1 Explainable Machine Learning

The first portion of my PhD focused on the fundamental question of interpreting or explaining machine learning models, and in particular neural networks. The goal of supervised learning is generally framed as follows: given some random variables x and y from a distribution $p_{x,y}$, learn a function $f(x)$ that approximates $p(y|x)$. Neural networks have become a popular tool for training supervised learning models, partially because of their representational power, and partially because of the rise of faster and more specialized hardware for the underlying matrix operations used to train them.

Such models are highly non-linear, and as a result can learn complex functions that map inputs to outputs, such as classifying objects in images or sentiment in text. The issue with using deep, non-linear models is that they are hard to interpret. Even if a model achieves high accuracy on a classification task, it gives no insight into why it made the prediction it did. Such insight is particularly important in applications in biology and medicine, where interpretability can garner trust and lead to biological insights.

On the flip side, using simpler models like regression models or additive models inherently lends it self to easy interpretation. However, such models often underperform neural networks, which are capable of learning pairwise and higher-order interactions between features that simpler models

are not. Therefore, there exists some balance between the complexity of the model used and the ability of the practitioner to interpret it. Many of the existing interpretability methods for complex models focus on feature attribution: the problem of distilling a model into locally-linear attributions scores on a per-instance basis, which allows the user to determine the most important features used to make a prediction. However, this framework is somewhat limited: it says nothing about how to interpret higher-order interactions between features, which is often important in correlated data, and how to use such insights to train better models.

This chapter of my thesis focuses on these questions: how do we use insights from feature attribution to go beyond single feature attributions? The chapter first lays the game-theoretic background for existing work on feature attributions. Then, it discusses some insights into how that game-theoretic background can be applied to the machine learning setting, and in particular discusses questions about what it means to have a feature missing to a model. It then uses these insights to derive three new methods: expected gradients, attribution priors, and integrated hessians, which not only improve our ability to interpret features in neural networks but also move towards pair-wise interaction attribution and discuss how to use feature attribution techniques to make more interpretable models.

1.2 Protein Language Modeling

This second chapter focuses on modeling protein sequences at scale. Modern language modeling techniques have shown that the right inductive biases in large deep learning models, plus a combination of large, unlabeled databases and massive compute can lead to powerful, semi-supervised representational models that learn from reconstruction losses on partial ablations of the input data, with the assumption that models learn something about the target domain via such reconstructive losses and that gathering unlabeled data is cheaper and easier than gathering labeled data. This is certainly true in natural language: the internet provides a wealth of unlabeled human text, and the ability to reconstruct words in context teaches the model something about which words naturally belong together syntatically and semantically.

Early work on applying such models to protein sequences seemed to indicated that this is also

true for protein sequences. Certainly, with modern sequencing methods it has become cheap and easy to sequence proteins. Evolutionary pressure also constraints the distribution of amino acids both locally and globally within a sequence. Protein language models have already accomplished much, from being used for design tasks to being fine-tuned for structure prediction.

However, it is important to acknowledge that there are also key differences between protein sequences and natural language. It would be remiss to simply accept the doctrine of natural language models without thinking carefully about how to apply them to a new domain, and particularly how to use existing tools and knowledge of that new domain to better adapt such models. This section focuses on adapting pre-training tasks for large language models to protein sequences. In particular, evolutionary history of a protein, captured through multiple sequence alignments, provide a rich structure that does not exist in natural language. This chapter discusses two possible, alternate pre-training tasks that take advantage of this structure that may work either stand-alone or in concert with existing pre-training tasks from natural language.

1.3 Protein-Ligand Co-folding

The final chapter in this thesis focuses on protein-ligand co-folding. Determining the structure of a protein given its amino acid sequence has been a long standing problem in structural bioinformatics, with early approaches relying on force-field and energy terms. As machine learning has progressed in concert with advanced hardware for large-scale computing, most modern approaches to protein structure prediction have turned towards machine learning methods. In particular, the last several years have seen a rise in machine learning methods taking advantage of advances in structural deep learning and pairwise attention to model structure given sequence with no auxiliary force-field minimization, and have lead to the creation of large structural databases of predicted structures for proteins that previously had no sequence.

Although protein structure prediction methods are now fairly mature, they ignore a central portion of overall picture: that proteins in nature do not fold alone. They act in concert with a variety of other, non-polymer molecules and the interactions between proteins and small molecules can influence the subsequent structures that a given protein adopts. Modeling these interactions is

critical for understanding how a protein functions, as well as designing new proteins with novel functions or novel therapeutics that inhibit a particular protein in a disease pathway.

Computationally, modeling the joint structure of proteins and small molecules is challenging for a variety of reasons. Unlike polymers, small molecules cannot be represented as linear chains of repeating sub-units, so computational methods have to be expanded to model arbitrary graph structures. A large portion of protein structure prediction is formed from learning on multiple sequence alignments, whose co-evolutionary statistics give information about which residues in a protein co-evolved and are likely close in 3D space in the folded structure of that protein. Such evolutionary history does not exist for small molecules, which do not, by necessity, face the same evolutionary patterns and pressures that govern proteins. The number of determined structures of protein small molecule complexes is far lower than that of protein structures alone, making the problem data limited. Finally, proteins and small molecules are, by definition, on different size scales, and so a model needs to reason over multiple resolutions during structure determination.

This last chapter details a model to jointly determine the structure of proteins and small molecules given their input sequence and chemistry. In particular, it details my work on benchmarking such a model - understanding how well it is performing at this task, and potential downstream applications of protein-ligand co-folding. Although this work is less glamorous than experimenting with the underlying architecture, it is equally as important, and helps lead to insights in how to improve the model and how we can use the model in a variety of surprising and insightful ways.

1.4 Summary

This thesis deals with three seemingly disconnected topics: explainable machine learning, protein language modeling and protein-ligand co-folding. The underlying link, however, is evident not in the results of each chapter but rather the underlying lessons learned from one chapter to the next. Insights in pairwise interaction detection lead naturally into thinking about what domains have strong pairwise interactions. Protein sequences are an obvious area, in which pairwise models of coevolution have long been central to structure modeling. The question arises about how best to train large, uninterpretable models on protein sequences, leading to the work about alternative

pre-training tasks. Limited by the domain of sequence-based representations, the natural next question is how to model biomolecular assemblies in more generality, leading towards the structural modeling work in the final chapter. I hope the following chapters, as disjoint as they may be, shed some insight into my path as a PhD student: from thinking about core questions in machine learning to moving closer and closer to how they can be used to truly learn something meaningful about biology.

Chapter 2

EXPLAINABLE MACHINE LEARNING

2.1 Feature Attribution

It is a well-understood problem that although neural networks often outperform simpler methods on structured data in domains like vision and language, they are black box models that are difficult to interpret. Why did the model predict that this patient's x-ray is indicative of pneumonia? Which residues were most important in contributing to the predicted stability of this protein? In many domains, particularly in medical and biological applications, an accurate model is only half of the picture. Interpreting *how* a model makes a predictions it does can built trust to use that model in sensitive applications and can give rise to biological insights in the underlying data. However, what it means to "interpret" a model is poorly-defined. A useful definition would allow for practical use-cases in such domains.

One way to narrow the scope of interpretability is to consider the problem of feature attribution. The problem of feature attribution is essentially to distill non-linear model f into a locally-linear model parameterized by $\phi(f, x) \in \mathbb{R}^d$ around some data point $x \in R^d$, such that $\phi(f, x)_i$ represents the local "importance" of the value of x_i in making the prediction $f(x)$. There are many ways to define how this function ϕ should behave, but no way to truly measure the ground truth value of how well this local function approximates the original model f. In the absence of ground truth benchmarks, we turn to game theory to provide guiding axioms that govern the behavior of ϕ in limited circumstances.

2.2 Game Theoretic Explanations

The axioms in this section come from game theory, specifically the Shapley value and its various extensions, that were originally developed for the fair allocation of credit in cooperative games [113].

In brief, let U be a set and v be a super-additive set function $v : U \mapsto \mathbb{R}$ satisfying the following:

$$v(\varnothing) = 0$$

$$v(S) \geq v(S \cap T) + v(S \setminus T), \forall S, T \subseteq U$$

We call v a game on n players where $|U| = n$. Let $\phi_i(v)$ be a value function for an arbitrary member $i \in U$ that maps game v on that subset to its value $\mathbb{R}$. We think of this function abstractly as denoting the "worth" of the member i, e.g. that element's contribution to the total value of the game $v(U)$. We define a carrier of the game v as any subset $N \subseteq U$ such that $v(S) = V(n \cap S)$ for any $S \subseteq U$. Intuitively, a carrier of a game represents the essential players of that game. No other members in U add additional value to the game once the carrier set is present.

We define the following three axioms:

1. Symmetry: $\phi_{\pi i}(\pi v) = \phi_i(v)$ for some permutation function π on U.

2. Efficiency: $\sum_{i \in N} \phi_i(v) = v(N)$ for any carrier set N.

3. Additivity: $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$ for any two games v and w.

The axioms are described in more detail in Shapley's original work, but they provide a framework for how to think about the value function ϕ_i. In particular, Lloyd Shapley proved that assuming the above three axioms is sufficient to uniquely determine ϕ_i:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S))$$

We can use the Shapley Value to interpret machine learning models in the following way: given some machine learning model $f : \mathbb{R}^d \mapsto \mathbb{R}$ and some point $x \in R^d$, we can interpret $f(x)$ as a coalitional game whose members are x_i, the features of the input x. We let the entire set of features be a carrier (which is true for any coalitional game), and define f_x as a function on subsets of features $\{x_i, ...\}$. The values $\phi_i(f_x)$ by definition tell us the unique contribution of feature x_i to the output value f_x such that $\sum_i \phi_i(f_x) = f(x)$.

Although this interpretation is appealing, it leaves two important problems. The first is that for large n, computing the Shapley Value is intractable (and indeed in the general case, NP-Hard). Approximating the Shapley Value or computing it directly for particular model classes has been the subject of much recent work. In particular, [86] showed that it is possible to define a least-weighted squares regression that is an unbiased estimator of the Shapley Value for arbitrary f and x with significantly less variance than uniform random sampling of S. [85] demonstrated that it is possible to compute the Shapley Value exactly in polynomial time for decision tree models, and [62] proposed an algorithm to amortize the cost of computing Shapley Values for all points x via training a neural network estimator.

The second problem is that coalitional games are functions defined on sets, which means that v should have a defined value for every subset $S \subseteq U$. However, it is not precisely clear how to evaluate an arbitrary machine learning model f at an arbitrary subset of features x_S where S represents dimensions in $\mathbb{R}^d$. For example, if you have a model trained on classifying images, it is not immediately clear how to evaluate such a model on only a portion of the image. Work by [130] extended attributions to use a variant of the Shapley Value, the Aumann-Shapley Value [10], which extends the coalition to an infinitely large set, and in doing so proposed computing these values relative to a "baseline" value that represents lack of information to the model. That is, in order to ablate a feature, they propose to set it to the baseline value, which they take to be zero. However, it is not clear that this choice of baseline value makes sense in many real-world settings, which is precisely what the next section discusses.

2.3 On Attribution Baselines in Game-Theoretic Explanations

The method introduced in [130], called integrated gradients, has the following definition:

$$\phi_i(x) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\delta f(x' + \alpha(x - x'))}{\delta x_i} \delta\alpha$$

for some model f, a datapoint $x \in \mathbb{R}^d$ and a baseline value $x' \in \mathbb{R}^d$ that is meant to represent that a feature is missing to the model, e.g. to allow the user to interpret f as a game. But how should

you choose x' in order to best represent this? It seems to be common practice to choose a baseline input x' to be the vector of all zeros. But consider the following scenario: you've learned a model on a healthcare dataset, and one of the features is blood sugar level. The model has correctly learned that excessively low levels of blood sugar, which correspond to hypoglycemia, is dangerous. Does a blood sugar level of 0 seem like a good choice to represent missingness?

The point here is that fixed feature values may have unintended meaning. The problem compounds further when you consider the difference from baseline term $x_i - x'_i$. For the sake of a thought experiment, suppose a patient had a blood sugar level of 0. To understand why our machine learning model thinks this patient is at high risk, you run integrated gradients on this data point with a baseline of the all-zeros vector. The blood sugar level of the patient would have 0 feature importance, because $x_i - x'_i = 0$. This is despite the fact that a blood sugar level of would be fatal!

We find similar problems when we move to the image domain. If you use a constant black image as a baseline, integrated gradients will not highlight black pixels as important even if black pixels make up the object of interest. More generally, the method is blind to the color you use as a baseline, which we illustrate with the figure below. Note that this was acknowledged by the original authors in [129], and is in fact central to the definition of a baseline: we wouldn't want integrated gradients to highlight missing features as important! But then how do we avoid giving zero importance to the baseline color?

It's clear that any constant color baseline will have this problem. Are there any alternatives? In [125], we run through a variety of choices and their various problems. Ultimately, however, we demonstrate that possibly the best solution is to think about the problem distributionally rather than trying to find a single, constant value that represents missingness. In particular, if we don't know the value of a feature, we may assume a distribution over what that feature's value should be either from prior knowledge or from the training data that we have available to us. We can do this approximately by averaging the value of integrated gradients over multiple baselines drawn from some distribution D.

At this point, it's worth connecting the idea of averaging over multiple baselines back to the original definition of integrated gradients. When we average over multiple baselines from the

same distribution D, we are attempting to use the distribution itself as our baseline. We use the distribution to define the notion of missingness: if we don't know a pixel value, we don't assume its value to be 0 - instead we assume that it has some underlying distribution D. Formally, given a baseline distribution D, we integrate over all possible baselines $x' \in D$ weighted by the density function p_D:

$$\phi_i(f, x) = \int_{x'} ((x_i - x'_i) \int_{\alpha=0}^{1} \frac{\delta f(x' + \alpha(x - x'))}{\delta x_i} \delta\alpha) p_D(x') dx'$$

In terms of missingness, assuming a distribution might intuitively feel like a more reasonable assumption to make than assuming a constant value. But this doesn't quite solve the issue: instead of having to choose a baseline x', now we have to choose a baseline distribution D. Have we simply postponed the problem? We will discuss one theoretically motivated way to choose D in an upcoming section, but before we do, we'll take a brief aside to talk about how we compute the formula above in practice, and a connection to an existing method that arises as a result.

2.3.1 Expectations, and Connections to SmoothGrad

Now that we've introduced a second integral into our formula, we need to do a second discrete sum to approximate it, which requires an additional hyperparameter: the number of baselines to sample. In [41], we introduced the idea of expected gradients, which we define formally as:

$$\phi_i(f, x) = \mathbb{E}_{x' \sim D, \alpha \sim U(0,1)}((x_i - x'_i) \frac{\delta f(x' + \alpha(x - x'))}{\delta x_i})$$

Expected gradients and integrated gradients belong to a family of methods known as "path attribution methods" because they integrate gradients over one or more paths between two valid inputs. To compute expected gradients in practice, we use the following formula:

$$\hat{\phi}_i^{EG}(f, x; D) = \frac{1}{k} \sum_{j=1}^{k} (x_i - x_i'^j) \times \frac{\delta f(x'^j + \alpha^j(x - x'^j))}{\delta x_i}$$

where x'^j is the jth sample from D and α^j is the jth sample from the uniform distribution between 0 and 1. Now suppose that we use the gaussian baseline with variance σ^2. Then we can re-write the

formula for expected gradients as follows:

$$\hat{\phi}_i^{EG}(f, x; N(x, \sigma^2 I)) = \frac{1}{k} \sum_{j=1}^{k} \epsilon_\sigma^j \times \frac{\delta f(x + (1 - \alpha^j)\epsilon_\sigma^j)}{\delta x_i}$$

where $\epsilon_\sigma \sim N(\bar{0}, \sigma^2 I)$. This looks awfully familiar to an existing method called SmoothGrad [119]. If we use the (gradients $\times$ input image) variant of SmoothGrad, then we have the following formula:

$$\phi_i^{SG}(f, x; N(\bar{0}, \sigma^2 I)) = \frac{1}{k} \sum_{j=1}^{k} (x + \epsilon_\sigma^j) \times \frac{\delta f(x + \epsilon_\sigma^j)}{\delta x_i}$$

We can see that SmoothGrad and expected gradients with a gaussian baseline are quite similar, with two key differences: SmoothGrad multiplies the gradient by $x + \epsilon_\sigma$ while expected gradients multiplies by just ϵ_σ, and while expected gradients samples uniformly along the path, SmoothGrad always samples the endpoint $\alpha = 0$.

Can this connection help us understand why SmoothGrad creates smooth-looking saliency maps? When we assume the above gaussian distribution as our baseline, we are assuming that each of our pixel values is drawn from a gaussian independently of the other pixel values. But we know this is far from true: in images, there is a rich correlation structure between nearby pixels. Once your network knows the value of a pixel, it doesn't really need to use its immediate neighbors because it's likely that those immediate neighbors have very similar intensities.

Assuming each pixel is drawn from an independent gaussian breaks this correlation structure. It means that expected gradients tabulates the importance of each pixel independently of the other pixel values. The generated saliency maps will be less noisy and better highlight the object of interest because we are no longer allowing the network to rely on only pixel in a group of correlated pixels. This may be why SmoothGrad is smooth: because it is implicitly assuming independence among pixels.

2.3.2 Using the Training Distribution

Is it really reasonable to assume independence among pixels while generating saliency maps? In supervised learning, we make the assumption that the data is drawn from some distribution D_{data}.

This assumption that the training and testing data share a common, underlying distribution is what allows us to do supervised learning and make claims about generalizability. Given this assumption, we don't need to model missingness using a gaussian or a uniform distribution: we can use D_{data} to model missingness directly.

The only problem is that we do not have access to the underlying distribution. But because this is a supervised learning task, we do have access to many independent draws from the underlying distribution: the training data! We can simply use samples from the training data as random draws from D_{data}. This brings us to the variant of expected gradients that we proposed in both [125] and [41]:

$$\frac{1}{k}\sum_{j=1}^{k} \underbrace{(x_i - x_i'^j) \times \frac{\delta f(\overbrace{x'^j + \alpha^j(x - x'^j)}^{\text{(1): Interpolated Image}})}{\delta x_i}}_{\text{(2): Gradients at Interpolation}} = \overbrace{\hat{\phi}_i^{EG}(f, x, k; D_{\text{data}})}^{\text{(3): Cumulative Gradients up to } \alpha}$$

The advantage of this formulation is twofold. First, since it doesn't use a constant baseline, no feature values will "automatically" get zero attribution. Second, since it more accurately reflects the conditional distribution of a feature value given other feature values, it should better reflect the importance of knowing a feature's value relative to the data the model was trained on.

2.4 Axiomatic Feature Interactions

This section deals with extending feature attribution methods, which assign an importance value to every feature in the model, into feature interaction methods, which assign interaction scores for pairwise groups of features. This extension allows interpretability beyond linear contributions into second-order terms.

2.4.1 Feature interaction methods

There are several existing methods that explain feature interactions in neural networks. [28] propose a method to explain global interactions in Bayesian Neural Networks (BNN) by examining pairs of features that have large second-order derivatives at the input. Neural Interaction Detection is a

method that detects statistical interactions between features by examining the weight matrices of feed-forward neural networks [133]. Furthermore, several authors have proposed domain-specific methods for finding interactions in the area of deep learning for genomics [51, 70]. For example, Deep Feature Interaction Maps detect interactions between two features by calculating the change in the attribution of one feature incurred by changing the value of the second [51]. [118] propose a generalization of Contextual Decomposition [94] that can explain interactions for feed-forward and convolutional architectures. In game theory literature, [50] propose the Shapley Interaction Index, which allocates credit to interactions between players in a coalitional game by considering all possible subsets of players. [32] suggest a modified version of the Shapley Interaction Index that provides a different weighting to certain subsets of players in order to achieve a different set of desired axioms.

2.4.2 Limitations of Prior Approaches

While previous approaches have taken important steps towards understanding feature interaction in neural networks, all suffer from practical limitations, including being limited to specific types of architectures. Neural Interaction Detection only applies to feed-forward neural network architectures, and can not be used on networks with convolutions, recurrent units, or self-attention. Contextual Decomposition has been applied to LSTMs, feed-forward neural networks and convolutional networks, but to our knowledge is not straightforward to apply to more recent innovations in deep learning, such as self-attention layers. The approach suggested by [28] is limited in that it requires the use of Bayesian Neural Networks; it is unclear how to apply the method to standard neural networks. Deep Feature Interaction Maps only work when the input features for a model have a small number of discrete values (such as genomic sequence data which can be *in silico* mutated), as the method involves marginalizing over all possible values of paired input features. The Shapley Interaction Index and Shapley Taylor Interaction Index, like the Shapley Value, are NP-hard to compute exactly [38].

Furthermore, most existing methods to detect interactions do not satisfy the common-sense axioms that have been proposed by feature attribution methods [86, 130]. This leads to these previous

approaches being provably unable to find learned interactions, or more generally finding counter-intuitive interactions. Existing methods that do satisfy such axioms, such as those based on the Shapley Interaction Index [32, 50], are computationally inefficient to compute or even approximate.

2.5 Explaining Explanations with Integrated Hessians

To derive our feature interaction values, we start by considering Integrated Gradients (IG), a feature attribution method based on the Aumann-Shapley value proposed by [130]. The Aumann-Shapley value is a variant of the Shapley value for cooperative games with continuous rather than discrete players [10]. We represent our model as a function $f : \mathbb{R}^d \mapsto \mathbb{R}$.[1]. For a function $f(x)$, the IG attribution for the ith feature is defined as:

$$\phi_i(x) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha, \tag{2.1}$$

where x is the sample we would like to explain and x' is a baseline value. Although f is often a neural network, the only requirement in order to compute attribution values is that f be differentiable along the path from x' to x. Our key insight is that the IG value for a differentiable model $f : \mathbb{R}^d \mapsto \mathbb{R}$ is *itself* a differentiable function $\phi_i : \mathbb{R}^d \mapsto \mathbb{R}$. This means that we can apply IG to itself in order to explain how much feature j impacted the importance of feature i:

$$\Gamma_{i,j}(x) = \phi_j(\phi_i(x)) \tag{2.2}$$

For $i \neq j$, we can derive that:

$$\Gamma_{i,j}(x) = (x_i - x_i')(x_j - x_j') \times \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta \tag{2.3}$$

In the case of $i = j$, the formula $\Gamma_{i,i}(x)$ has an additional first-order term:

$$\begin{aligned} \Gamma_{i,i}(x) = (x_i - x_i') \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha\beta(x - x'))}{\partial x_i} d\alpha d\beta + \\ (x_i - x_i')^2 \times \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta \end{aligned} \tag{2.4}$$

[1] In the case of multi-output models, such as multi-class classification problems, we assume the function is indexed into the correct output class.

We interpret $\Gamma_{i,j}(x)$ as the explanation of the importance of feature i in terms of the input value of feature j.

2.5.1 Baselines and Expected Hessians

Several existing feature attribution methods have pointed out the need to explain relative to a baseline value that represents a lack of information that the explanations are relative to [15, 86, 115, 130]. However, more recent work has pointed out that choosing a single baseline value to represent lack of information can be challenging in certain domains [6, 45, 66, 68, 125, 129]. As an alternative, [41] proposed an extension of IG called Expected Gradients (EG), which samples many baseline inputs from the training set. We can therefore apply EG to itself to get Expected Hessians.

$$\Gamma^{EG}_{i,j}(x) = \mathbb{E}_{\alpha\beta\sim U(0,1)\times U(0,1), x'\sim D}\left[(x_i - x'_i)(x_j - x'_j)\alpha\beta\frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial dx_i \partial dx_j}\right] \tag{2.5}$$

$$\begin{aligned}\Gamma^{EG}_{i,i}(x) = \mathbb{E}_{\alpha\beta\sim U(0,1)\times U(0,1), x'\sim D}\Bigg[&(x_i - x'_i)\frac{\partial f(x' + \alpha\beta(x - x'))}{\partial x_i} \\ &+ (x_i - x'_i)^2\alpha\beta\frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial dx_i \partial dx_j}\Bigg]\end{aligned} \tag{2.6}$$

where the expectation is over $x' \sim D$ for an underlying data distribution D, $\alpha \sim U(0, 1)$ and $\beta \sim U(0, 1)$. This formulation can be useful in the case where there does not exist a single, natural baseline. The derivation for Expected Hessians follows identical steps as the derivation for Integrated Hessians while observing that the integrals can be viewed as integrating over the product of two uniform distributions $\alpha\beta \sim U(0, 1) \times U(0, 1)$. We use Integrated Hessians for all of the examples in the main text - however, some of the additional examples shown in the Appendix use the Expected Hessians formulation.

We note that almost every attribution and interaction method to date requires a choice of baseline. Some approaches explicitly discuss their reliance on baselines [128, 130, 134], while other methods rely on baselines without explicitly mentioning it in their work. For example, the Contextual Decomposition method models turning features "off" by setting them equal to a single reference baseline of 0 [94]. By implementing a solution for domains where it is hard to choose a single

baseline, we hope to alleviate some of the issues common to many interaction methods.

2.5.2 Fundamental Axioms for Interaction Values

In this section, we describe common-sense axioms that any interaction method should satisfy, and show that Integrated Hessians satisfies them all.

Self and Interaction Completeness

[130] showed that, among other theoretical properties, IG satisfies the completeness axiom, which states: $\sum_i \phi_i(x) = f(x) - f(x')$. We can show the following two equalities, which are immediate consequences of completeness:

$$\sum_i \sum_j \Gamma_{i,j}(x) = f(x) - f(x') \tag{2.7}$$

$$\Gamma_{i,i}(x) = \phi_i(x) - \sum_{j \neq i} \Gamma_{i,j}(x) \tag{2.8}$$

We call equation (2.7) the *interaction completeness* axiom: the sum of the $\Gamma_{i,j}(x)$ terms adds up to the difference between the output of f at x and at the baseline x'. This axiom lends itself to another natural interpretation of $\Gamma_{i,j}(x)$: as the interaction between features i and j. That is, it represents the contribution that the pair of features i and j together add to the output $f(x) - f(x')$. Satisfying interaction completeness is important because it demonstrates a relationship between model output and interaction values. Without this axiom, it is unclear how to interpret the scale of interactions.

Equation (2.8) provides a way to interpret the self-interaction term $\Gamma_{i,i}(x)$: it is the *main effect* of feature i after interactions with all other features have been subtracted away. We note that equation (2.8) also implies the following, intuitive property about the main effect: if $\Gamma_{i,j} = 0$ for all $j \neq i$, or in the degenerate case where i is the only feature, we have $\Gamma_{i,i} = \phi_i(x)$. We call this the *self completeness* axiom. Satisfying self-completeness is important because it provides a guarantee that

the main effect of feature i equals its feature attribution value if that feature interacts with no other features.

The proof of these two equations is straightforward. First, we note that $\phi_i(x') = 0$ for any i because $x'_i - x'_i = 0$. Then, by completeness of Integrated Gradients, we have that:

$$\sum_j \Gamma_{i,j}(x) = \phi_i(x) - \phi_i(x') = \phi_i(x). \tag{2.9}$$

Re-arrangement gives us the *self completeness* axioms:

$$\Gamma_{i,i}(x) = \phi_i(x) \text{ if } \Gamma_{i,j}(x) = 0, \forall j \neq i. \tag{2.10}$$

Since Integrated Gradients satisfies completeness, we have:

$$\sum_i \phi_i(x) = f(x) - f(x'). \tag{2.11}$$

Making the appropriate substitution from equation 2.9 shows the *interaction completeness* axiom:

$$\sum_i \sum_j \Gamma_{i,j}(x) = f(x) - f(x'). \tag{2.12}$$

Sensitivity

Integrated Gradients satisfies an axiom called sensitivity, which can be phrased as follows. Given an input x and a baseline x', if $x_i = x'_i$ for all i except j where $x_j \neq x'_j$ and if $f(x) \neq f(x')$, then $\phi_j(x) \neq 0$. Specifically, by completeness we know that $\phi_j(x) = f(x) - f(x')$. Intuitively, this is saying that if only one feature differs between the baseline and the input and changing that feature changes the output, then the amount the output changes should be equal to the importance of that feature.

We can extend this axiom to the interaction case by considering the case when two features differ from the baseline. We call this axiom *interaction sensitivity*, and can be described as follows. If an input x and a baseline x' are equal everywhere except $x_i \neq x'_i$ and $x_j \neq x'_j$, and if $f(x) \neq f(x')$, then: $\Gamma_{i,i}(x) + \Gamma_{j,j}(x) + 2\Gamma_{i,j}(x) = f(x) - f(x') \neq 0$ and $\Gamma_{\ell,k} = 0$ for all $\ell, k \neq i, j$. Intuitively,

this says that if the only features that differ from the baseline are i and j, then the difference in the output $f(x) - f(x')$ must be solely attributable to the main effects of i and j plus the interaction between them. This axiom holds simply by applying *interaction completeness* and observing that $\Gamma_{\ell,k}(x) = 0$ if $x_\ell = x'_\ell$ or $x_k = x'_k$.

Implementation Invariance

The implementation invariance axiom, described in the original paper, states the following. For two models f and g such that $f = g$, then $\phi_i(x; f) = \phi_i(x; g)$ for all features i and all points x regardless of how f and g are implemented. Although it seems trivial, this axiom does not necessarily hold for attribution methods that use the implementation or structure of the network in order to generate attributions. Critically, this axiom also does not hold for the interaction method proposed by [133], which looks at the first layer of a feed forward neural network. Two networks may represent exactly the same function but differ greatly in their first layer.

This axiom is trivially seen to hold for Integrated Hessians since it holds for Integrated Gradients. However, this axiom is desirable because without it, it may mean that attributions/interactions are encoding information about unimportant aspects of model structure rather than the actual decision surface of the model.

Linearity

Integrated Gradients satisfies an axiom called linearity, which can be described as follows. Given two networks f and g, consider the output of the weighted ensemble of the two networks $af(x) + bg(x)$. Then the attribution $\phi_i(x; af + bg)$ of the weighted ensemble equals the weighted sum of attributions $a\phi_i(x; f) + b\phi_i(x; g)$ for all features i and samples x. This axiom is desirable because it preserves linearity within a network, and allows easy computation of attributions for network ensembles.

We can generalize linearity to interactions using the *interaction linearity* axiom:

$$\Gamma_{i,j}(x; af + bg) = a\Gamma_{i,j}(x; f) + b\Gamma_{i,j}(x; g) \tag{2.13}$$

for any i, j and all points x. Given that $\Gamma_{i,j}$ is composition of linear functions ϕ_i, ϕ_j in terms of the parameterized networks f and g, it is itself a linear function of the networks and therefore Integrated Hessians satisfies *interaction linearity*.

Symmetry-Preserving

We say that two features x_i and x_j are *symmetric* with respect to f if swapping them does not change the output of f anywhere. That is, $f(\cdots, x_i, \cdots, x_j, \cdots) = f(\cdots, x_j, \cdots, x_i, \cdots)$. The original paper shows that Integrated Gradients is *symmetry-preserving*, that is, if x_i and x_j are symmetric with respect to f, and if $x_i = x_j$ and $x'_i = x'_j$ for some input x and baseline x', then $\phi_i(x) = \phi_j(x)$. We can make the appropriate generalization to interaction values: if the same conditions as above hold, then $\Gamma_{k,i}(x) = \Gamma_{k,j}(x)$ for any feature x_k. This axiom holds since, again $\Gamma_{k,i}(x) = \phi_i(\phi_k(x))$ and ϕ_i, ϕ_j are symmetry-preserving. This axiom is desirable because it says that if two features are functionally equivalent to a model, then they must interact the same way with respect to that model.

Interaction Symmetry

There is another form of symmetry that is important to note: *interaction symmetry*. This axiom states that, for any i, j, we have $\Gamma_{i,j}(x) = \Gamma_{j,i}(x)$. Although this axiom is simple, it guarantees that the interaction function itself is symmetric with respect to the features it explains. It is straightforward to show that existing neural networks and their activation functions have continuous second partial derivatives, which implies that Integrated Hessians satisfies interaction symmetry.[2]

2.5.3 Approximating Integrated Hessians in Practice

Our interaction values involve a double integral, which is intractable to compute analytically in the general case. To compute Integrated Gradients in practice, [130] introduced the following discrete sum approximation:

[2]We discuss the special case of the ReLU activation function in Section 2.6.

$$\hat{\phi}_i(x) = (x_i - x_i') \times \sum_{\ell=1}^{k} \frac{\partial f(x' + \frac{\ell}{k}(x - x'))}{\partial x_i} \times \frac{1}{k} \tag{2.14}$$

where k is the number of points used to approximate the integral. To compute Integrated Hessians, we introduce a similar discrete sum approximation:

$$\hat{\Gamma}_{i,j}(x) = (x_i - x_i')(x_j - x_j') \times \sum_{\ell=l}^{k} \sum_{p=1}^{m} \frac{\ell}{k} \times \frac{p}{m} \times \frac{\partial f(x' + (\frac{\ell}{k} \times \frac{p}{m})(x - x'))}{\partial x_i \partial x_j} \times \frac{1}{km} \tag{2.15}$$

Typically, it is easiest to compute this quantity when $k = m$ and the number of samples drawn is thus a perfect square - however, when a non-square number of samples is desired we can generate a number of sample points from the product distribution of two uniform distributions such that the number is the largest perfect square above the desired number of samples, and index the sorted samples appropriately to get the desired number. The above formula omits the first-order term in $\Gamma_{i,i}(x)$ but it can be computed using the same principle.

Expected Hessians has a similar, if slightly easier form:

$$\hat{\Gamma}_{i,j}^{EG}(x) = (x_i - x_i')(x_j - x_j') \sum_{\ell}^{k} \zeta_\ell \times \frac{\partial f(x' + \zeta_\ell(x - x'))}{\partial x_i \partial x_j} \times \frac{1}{k} \tag{2.16}$$

where ζ_ℓ is the ℓth sample from the product distribution of two uniform distributions. We find that in general that less than 300 samples are required for any given problem to approximately satisfy interaction completeness. For most problems, a number far less than 300 suffices (e.g. around 50) although this is model and data dependent: larger models and higher-dimensional data generally require more samples than smaller models and lower-dimensional data.

2.6 Smoothing ReLU Networks

One major limitation that has not been discussed in previous approaches to interaction detection in neural networks is related to the fact that many popular neural network architectures use the ReLU

activation function, $\text{ReLU}(x) = \max(0, x)$. Neural networks that use ReLU are piecewise linear and have second partial derivatives equal to zero in all places. Previous second-order approaches (based on the Hessian) fail to detect any interaction in ReLU-based networks.

Fortunately, the ReLU activation function has a smooth approximation – the SoftPlus function: $\text{SoftPlus}_\beta(x) = \frac{1}{\beta}\log(1 + e^{-\beta x})$. SoftPlus more closely approximates the ReLU function as β increases, and has well-defined higher-order derivatives. Furthermore, [33] have proved that both the outputs and first-order feature attributions of a model are minimally perturbed when ReLU activations are replaced by SoftPlus activations in a trained network. Therefore, we can apply Integrated Hessians on a network with ReLU activations by first replacing ReLU with SoftPlus. We note that no re-training is necessary for this approach.

In addition to being twice differentiable and allowing us to calculate interaction values in ReLU networks, replacing ReLU with SoftPlus leads to other desirable behavior for calculating interaction values. We show that smoothing a neural network (decreasing the value of β in the SoftPlus activation function) lets us accurately approximate the Integrated Hessians value with fewer gradient calls.

Theorem 1. *For a one-layer neural network with softplus$_\beta$ non-linearity, $f_\beta(x) =$ softplus$_\beta(w^T x)$, and d input features, we can bound the number of interpolation points k needed to approximate the Integrated Hessians to a given error tolerance ϵ by $k \leq \mathcal{O}(\frac{d\beta^2}{\epsilon})$.*

Proof of the above is mainly symbolic manipulation and doesn't lend itself to intuitive results, so we leave the proof in [61]. Here we lend some more intuition behind these results. As we replace ReLU with SoftPlus, the decision surface of the network is smoothed - see Figure 2.1 and [33]. We can see that the gradients tend to all have more similar direction along the path from reference to foreground sample once the network has been smoothed with SoftPlus replacement.

2.6.1 Interpreting Language Models

In the past decade, neural networks have been the go-to model for language tasks, from convolutional [67] to recurrent [131]. More recently large, pre-trained transformer architectures [30, 96] have

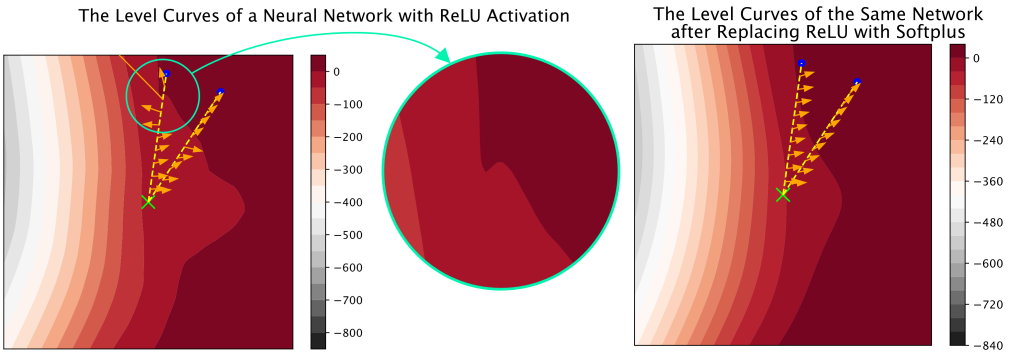


Figure 2.1: Replacing ReLU activations with SoftPlus$_\beta$ activations with $\beta = 10$ smooths the decision surface of a neural network: gradients tend to be more homogeneous along the integration path. Orange arrows show the gradient vectors at each point along the path from the reference (green x) to the input (blue dots). ReLUs can cause small bends in the output space with aberrant gradients.

achieved state of the art performance on a wide variety of tasks. Some previous work has suggested looking at the internal weights of the attention mechanisms in attention-based models [47,76,79,141]. However, more recent work has suggested that looking at attention weights may not be a reliable way to interpret models with attention layers [16, 60, 112]. To overcome this, feature attributions have been applied to text classification models to understand which words most impacted the classification [73, 80]. However, these methods do not explain how words interact with their surrounding context.

We download pre-trained weights for DistilBERT [109] from the HuggingFace Transformers library [142]. We fine-tune the model on the Stanford Sentiment Treebank dataset [120] in which the task is to predict whether or not a movie review has positive or negative sentiment. After 3 epochs of fine-tuning, DistilBERT achieves a validation accuracy of 0.9071 (0.9054 TPR / 0.9089 TNR).[3] We leave further fine-tuning details to Appendix D.

In Figure 2.2, we show interactions generated by Integrated Hessians and attributions generated

[3]This performance does not represent state of the art, nor is sentiment analysis representative of the full complexity of existing language tasks. However, our focus in this paper is on explanation and this task is easy to fine-tune without needing to extensively search over hyper-parameters.

by Integrated Gradients on an example drawn from the validation set. The figure demonstrates that DistilBERT has learned intuitive interactions that would not be revealed from feature attributions alone. For example, a word like “painfully," which might have a negative connotation on its own, has a large positive interaction with the word “funny" in the phrase “painfully funny." In Figure 2.3, we demonstrate how interactions can help us understand one reason why a fine-tuned DistilBERT model outperforms a simpler model: a convolutional neural network (CNN) that gets an accuracy of 0.82 on the validation set. DistilBERT picks up on positive interactions between negation words (“not”) and negative adjectives (“bad”) that a CNN fails to fully capture. Finally, in Figure 2.2, we use interaction values to reveal saturation effects: many negative adjectives describing the same noun interact positively. Although this may seem counter-intuitive at first, it reflects the structure of language. If a phrase has only one negative adjective, it stands out as the word that makes the phrase negative. At some point, however, describing a noun with more and more negative adjectives makes any individual negative adjective less important towards classifying that phrase as negative.

2.7 Attribution Priors

This section introduces a new training framework that leverages feature attribution techniques, like the expected gradients method introduced above, in order to make better and more interpretable models. Let $X \in \mathbb{R}^{n \times p}$ denote a dataset with labels $y \in \mathbb{R}^{n \times o}$, where n is the number of samples, p is the number of features, and o is the number of outputs. In standard deep learning, we find optimal parameters θ by minimizing loss, with a regularization term $\Omega'(\theta)$ weighted by λ' on the parameters:

$$\theta = \text{argmin}_\theta \mathcal{L}(\theta; X, y) + \lambda' \Omega'(\theta).$$

Attribution priors involve a model’s attributions, represented by the matrix $\Phi(\theta, X)$, where each entry ϕ_i^ℓ is the importance of feature i in the model’s output for sample ℓ. The attribution prior is a scalar-valued penalty function of the feature attributions $\Omega(\Phi(\theta, X))$, which represents a log-transformed prior probability distribution over possible attributions (λ is the regularization strength). The attribution prior is modular and agnostic to the particular attribution method. This results in the

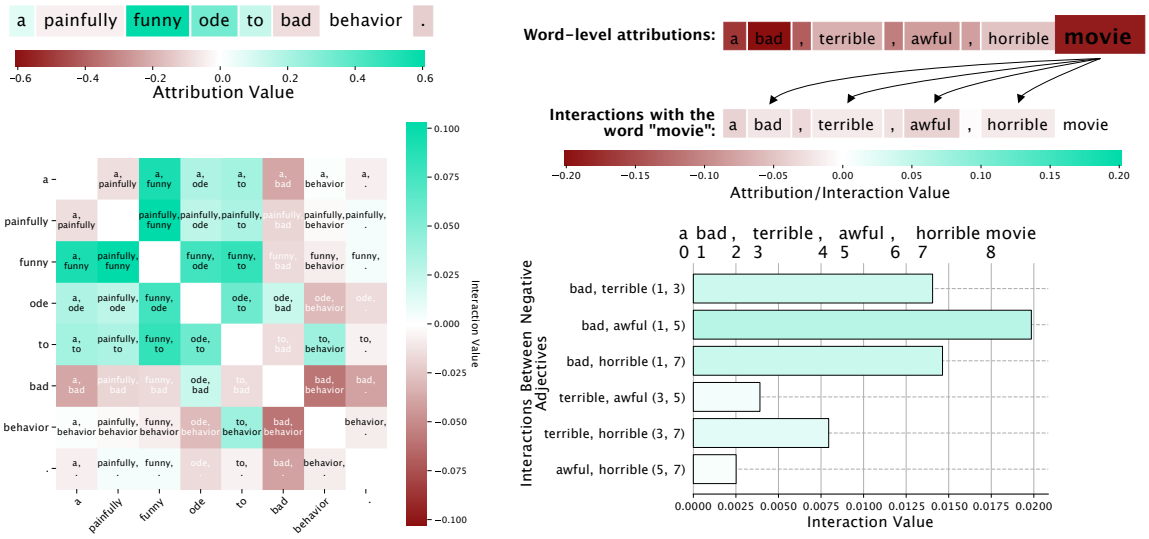


Figure 2.2: *Left*: Interactions in text reveal learned patterns such as the phrase "painfully funny" having positive interaction despite the word "painfully" having negative attribution. These interactions are not evident from attributions alone. *Right*: Interactions help us reveal an unintuitive pattern in language models: saturation. Although the word "movie" interacts negatively with all negative modifying adjectives, those negative adjectives themselves all interact positively. The more negative adjectives are in the sentence, the less each individual negative adjective matters towards the overall classification of the sentence.

optimization:

$$\theta = \text{argmin}_\theta \mathcal{L}(\theta; X, y) + \lambda \Omega(\Phi(\theta, X)),$$

where the standard regularization term has simply been replaced with an arbitrary, differentiable penalty function on the feature attributions.

If the attribution function Φ in our attribution prior $\Omega(\Phi(\theta, X))$ is integrated gradients, regularizing Φ would require hundreds of extra gradient calls every training step (the original integrated gradients paper [130] recommends 20 to 300 gradient calls to compute attributions). This makes training with integrated gradients prohibitively slow – in fact, [80] find that using integrated gradients can take up to 30 times longer than standard training even when only back-propagating gradients through part of the network. Even more gradient calls would be necessary if multiple

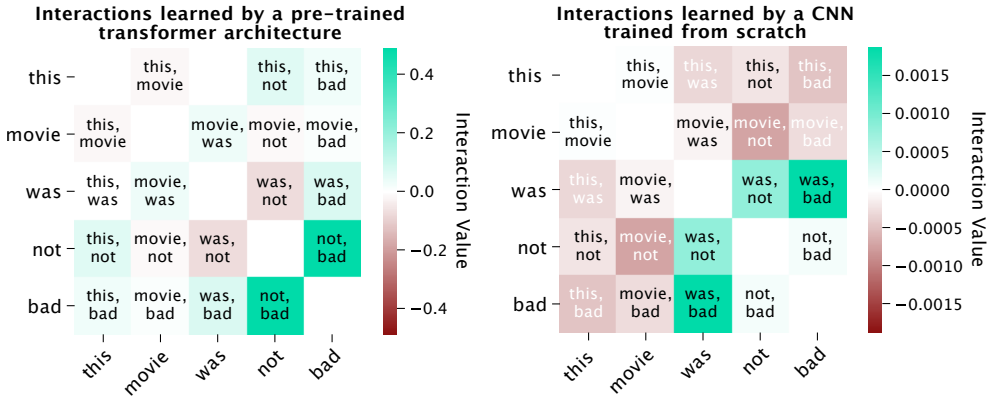


Figure 2.3: Interactions help us understand why certain models perform better than others. Here, we examine word interactions within the sentence “this movie was not bad." We compare two models trained to perform sentiment analysis on the Stanford Sentiment dataset: a pre-trained transformer, DistilBERT (left), which predicts the sentence has a positive sentiment with 98.2% confidence, and a convolutional neural network trained from scratch (right), which predicts a negative sentiment with 97.6% confidence. The transformer picks up on negation patterns: “not bad” has a positive interaction, despite the word “bad” being negative. The CNN mostly picks up on negative interactions like “movie not” and “movie bad”.

references were used, as proposed above. However, most deep learning models today are trained using some variant of batch gradient descent, where the gradient of a loss function is approximated over many training steps using mini-batches of data. We can use a batch training procedure to approximate expected gradients as well. During training, we let k be the number of samples we draw to compute expected gradients for each mini-batch. Remarkably, as small as $k = 1$ suffices to regularize the explanations because of the *averaging effect* of the expectation formulation over many training samples. This choice of k uses every sample in the training set as a reference over the course of an epoch, with only one additional gradient call per training step. This lets us efficiently regularize expected gradients using the entire training dataset as the reference – as opposed to the single reference in integrated gradients – with far fewer gradient calls per epoch.

2.7.1 A pixel attribution prior improves robustness to image noise.

Prior work on interpreting image models focused on creating *pixel attribution maps*, which assign a value to each pixel indicating how important that pixel was for a model's prediction [111, 130]. Attribution maps can be noisy and difficult to understand due to their tendency to highlight seemingly unimportant background pixels, indicating the model may be vulnerable to adversarial attacks [105]. Although we may prefer a model with smoother attributions, existing methods only post-process attribution maps but do not change model behavior [45, 111, 119]. Such techniques may not be faithful to the original model [59]. In this section, we describe how we applied our framework to train image models with naturally smoother attributions.

To regularize pixel-level attributions, we used the following intuition: neighboring pixels should have a similar impact on an image model's output. To encode this intuition, we chose a total variation loss on pixel-level attributions:

$$\Omega_{\text{pixel}}(\Phi(\theta, X)) = \sum_{\ell} \sum_{i,j} |\phi^{\ell}_{i+1,j} - \phi^{\ell}_{i,j}| + |\phi^{\ell}_{i,j+1} - \phi^{\ell}_{i,j}|.$$

We applied this pixel smoothness attribution prior to the MNIST and CIFAR-10 datasets [72, 75]. On MNIST we trained a two-layer convolutional neural network; for CIFAR-10 we trained a VGG16 network from scratch [117]. In both cases we optimized hyperparameters for the baseline model

without an attribution prior. To choose λ, we searched over values in $[10^{-20}, 10^{-1}]$ and chose the λ that minimized the attribution prior penalty and achieved a test accuracy within 10% of the baseline model. Figures 2.4 and 2.5 display expected gradients attribution maps for both the baseline and the model regularized with an attribution prior on 5 randomly selected test images on MNIST and CIFAR-10, respectively. In all examples, the attribution prior yields a model with visually smoother attributions. Remarkably, in many instances smoother attributions better highlight the target object's structure.

Recent work has suggested that image classifiers are brittle to small domain shifts: small changes in the underlying distribution of the training and test set can significantly reduce test accuracy [100]. To simulate a domain shift, we applied Gaussian noise to images in the test set and re-evaluated the performance of the regularized and baseline models. As an adaptation of [106], we also compared the attribution prior model to regularizing the total variation of gradients with the same criteria for choosing λ. For each method, we trained 5 models with different random initializations. In Figures 2.4 and 2.5, we plot the mean and standard deviation of test accuracy on MNIST and CIFAR-10, respectively, as a function of standard deviation of added Gaussian noise. The figures show that our regularized model is more robust to noise than both the baseline and gradient-based models.

Both the robustness and more intuitive saliency maps our method provides come at the cost of reduced test set accuracy (0.93 ± 0.002 for the baseline vs. 0.85 ± 0.003 for pixel attribution prior model on CIFAR-10). The trade-off between robustness and accuracy that we observe is consistent with previous work that suggests image classifiers trained solely to maximize test accuracy rely on features that are brittle and difficult to interpret [59, 135, 145]. Despite this trade-off, we find that at a stricter hyperparameter cutoff for λ – within 1% test accuracy of the baseline, rather than 10% – our methods still achieve modest but significant robustness relative to the baseline.

2.8 Discussion

In this section we discussed work extending existing feature attribution methods. The first section introduced the expected gradients method and uncovered connections between distributional sampling and prior work on integrated gradients. The second section offered extensions of attribution

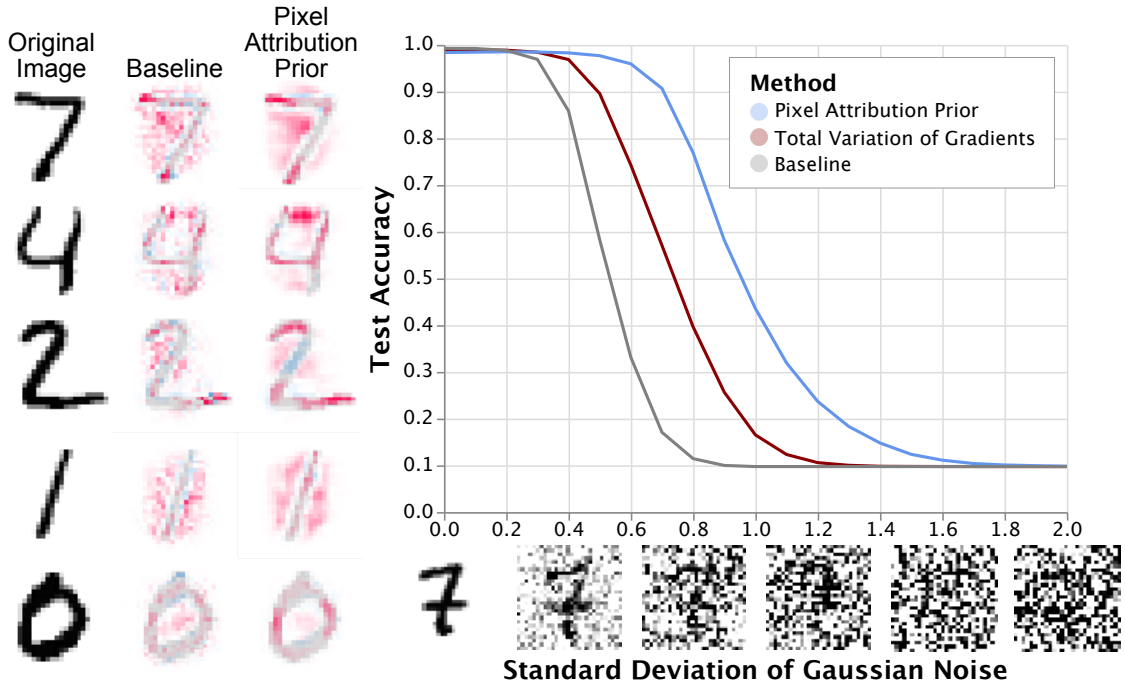


Figure 2.4: Left: Expected gradients attributions (from 100 samples) on MNIST for both the baseline and attribution prior models. The latter achieves visually smoother attributions, and it better highlights how the network classifies digits (e.g., the top part of the 4 being very important). Unlike previous methods which take additional steps to smooth saliency maps after training [45, 119], these are *unmodified* saliency maps directly from the learned model. Right: Inducing smoothness in saliency maps leads to robustness to input noise without specifically training for robustness.

methods to axiomatic interaction methods, and applied them to large language models. The third section described a training procedure to incorporate feature attribution methods into training and make more robust and interpretable models.

More generally, the goal of the work in this section is to understand how to go beyond existing feature attribution methods, both in terms of what they are used for and deeper understanding of model behavior beyond first-order terms. The potential impact of understanding second order terms is broad - almost every modern machine learning task involves interactions between features, which is precisely why deep learning models outperform linear models on said tasks. Being able to impose

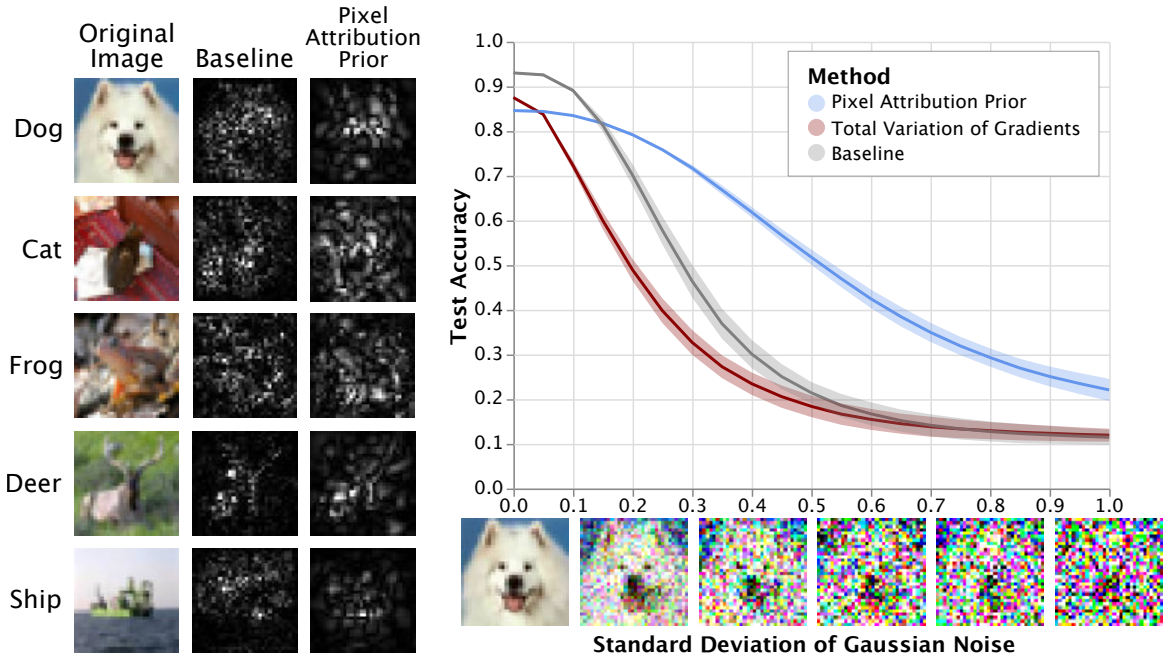


Figure 2.5: Left: Expected gradients attributions (from 100 samples) on CIFAR10 for both the baseline model and the model trained with an attribution prior for five randomly selected images classified correctly by both models. Training with an attribution prior generates visually smoother attribution maps in all cases. Notably, these smoothed attributions also appear more localized towards the object of interest. Right: Training with an attribution prior induces robustness to Gaussian noise, achieving more than double the accuracy of the baseline at high noise levels. This robustness is not achievable by choosing gradients as the attribution function.

penalties on how attributions behave is also impactful - it provides a framework to incorporate more human-level priors that are otherwise intractable to encode into model parameters.

Although the game theoretic foundations of existing feature attribution methods provide a strong framework for understanding existing models, feature attribution is still in and of itself a limited lens. There exist many other definitions of model interpretability in this space - from interpreting concepts rather than features to disentangling dimensions in a latent space - so the work here only encompasses a very narrow definition of what it means to interpret a model. Nonetheless, this chapter has also shown that this limited framework is still rich with open questions and avenues for research, and can provide useful insights into otherwise uninterpretable models.

Chapter 3

PROTEIN LANGUAGE MODELING

3.1 Motivation

The steady progress seen by applying large transformers to natural language processing has inspired interest in applying those same models to large protein sequence datasets [89, 98, 101, 138]. Indeed, a comparison between modeling natural language and protein sequences makes some sense: there are large corpuses of unlabeled sequence data in both domains, and that sequence data also has a governing, underlying structure, be it through meaning and syntax or physical structure and evolutionary constraints. However, there are also clear differences between domains. Natural language has far more tokens than the standard 20 amino acids, and each token or word usually has more syntatic meaning than a single amino acid. There is no apriori obvious way to tokenize protein sequences at the same granularity as language. Furthermore, the goal of modeling protein sequences is generally more "out-of-domain" than modeling language: a model that has memorized all written texts has essentially solved the language modeling problem, but the penultimate goal of de novo protein design is to move beyond naturally-evolved protein sequences.

In this chapter, we take a brief overview of some work done in exploring the choices made during protein language modeling. In particular, we explore the pre-training task and discuss alternative choices of pre-training tasks that are more domain inspired than masked language modeling or autoregressive modeling. We note that this is only an exploratory discussion of possible modeling choices: it remains an open question how to best adapt models from natural language to protein sequences and what role, if any, such models may play in applications ranging from biological discovery to de novo design.

3.2 Masked Language Modeling

Masked language modeling is a standard pre-training task in natural language processing. In this task, a model takes as input a language sequence with some fraction of tokens masked and is trained via cross-entropy to predict the missing tokens. The task requires no sequence labels and takes advantage of the natural structure of language, which informs which words should be in which positions given syntatic and semantic context. Specific hyper-parameters for masked language modeling were largely inspired by the BERT paper, a seminal paper in the field [31].

Masked language modeling has proven to be an extremely useful pre-training task, both from its ease of implementation and from its representational power. However, it is not clear if this task will transfer appropriately to protein sequence modeling. Contacting residues and evolutionary constraints do give rise to contextual clues about which amino acids should be in which positions, but in large portions of the protein many amino acids are equally likely. Furthermore, the distribution observed in natural sequences may not perfectly correlate with experimental measurements like fitness or solubility which are desirable downstream properties to engineer. This motivates sequences models that use alterenative pre-training tasks either as an alternative to masked language modeling or to supplement it. In the next section, we discuss an alternative.

3.3 Profile Prediction

In this section, we introduce a new pre-training task for protein sequence models. Our task is inspired by alignment-based protein structure predictors: the very models that outperform existing deep NLP models at protein structure prediction [40, 69, 87]. Such models take as input features derived from multiple sequence alignments (MSAs), which cluster proteins with related sequences. Features derived from MSAs, such as position-specific scoring matrices and hidden Markov model profiles, have long known to be useful features for predicting the structure of a protein [26, 27, 63, 107]. Our task posits the reverse: that, in order to predict profiles derived from MSAs from a single protein in the alignment, a network must learn information about that protein's structure. Specifically, we propose to use HMM profiles derived from MSAs as labels during pre-training, rather than as input

features in a downstream task. An overview of the task is depicted in Figure 3.1.

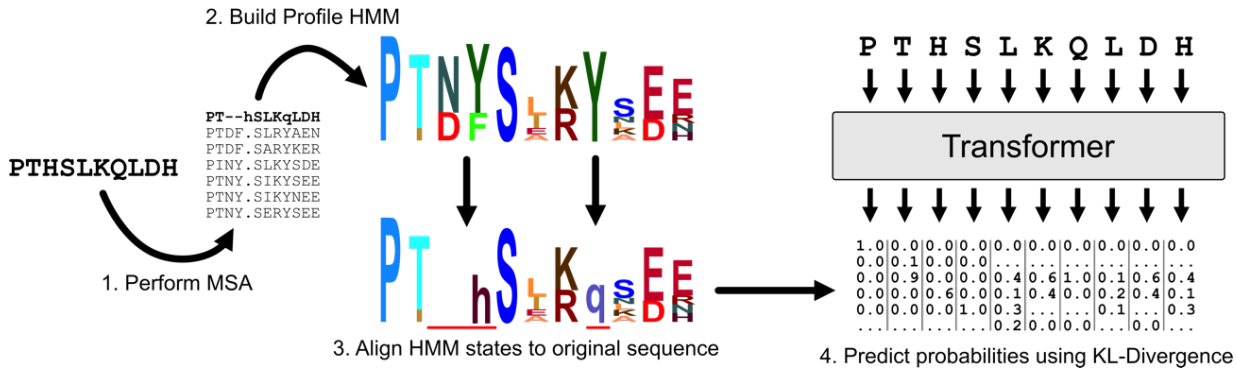


Figure 3.1: An overview of our proposed task. We start with an initial sequence "PTHSLKQLDH". We generate a multiple sequence alignment for that sequence by searching it against a reference database, and we then generate a profile HMM for the alignment. The first H and the Q in our sequence correspond to inserted amino acids that didn't match any columns in the alignment. Therefore, for those amino acids we use insertion state emissions as labels rather than match state emissions. The rest of the amino acids in our sequence were in match states, so we use the match state emission probabilities as labels. Our protein has deletions in two of the match states in the MSA (columns 2 and 3). We omit these from the label since they have no corresponding amino acids as inputs. Finally, we predict the corresponding label using KL divergence, averaged over the length of the sequence.

3.3.1 Multiple Sequence Alignments

Multiple sequence alignment (MSA) is a method for grouping a series of related proteins. MSA arranges proteins in a matrix whose rows are individual protein sequences and whose columns contain amino acids that either come from the same position in some ancestral sequence (homologous), or play a common structural or functional role. Building an MSA is a common first step in many downstream tasks including predicting structure and finding homologous proteins [36]. For our models, we use the pre-training data introduced in [98], which comprises 32 million sequences from Pfam [37]. Pfam further contains pre-built MSAs for each of its entries, grouped into a set

of families. Although it would be possible to build a set of multiple sequence alignments for any protein sequence dataset using standard alignment tools [23, 54, 116], we use the existing multiple sequence alignments from the 32.0 release of Pfam.

3.3.2 Predicting Alignment Profiles

Once an MSA is built, it is common to fit a profile HMM, which model the probabilities of amino acids appearing in the columns of an MSA, as well as the probability of inserting additional amino acids between columns or deleting existing columns. It is common to use features derived from profile HMMs as input to structure prediction models, as these profiles contain information about the evolutionary history of a protein [69]. In particular, the emission probabilities give insight into which positions in the proteins are likely to mutate or remain constant over the course of evolution. This in turn illuminates which portions of the protein are critical for the protein's structure or function. For a review on profile HMMs, see [34]. We build profile HMMs from multiple sequence alignments using HMMER [44] with the default arguments.

Our proposed task is to predict a protein's profile HMM directly from it sequence. Formally, we represent a protein sequence x of length n as $x_1 x_2 \dots x_n$ and the MSA it belongs to as a matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{km} \end{pmatrix}$$

where k is the number of sequences in the alignment and $m \geq n$ is the length of the alignment. Without loss of generality, we assume that x is the first sequence in the alignment; that is, there exists an injective map $g : [n] \mapsto [m]$ such that $i \leq g(i)$ and $x_i = a_{1g(i)}$ for all $i \in [n]$. We let $h : \{a_{ij} \in A\} \mapsto \{M, I, D\}$ be the MSA state function which maps amino acids to the three possible states in an MSA:

1. Match: a_{ij} is an amino acid that is related, evolutionarily or structurally, to other amino acids in column j.

2. Insertion: a_{ij} is an amino acid that is not related to other amino acids in its column, but is more likely the result of a mutation that inserted additional amino acids.

3. Deletion: a_{ij} is not an amino acid, but rather a column in which protein i is missing an amino acid where other proteins in the MSA have amino acids that are either matched or inserted.

A profile HMM built from this MSA is represented by the match state emissions $p_1^M p_2^M \dots p_\ell^M$ and the insertion state emissions $p_1^I p_2^I \dots p_\ell^I$, as well as an injective function $f : [\ell] \mapsto [m]$ which maps the indices of the profile back to the columns of the MSA. p_j^M and p_j^I are probability vectors of size S containing the probability of seeing each amino acid in column $f(j)$ in match or insertions states respectively:

$$\sum_{s=1}^{S} (p_j^M)_s = 1, (p_j^M)_s \geq 0 \text{ and } \sum_{s=1}^{S} (p_j^I)_s = 1, (p_j^I)_s \geq 0$$

for an amino acid alphabet of size S. In our experiments, we use the standard 20 amino acids during profile creation. Note also that f has a well-defined inverse $f^{-1} : [m] \mapsto [\ell]$.

We define a sequence of vector labels, $l_1 l_2 \dots l_n$ associated with the input sequence x, defined as:

$$l_i(x) = \begin{cases} p^M_{f^{-1}(g(i))} & \text{if } h(a_{1g(i)}) = M \\ p^I_{f^{-1}(g(i))} & \text{if } h(a_{1g(i)}) = I \end{cases}$$

The $l_i(x)$ are well-defined: $h(a_{1g(i)}) \neq D, \forall i$ since $g(i)$ only maps to columns in the alignment where x contains amino acids. Given a network function F, our proposal task's loss function can be written as follows:

$$L_{\text{PP}}(x, \theta) = \frac{1}{n} \sum_{i=1}^{n} \sum_{s=1}^{S} l_{i,s}(x)(\log(l_{i,s}(x)) - \log(F_{i,s}(x; \theta))) = \frac{1}{n} \sum_{i=1}^{n} \text{KLDiv}(l_i(x), F_i(x; \theta))$$

where for $F_{i,s}(x; \theta)$ and $l_{i,s}$ the i indexes into the sequence position and the s indexes into the amino acid output probability. The process is described graphically in Figure 3.1. We contrast this loss with the standard masked language modeling objective:

$$L_{\text{MLM}}(x, \theta) = \frac{1}{n} \sum_{i \in \text{mask}} \sum_{s=1}^{S} L_{i,s}(x) \log(F_{i,s}(x; \theta)) = \frac{1}{n} \sum_{i \in \text{mask}} \text{CrossEntropy}(L_i(x), F_i(x; \theta))$$

for one-hot labels $L_{i,s}(x)$ that are equal to 1 if x_i is the sth amino-acid in the vocabulary, and 0 otherwise. We additionally introduce the joint loss:

$$L_{\text{JOINT}}(x, \theta, \lambda) = \lambda L_{\text{MLM}}(x, \theta) + (1 - \lambda) L_{\text{PP}}(x, \theta)$$

for a scaling parameter λ, which in practice we set such that $L_{\text{MLM}}(x, \theta) \approx L_{\text{PP}}(x, \theta)$ over the course of training.

We note here that our task L_{PP} is non-trivial. From an NLP perspective, it is akin to predicting a distribution over possible ways to rephrase a sentence while preserving its meaning from only the original sentence itself. This requires not only knowing which words carry the meaning of the sentence but also knowing the synonyms of these words in the context of that sentence. Doing so would require a significant understanding of language. As such, we believe our task encourages the transformer to learn about underlying protein biology more than simply predicting masked-out amino acids.

3.3.3 Data

To evaluate our pre-training task we make use of the TAPE benchmark: a set of five standardized protein sequence prediction tasks with associated datasets plus a large unlabeled pre-training dataset derived from Pfam [98]. We build labels for the pre-training data set using the procedure described above. We then evaluate our pre-trained models on the five downstream TAPE tasks: secondary structure prediction [69], contact prediction [4], remote homology detection [56], fluorescence prediction [110], and stability prediction [104], using the metrics specified by TAPE.

3.3.4 Hyperparameters and Training Details

For all experiments we train the default transformer architecture used by [98], but we note that our pre-training task is not architecture-specific. We train three models with the three different objectives above. The profile prediction model used a learning rate of 0.00025, while the multi-task and masked language modeling models use a learning rate of 0.0001. These learning rates

represented the largest learning rates that did not cause the model to diverge during the course of training, searching from 0.00001 in increments of 0.00005. All models were pre-trained for 34 epochs.[1] The learning rate uses a warm-up schedule and dynamic batch sizing, both of which are described in [98]. Pre-training a single model took approximately two weeks with 8 NVIDIA Tesla V100 GPUs.

Training details for all downstream tasks follow the procedure laid out by [98]: namely, a learning rate of 0.0001 with linear warm-up schedule, the Adam optimizer and backpropagation through the entire pre-trained model. The downstream prediction heads all follow those in [98], except for contact prediction which uses a single linear layer rather than a 30-layer convolutional architecture.[2]

3.4 Results

3.4.1 Comparing Pre-training Tasks

We first compare our pre-training task against masked language modeling and the multitask model which combines both tasks, keeping hyperparameters and architecture fixed. The results are shown in Tables 3.1, 3.2, and 3.3. For both structure prediction tasks—secondary structure and contact prediction—profile prediction pre-training outperforms multitasking, which in turn outperforms masked language modeling. All three tasks outperform the same model that was not pre-trained. Although it is not surprising that profile pre-training outperforms mask language modeling on structure prediction—namely because HMM profiles are known to contain information relevant to a protein's structure—the differences between the evaluated models are not large. This may mean that potentially more than just a new pre-training task is needed to continue to improve structure predictors, such as different architectures, or larger pre-training datasets [40].

The remote homology detection task demonstrates the largest gap between profile prediction and mask language modeling. The model pre-trained with profile prediction is about 2 to 3 times

[1] We cut off pre-training at 34 epochs due to time constraints. It is possible that further gains may result from continuing to pre-train for longer.

[2] This new linear prediction head is the default in the current version of the `tape-proteins` package.

Pretraining Task	Secondary Structure			Contact Prediction
	CASP12	CB513	TS115	CASP12
Profile Prediction	**0.71**	**0.74**	**0.77**	**0.33**
Masked Language Modeling	0.67	0.72	0.75	0.24
Multi-Task	**0.71**	0.72	0.76	0.28
No Pretraining	0.67	0.64	0.68	0.05

Table 3.1: Comparing Downstream Performance on Structure Prediction

Pretraining Task	Fold	Superfamily	Family
Profile Prediction	**0.23**	**0.45**	**0.86**
Masked Language Modeling	0.14	0.16	0.45
Multi-Task	0.16	0.28	0.68
No Pretraining	0.05	0.05	0.21

Table 3.2: Comparing Downstream Performance on Homology Detection

more accurate than the model pre-trained using masked language modeling. The performance of the multitask model lies between that of the other two models and all three again outperform a randomly initialized model. This may be because HMM profiles also contain significant amounts of information about evolutionarily related proteins, which is closely related to the structural or functional groupings that a protein falls into.

We find the same pattern on the fluorescence task: profile prediction leads to the best test set performance, followed by multitasking, masked language modeling and no pre-training in that order. Finally, on the stability task, the masked language modeling model and the multitask model both outperform profile prediction. This may be because this task tests models' ability to generalize to proteins with a single amino acid difference from proteins in the training set—a task that masked

Pretraining Task	Downstream Task	
	Fluorescence	Stability
Profile Prediction	**0.38**	0.55
Masked Language Modeling	0.25	**0.63**
Multi-Task	0.28	**0.63**
No Pretraining	0.04	0.45

Table 3.3: Comparing Downstream Performance on Engineering Tasks

language modeling is particularly suited for. Taken as a whole, these results indicate that there may not be a one-size-fits-all pre-training task for all downstream prediction tasks. Rather, it may be beneficial to tailor the pre-training task to the downstream task: for structure or evolutionary tasks, incorporating profile information may be beneficial, but for fine-grained engineering tasks, masked language modeling may be a better choice.

3.4.2 Comparing Against TAPE Benchmarks

On the secondary structure task, we fall short of the NetsurfP2.0 model presented by [69], which is the alignment baseline from [98]. We also fall short of the LSTM and ResNet models from [98], but outperform both the Transformer model as well as all previous work proposing protein-specific pre-training tasks [13, 83], and a recent paper employing an auto-regressive LSTM [3]. On the remote homology task we outperformed all existing models except the TAPE benchmark's LSTM model and the LSTM presented by [3]. We again note that we outperform the protein-specific pre-training tasks in [13] and [83]. In the engineering tasks, all of our models are outperformed by existing work, including models from the original TAPE benchmark and the related works mentioned above.

We also note that we do not view most existing work as mutually exclusive with ours. Instead, we believe that combining our task with the architectures and pre-training tasks present in existing

	Method	Structure	Evolutionary	Engineering	
		SS	Homology	Fluorescence	Stability
Pre-trained models from [98]	Transformer	0.73	0.21	**0.68**	**0.73**
	LSTM	0.75	**0.26**	0.67	0.69
	ResNet	0.75	0.17	0.21	**0.73**
Baselines from [98]	One-hot	0.69	0.09	0.14	0.19
	Alignment	**0.80**	0.09	N/A	N/A
	[13]	0.73	0.17	0.33	0.64
	[3]	0.73	0.23	0.67	**0.73**
	[83]	0.70	0.13	**0.68**	0.68
Our models	Profile Prediction	0.74	0.23	0.38	0.55
	Masked Language Modeling	0.72	0.14	0.25	0.63
	Multi-Task	0.74	0.16	0.28	0.63

Table 3.4: Results on downstream supervised tasks

work may lead to further performance gains.

3.5 Seq2MSA

There has also been growing interest in using protein language models generatively [43, 89]. Although demonstrating unconstrained generative capacity is impressive, it remains an active area how best to use such models for protein design and what form conditioned or controlled input should take. The main approaches in language modeling, namely prompting and reinforcement learning with human feedback [21, 82], are non-trivial to apply to protein sequences. Unlike language, humans - even expert biologists - do not have strong priors about the right way to prompt protein language models for design tasks, particularly tasks involving novel function. It is also not possible

to simply look at a sequence or a structure and say if it accomplishes the desired function or not. The requisite experiments required to screen for function are costly and time consuming, and are difficult to perform at the scales necessary to fine-tune large language models.

As an alternative, we consider the problem of diversification from a generative, language modeling perspective. Instead of generating novel sequences completely from scratch, diversification aims to take a given sequence and generate related sequences or sequences that have similar structure and function. Such a model could be prompted not by human input but rather by an existing natural sequence known to have a desired function, or function similar to that desired. The model could subsequently be used to model diversity around that input sequence.

Modeling natural diversity around a target sequence has broad relevance for experimental and computational design of proteins. Generative models of protein sequences have been shown to provide rational single-site mutations that improve the efficiency of directed evolution [53]. More generally, modeling the space of natural sequences in a family has the potential to not only improve the efficiency of protein design, but also may help design trajectories escape local optimization minima by recapitulating useful modifications learned from evolutionary history. The ability to model the evolutionary landscape around a given sequence allows protein designers to search the manifold of natural sequences to optimize an objective function, while avoiding 'adversarial' parts of sequence space that may be arrived at via unconstrained search.

We present a novel protein language model, Seq2MSA, which is capable of consistently generating highly diverse sequences (sequence identity $< 40\%$) from an initial seed sequence while maintaining structural similarity. The Seq2MSA model is trained with a novel task where the model is provided with a seed sequence and is trained to generate sequences from a Multiple Sequence Alignment (MSA) built from that seed. We validate the model's generations on held out proteins both using a single-sequence folding model as an oracle and aggregation scores derived from rosetta [2]. In addition, we confirm that a significant portion of generated sequences are novel both with respect to the entire pool of generated proteins and the language model's entire training set.

3.5.1 Related Work

There exist many wet-lab methods for sequence diversification, often based around single or multi-codon mutagenesis at the DNA level [9,81,93,108,114,143]. Such methods often consist of multiple rounds of mutation and functional screening and can be both time consuming and expensive.

More closely related to this work, there have been several recent computational approaches for protein sequence diversification, including using a large autoregressive protein language model fine-tuned on specific protein families [88], a sequence-to-sequence denoising model [48], and a fitness prediction model combined with multiple iterations of single-site mutations [17]. Unlike existing methods, our model: (1) is general purpose for any protein sequence, and does not need to be fine-tuned on specific families, (2) allows for arbitrary insertions and deletions rather than just single-site mutations and (3) creates a structurally similar but not identical ensemble of sequences. This latter point is important in comparison to inverse-folding methods, which may generate an ensemble of sequences that fold to a specific backbone, but do not allow any flexibility or diversity in that backbone [29,57].

More broadly, there is a wide range of recent literature on training large language models on protein sequences, including [14, 39, 89, 98, 102]. In particular, both [88] and [43] train large autoregressive models to generate protein sequences unconditionally and conditioned on functional tags. In our work, we focus specifically on generating proteins within a family of a given seed protein. Related to our task, [99] develop a transformer trained on MSAs. Unlike this work, our model does not directly take an MSA as input, and instead seeks to generate sequences from the MSA as output. As a result, our model does not require MSA generation at inference time.

3.5.2 Methods

Here we describe the core task and pipeline we use to generate diverse sequence ensembles. Training and architectural details, as well as datasets used, are described in Sections 3.5.3 and 3.5.4.

Multiple Sequence Alignment

A multiple sequence alignment (MSA) is a core object in biological sequence analysis that groups related proteins in a matrix whose columns represent homologous or structurally related amino acids. In brief, a multiple sequence alignment is typically created by searching a "seed" sequence against a large database of reference sequences using a search algorithm such as HHBlits [124]. The search tool returns target sequences that are *aligned* to the seed. For each index in the seed sequence, the target sequence can either have a corresponding match state residue at that index, or a gap token indicating that the target sequence is missing a residue at this index. Additionally, residues in the target sequence that do not match to any index in the seed sequence are called inserted residues. See [20] for a more complete overview.

The Seq2MSA Task

Our proposed task is to generate sequences in an MSA given the seed sequence that was used to generate that MSA. Formally, given a seed sequence and members of its corresponding MSA, at training time we sample a single target sequence from the MSA uniformly at random. The loss at each iteration is the cross entropy between the target sequence and the model output given the seed sequence, where cross entropy is computed over all tokens equally [137].

We tokenize the seed sequence following the amino acid alphabet from [102]. There are two variants of the Seq2MSA task depending on how the target sequence is tokenized. In the first variant, called *unaligned*, we remove all alignment information from the target sequence: we discard gap tokens and treat insertions the same as match states. In the second variant, called *aligned*, we maintain the alignment information given in the multiple sequence alignment. We double the vocabulary size by distinguishing between match state and insertion state residues in the target sequence, and add an additional token for gaps. We compare the two variants in Section 3.5.5; we generally find that using alignment information produces sequences that have higher predicted structural stability and use this variant of the task for all experiments below.

Diversification Pipeline

We predict the structure of each protein in our evaluation set using a single-sequence folding model as an oracle [78]. We discard proteins that have a pTM < 0.8. For the remainder of proteins, we generate 2k proteins using the Seq2MSA model. During generation, we sample directly from the model distribution without using strategies like nucleus or low-temperature sampling to improve quality at the cost of diversity [139]. We find that prompting the model with the first five residues of the seed sequence reduces the prevalence of generations consisting of only small aligned fragments, and do so for all experiments. We filter generations for pTM > 0.8 and TM-score between the predicted structures of the seed and generated sequences (TM-to-seed) > 0.8. We then sort the remaining sequences by increasing sequence identity percentage to the seed, where we define sequence identity percentage using definition 3 from [92]. We keep the top 100 proteins farthest away from the seed and discard the remaining generations.

3.5.3 Training Setup

Our training architecture is a 12-layer encoder-decoder sequence-to-sequence transformer [137] with architecture parameters taken from [77]. We trained for a total of 250k steps with a batch size of 2^{21} tokens per step. Data-parallel training was performed on a total of 128 Nvidia Volta GPUs, and took approximately one week. We used an initial learning rate of $3e^{-4}$ with a linear warmup from 0 for the first 2k steps, and then a polynomial decay to $4e^{-4}$ by the end of training.

3.5.4 Datasets

We create a large training dataset of MSAs by searching every protein sequence in the 2020 release of UniRef50 [132] against every sequence in (the 2020 release of) Uniref90 using HHBlits [124] with the default parameters. From each MSA we randomly selected a maximum of 1024 sequences to be part of the training set due to disk space constraints.

As an evaluation set, we use the proteins from the 04/01/22-06/25/22 release of the CAMEO [103]. The structures for these proteins are held-out from our structure prediction oracle. We predict

the structure of each protein and discard those proteins with predicted TM-score [146] (pTM) < 0.8, which has generally been found to be indicative of good performance for single sequence folding models [78, 144]. This leaves an evaluation set of 117 sequences from both easy and medium difficulties.

3.5.5 Generation Set Up

Figure 3.2 compares the effects of training a model with alignment information vs. without, as well as prompting the model at decoding time with the first five residues of the seed sequences vs. sampling with no prompting. As expected, adding alignment information into the model generally improves the predicted TM-score of the sequences under our folding oracle, which is a proxy for how well the sequence folds in vivo [78]. Sequence alignment also helps the generated sequences better align structurally to their original seed. Fixing the first five residues at decoding time also increases both structural metrics, at the cost of increasing the percent sequence identity to the seed. Since our model was trained on very diverse MSAs (mean sequence identity to the seed 20% since many of the sequences found in natural MSAs are fragments that only match portions of the original seed), prompting is an ad hoc fix to keep generations slightly closer to the seed. A more complete solution would re-train our original model on MSAs that are closer to their original seed sequence.

3.5.6 Full Counts on the Evaluation Set

Table 3.5 and Figure 3.3 shows a more complete picture of our diversification efforts, on each of the proteins in the evaluation set. The figure shows an interesting and sharp tail off: for some sequences, we are able to generate many (> 100) sequences that have high structural confidence under a folding oracle and low sequence identity percentage to the original seed sequence. For many others, none of the 2000 generated proteins per seed fulfils this criteria. It is not immediately obvious why certain proteins are “harder” than others to diversify - the success and failure cases both contain a variety of structures (helix only, sheet only, and mixed helix/sheet), as well as a variety of lengths.

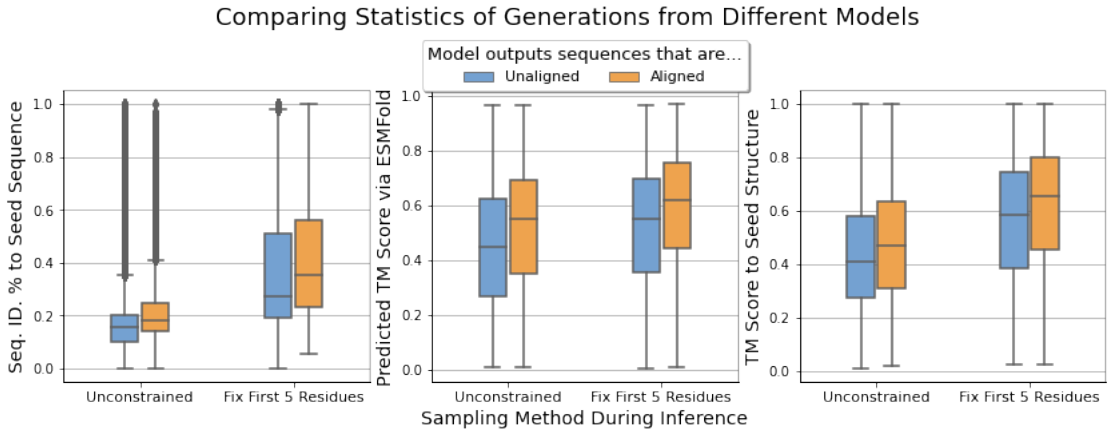


Figure 3.2: A comparison of different generation methods, and different ways of training the original Seq2MSA model. Adding alignment information and fixing the first five residues at decoding time increase confidence in the predicted structures of the generated sequences, and the structural alignment between the generated sequences and the seed used to generate them.

3.5.7 Agreement Between Folding Methods

In order to increase the confidence in our sequences folding to their predicted structure, we take the top 100 generated sequences for each seed sequence (that pass pTM > 0.8 and TM-to-seed > 0.8 under our single sequence model) and fold it using AlphaFold 2 [64] in single sequence mode: no templates and no MSA. AlphaFold in single sequence mode leverages far less evolutionary information than our single sequence model, since AlphaFold gets that information from MSAs while our single sequence model memorizes evolutionary information during language model pre-training [78]. The top of Figure 3.4 shows the results. In general, if AlphaFold 2 in single sequence mode folds the seed sequence well, it tends to also have high pLDDT of the generations, which adds confidence that our generated sequences really do fold to their predicted structures and are not adversarial examples leveraging MSA information.

For the sequences generated by our Seq2MSA model that satisfy both pLDDT under AlphaFold > 80 and pTM under our single sequence model > 0.8, we compare the TM-score between structures predicted by AlphaFold in single sequence mode and structures predicted by our single sequence

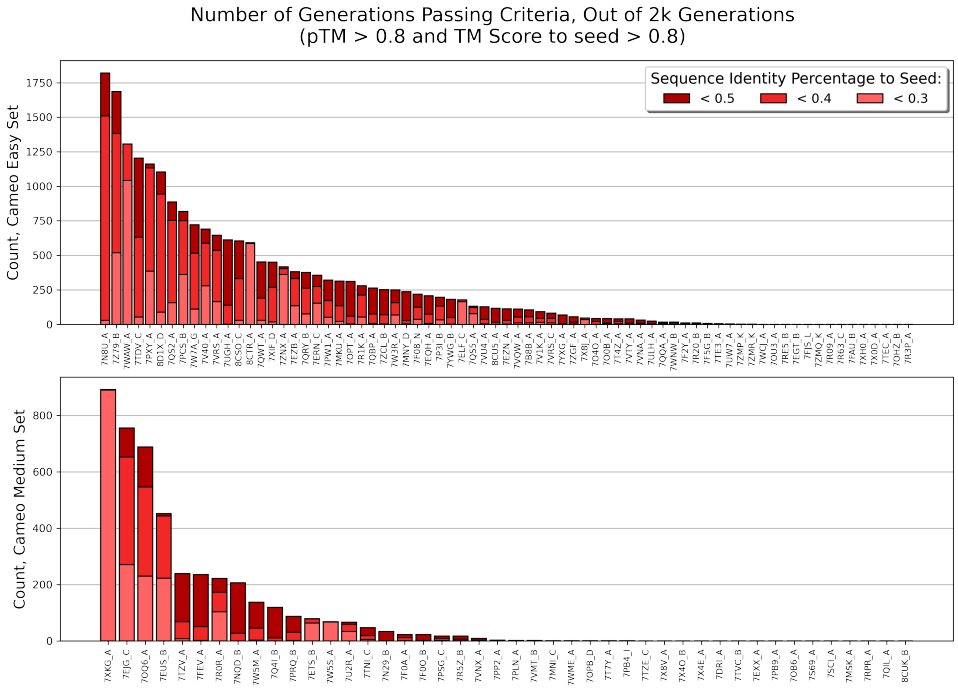


Figure 3.3: The number of generated sequences passing specified diversification criterion, out of 2000 generations per seed sequence. The seed sequences in the evaluation seed have PDB ids that are listed (ID and chain) on the x axis. Our model successfully generates 100s of structurally similar structures with diverse sequences for some proteins, and fails to generate any for others.

model. The result is in the bottom of Figure 3.4. The histogram shows that the majority of high-confidence predictions by both models also have high structural agreement, further raising confidence in the predicted structures of each generated sequence.

3.5.8 Hallucinations

In order to test whether our pipeline could generalize to de novo sequences, we download the hallucinated sequences from [8]. We select only those sequences that were experimentally characterized by size-exclusion chromatography, and further filter by folding oracle pTM > 0.7. We reduce the cutoff from 0.8 since only two hallucinations passed pTM > 0.8. We then run the same pipeline from the main text on each denovo protein. The results are in Figure 3.5. The plot clearly shows that de novo proteins are harder than natural proteins - there are far fewer generations per sequence satisfying the diversification criterion. However, the plot also shows that for many de novo sequences, we are able to generate diverse ensembles at the < 0.5 and even < 0.4 sequence identity level.

3.5.9 Results

Using the diversification pipeline outlined in the previous section, we find that: (1) we are able to diversify a broad range of structurally held-out sequences down to as low as < 30% sequence identity, (2) these sequences are not only diverse relative to the original sequence but also to the model's entire training set, and (3) our model's best generations not only align well to the structure of the original protein, but also show similar characteristics on a range of Rosetta-based aggregation metrics.

Structural Similarity

Under our folding oracle, our Seq2MSA models generates at least 10 sequences with pTM > 0.8 and TM-to-seed > 0.8 for 103 out of the 117 seed proteins, and over 100 such sequences for 78 of the 117 seed proteins. This indicates that our diversification pipeline is able to generate high quality structures for a majority of proteins it was run on. Moreover, it indicates that our language model

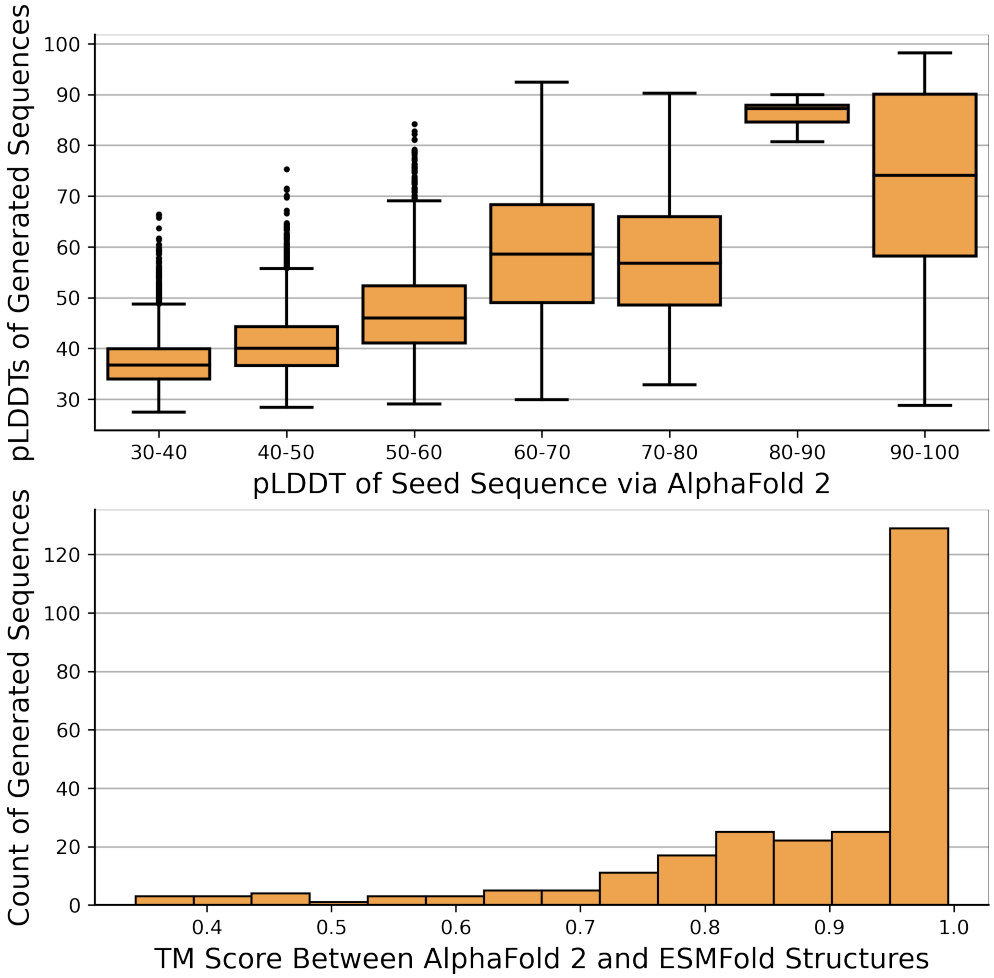


Figure 3.4: Validating our single-sequencing folding oracle by comparing it against AlphaFold 2 in single sequence mode. It is possible that our sequences simply "look" like natural sequences in a family because they have similar distributions and placements of residues to the MSAs they were trained on, and that our single sequence folding oracle is predicting their structure solely based on these family features rather than what the sequence would actually fold to in vitro. However, we find that not only can AlphaFold 2 in single sequence mode fold many of our generations just as well as it folds the natural seed protein used to generate them, we also find that the structures between AF2 and the single sequence oracle have high agreement when AlphaFold 2 has pLDDT > 80. This gives confidence that our sequence structures are in fact stable and structurally similar to their original seeds.

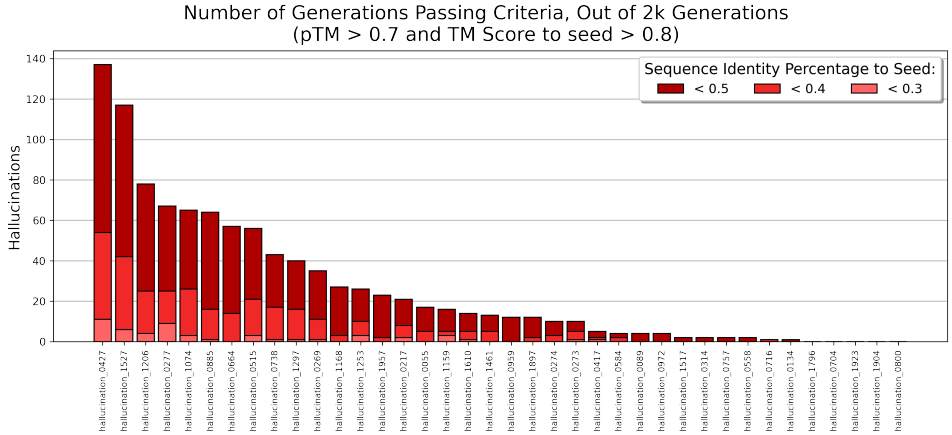


Figure 3.5: The number of generated sequences passing diversification criterion, when applying the Seq2MSA model on completely de novo proteins. Remarkably, even though our model was trained only on natural proteins, it manages to succesfully generate diverse ensembles for many of the de novo designs.

can generate sequences that preserve the structure of the input seed sequence.

To gain more confidence in the predicted structures of our generated sequences, we compute two metrics of stability for our generated proteins and compare them to original seed sequence: the SAP score and the hydrophobic solvent accessible surface area (SASA). Both metrics are based on the number of hydrophobic residues exposed at the surface of the protein and indicate aggregation propensity of the protein. See [25] for a complete definition. Figure 3.6 shows that both metrics correlate well between each seed protein and its respective generations, and also indicate that the generated proteins tend to expose only a small proportion of hydrophobic residues, which we would expect of proteins with clearly defined secondary structure elements. We run additional sanity checks of our single-sequence folding oracle against AlphaFold 2 (AF2) in Section 3.5.7 - in general we find high agreement between our oracle and AF2 in single sequence mode.

Sequence Diversity

Out of 117 proteins, our pipeline is able to successfully generate diverse sequence ensembles for more than half of them: counts are in Table 3.5 and Figure 3.3. For comparison, several recent

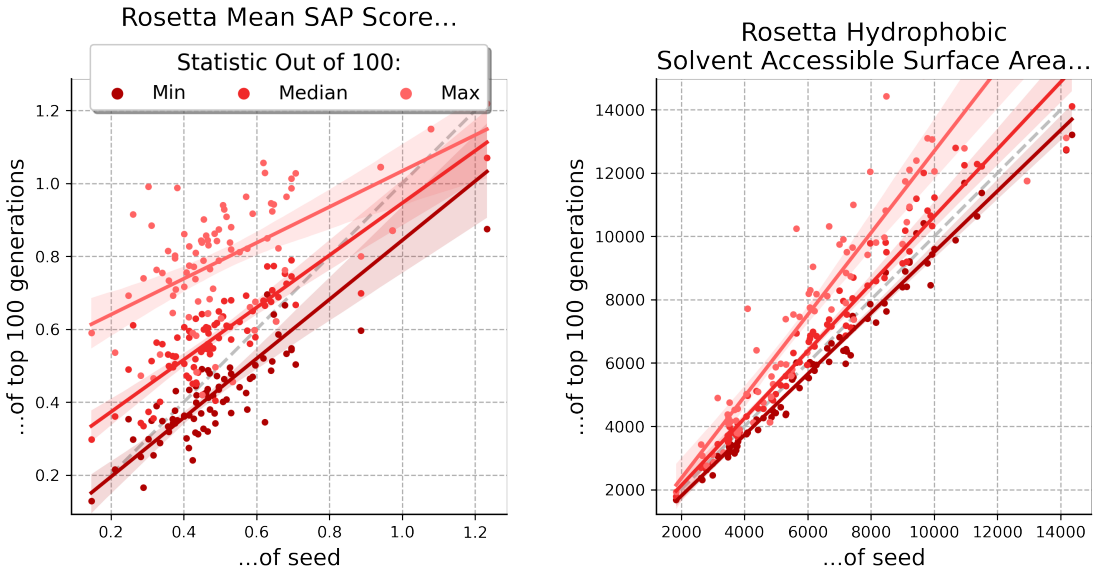


Figure 3.6: We run the diversification pipeline described in Section 3.5.2, which leaves 100 generations per seed sequence that satisfy pTM > 0.8 and TM-to-seed > 0.8. We sort the generations under each seed by the respective metric, and plot points for the minimum, median and maximum values on the y axis. The generated sequences share similar characteristics to their seed sequence, indicating the generated sequences will have stable structures similar to their respective seed sequences.

methods for diversification using deep models fail to generate any functional sequences at this sequence identity range [48, 88]. Figure 3.3 depicts a more detailed breakdown of the number of sequences that pass our computational criteria. Figure 3.7 depicts a representative example: an anti-cD40 DARPin protein domain (pdb id: 7P3I_2, chain B). In particular, this example demonstrates that the generated proteins preserve major secondary structure elements, but contain diversity in the variable loop regions, including variations in both length and placement.

To more stringently verify that our sequences are novel, we perform three sanity checks. First we verify that the sequence identities computed by our aligned model are close to the optimal alignment. Second, we verify that each generated sequence is diverse with respect to the entire pool of generated sequences. Third, we verify that each generated sequence is far away from any protein the language model has seen during training time.

Sequence Identity Cutoff	$\leq 20\%$	$\leq 40\%$	$\leq 60\%$	$\leq 80\%$	$\leq 100\%$
Count	12	63	81	100	103

Table 3.5: The number of sequences in the evaluation set for which our model generates at least 10 sequences (out of 2000) that satisfy pTM > 0.8, TM-to-seed > 0.8 and the given sequence identity cutoff, out of 117 total sequences in the evaluation set.

We compute sequence identities with respect to the model's output alignment, but there may exist a better alignment with a higher sequence identity percentage. After running the diversification pipeline, we re-align each of the 100 generations to their seed using the pairwise aligner *needle* [90]. The left plot in Figure 3.8 demonstrates close agreement between the model's output alignment and the optimal pairwise alignment.[3] We then take the same 100 generated sequences from the diversification pipeline above and compare them to all 2k sequences originally generated for that seed in the initial step of the pipeline. The middle plot in same figure demonstrates that if a generated sequence is far from the seed used to generate it, it will also be far in sequence space from any other generated sequence. Finally, we use *blastp* to search every generated sequence against Uniref90 - the model's training set - and report the maximum sequence identity for each generation to any sequence in that set [5]. The right plot in the same figure indicates that if a generated sequence is far from the seed used to generate it, it will also be far away from any protein the model has seen during training.

3.6 Discussion

This chapter has discussed two alternative pre-training tasks for protein language with two different end goals: representation learning and protein design. The results from this chapter indicate that thinking more carefully about pre-training tasks on protein sequences, particularly tasks that take advantage of domain-specific structure, may yield improved models at both prediction and

[3]Note that the optimal pairwise alignment does not mean the alignment with the greatest sequence identity percentage, so the model may learn alignments that have greater percent identities than the optimal one.

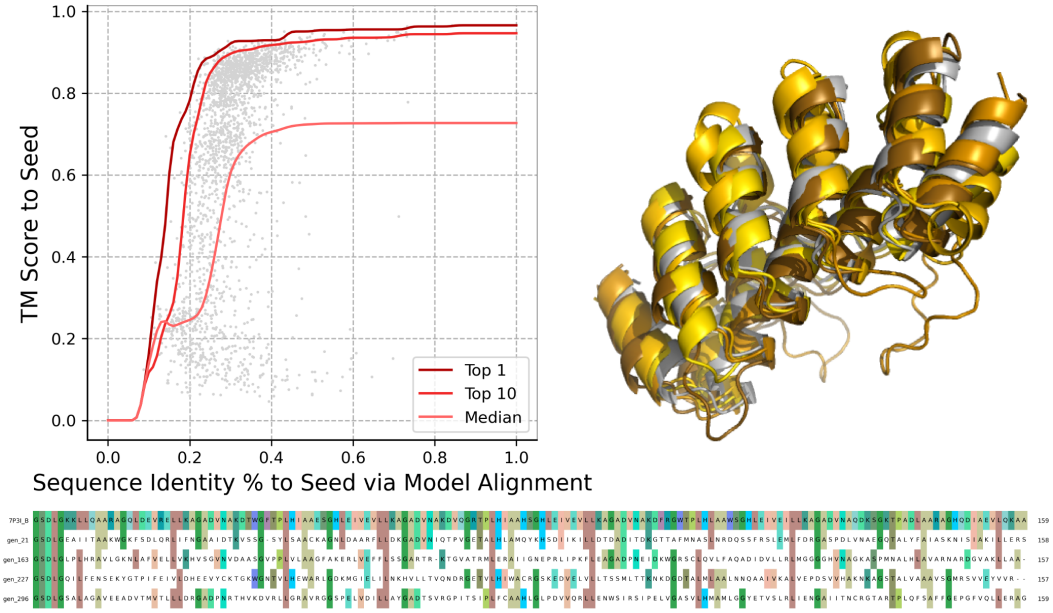


Figure 3.7: The tradeoff between sequence identity and structural alignment to the original seed using the Seq2MSA model. The lines indicate the TM-score of the top-1, top-10 and median generation up to the given sequence identity percentage, and the gray dots are individual generations. Note that the lines are smoothed with a Gaussian filter. The structural ensemble and the corresponding alignment show the top 4 generated proteins with sequence identity < 30% and greatest alignment score to the original protein, which appears as the gray structure and the first sequence in the alignment.

generative capacity. In particular, both tasks here take advantage of the copious existing machinery in protein sequence biology around homology searching and multiple sequence alignment: highly optimized tools to determined which protein sequences are potentially evolutionarily related.

There are many underlying structures in protein sequences beyond just multiple sequence alignment: function, structure, non-polymer binding partners, amino acid properties, conformational dynamics and so on. Incorporating this rich domain knowledge into large sequence models may be the next frontier of large-scale protein modeling, and has the potential to unlock tremendous capacity not only in biological discovery but also in designing new proteins to accomplish completely novel function.

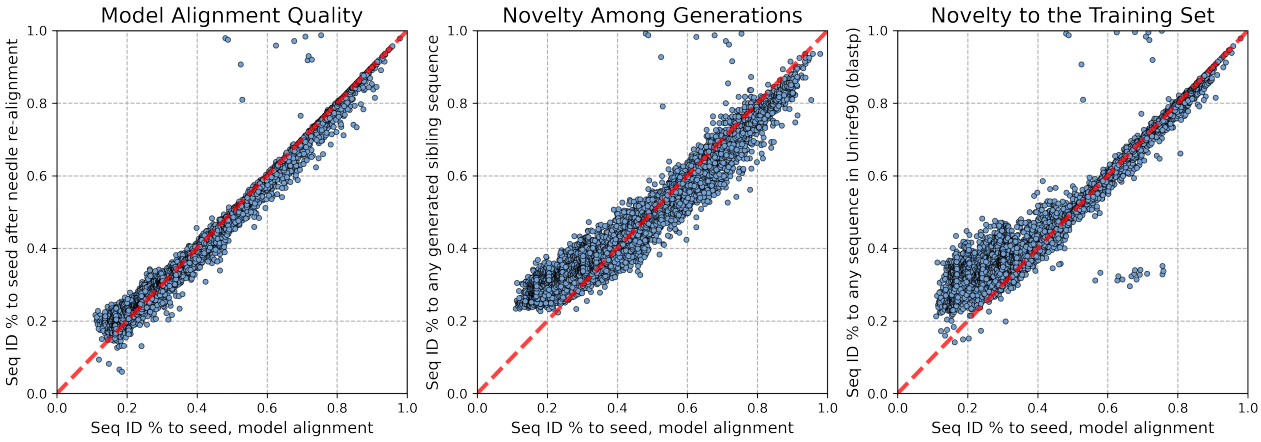


Figure 3.8: A comparison between the sequence identity of generations to their seed sequence with other metrics of novelty. The left plot compares sequence identities between model generated alignments and the optimal dynamic programming alignment. The middle plot shows the percent identities between any generated sibling sequences, e.g. those sequences that were generated by the same seed. The right plot shows percent identities between generations and their closest Uniref90 blastp hit. All three metrics show a clear correlation to the percent identity between generation and its original seed, implying that if a generation is far from its seed sequence it is in fact truly novel.

Chapter 4

PROTEIN-LIGAND CO-FOLDING

4.1 Motivation

Predicting the folded structure of a protein given its amino-acid sequence has been a long-standing problem in structural biology [7]. This partially driven by the disparity between the cost of sequencing and the cost of structure determination: given that sequencing is far cheaper than structure-determination methods like crystallography or nuclear magnetic resonance spectroscopy, sequences databases have been growing exponentially faster than corresponding structure databases [22, 49, 136]. The advent of large structure prediction networks - namely AlphaFold 2 and RoseTTAFold - has demonstrated how useful deep learning can be for biological applications [12, 64]. In particular, advances in deep learning architectures like the transformer combined with a better theoretical understanding of how to model 3D atomic systems and their associated symmetries implies the promise of a host of foundational models in structural biology.

However, deep learning has still fallen short on a host of structural problems: in particular, accurately docking ligands to protein structures remains an ongoing challenge. There exist many deep learning methods for ligand docking, often making use of equivariant graph neural network architectures and taking as context a ligand graph as well as some or all of the target protein [24, 84, 123]. However, these methods can fail to generate chemically-plausible poses and do not always outperform classical, force-field docking tools like Vina [18, 58]. Furthermore, most of these existing methods either treat the protein as static, e.g. "rigid-body docking" or treat the side-chains of the protein as flexible but the backbone as rigid. However, proteins are intrinsically dynamic objects that may change in response to ligand binding partners. A method that fails to account for protein flexibility may fail to dock to proteins with cryptic pockets, flexible side-chains or multiple conformational states.

This motivates the field of protein-ligand co-folding: the problem of jointly predicting the structure of a protein and a ligand in a complex given both the protein sequence and ligand chemistry as input. The formulation is well-motivated: there is ample protein-ligand complex training data in the Protein Data Bank [49]. Although ligands may adopt multiple bound poses against a protein target, for protein-ligand complexes that crystallize the crystal tends to display the lowest energy minima pose. Finally, networks to predict protein structure are already well-studied: therefore it suffices to adapt the input representation of these networks to also include small molecules, rather than have to develop networks entirely from scratch.

In this chapter, I will describe joint efforts to develop a large structure prediction network - RoseTTAFold All-Atom (RFAA) - that can both predict the structures of proteins and simultaneously dock ligand structures to those protein structures. The hope of designing such a network is that it will not only be useful for protein-ligand docking, but will also enable a host of downstream biological applications, from the de-novo design of enzymes to high-throughput in-silico virtual screening. I will discuss how I evaluated the performance of the network relative to existing methods, and some emergent properties that arise from having a deep learning method capable reasoning over both residue-level and atom-level context. I will talk about adapting the model to perform virtual screening and the potential to adapt it for other related downstream tasks. I will then discuss the outlook for the future of the model - where it is relative to the rapidly growing industry surrounding the problem of protein-ligand co-folding, and how we can improve the performance of the model to keep pace with that industry.

4.2 The RoseTTAFold All-Atom Architecture

RoseTTAFold All-Atom shared the same base structure as the original RoseTTAFold architecture [12]. It is a three-track transformer that takes as input a protein sequence and produces as output three dimensions coordinates representing the structure of the system. It optionally takes in multiple sequence alignments and homologous template structures of similar proteins build by standard tools like the HH-Suite [124]. Multiple sequence alignments are a set of similar sequences found by a computational search of a large sequence database and aligned via dynnamic programming

algorithms to the query protein, and have long been known to improve the performance of structure prediction models, likely because the covariance statistics in multiple sequence aligments are indicative of which residues in the protein are physically contacting [27]. The network's three tracks process the multiple sequence alignment, the 2D distance matrix from homologous templates, and an iteratively growing 3D representation of the protein via the attention mechanism [137] and pass information via cross-attention and an attention biasing mechanism similar to those seen in AlphaFold 2.

In more detail, the first track of the network processes multiple sequence alignments using row and column-wise axial attention [55]. The network maintains a (N, L, d_1) tensor of embeddings corresponding to number of sequences, length of the input, and dimensionality, respectively. The row-wise axial attention is biased by the pair attention, which is meant to help the network attend to residues that are close in 3D space. Specifically, the 2D track is projected down to a bias matrix that is added to the attention weights during row-wise axial attention. Layer normalization is applied before each axial attention block [11]. Features from the 3D track, which represent the 3D atomic coordinates of the model, are projected down and added to the first index of the MSA embeddings to share structural information from the 3D track.

The second track of the network processes inter-residue and inter-atom distnaces in an (L, L, d_2) tensor. The features from this track are initialized from the inter-residue distances of a homologous protein template, and at random for ligand atoms. The network attends to the tensor using tied column and row axial attention. The attention is biased by the current predicted coordinates in the 3D track, by binning predicted inter-residue distances using the radial basis function. Triangle multiplication, both ingoing and outgoing, are also applied between attention blocks [64].

The third track is an SE(3)-equivariant transformer that represents the predicted coordinates of the input at each block [46]. The scalar features input to the transformer come from a projection of the concatenated msa features and 3D node features. Chirality for small molecules is provided as a vector-valued feature for all small molecule atoms indicating the direction of idealized tetrahedral geometry. Edge features in the SE(3) graph are derived from the relative positional encoding, a bond matrix indicating which atoms are bonded and how, and the pairwise inter-residue distances.

The graph is fully connected. The SE(3) transformer outputs predicted coordinates and a quaternion for frame orientation.

Unlike proteins, which can be represented as a linear chain of amino acids, small molecules have arbitrary graph structures whose nodes are atoms and whose edges are covalent bonds. In order to represent this new modality, RoseTTAFold All-Atom has an expanded token representation including 47 of the most common heavy atoms seen in the protein data bank [49]. It takes as additional input a bond connectivity graph that represent the connectivity of the input molecules. The bond matrix has four different small-molecule bond types - single, double, triple and aromatic - that get embedded and added to the pairwise features in the model. For proteins, neighboring residues are connected by a special *residue-residue* bond feature. In order to represent rigid frames at each atom - which are necessary in order to apply equivariant structure losses - a neighborhood ranking algorithm is applied to each atom which produces deterministic 3-body frames. The chirality of small molecule tetrahedral centers is represented via 3D coordinate gradients that push the atoms in the local direction of correct stereochemistry.

In addition to 3D coordinate predictions, the model also outputs an estimate of its own error called the Predicted Alignment Error (PAE). The predicted alignment error is an $L \times L$ matrix of values whose i, jth entry is trained to predict the error of atom j in the frame of reference of atom (or residue) i. The PAE is useful at inference time to determine how confidence a model is in its relative placement of each atom or residue - to determine overall confidence in placement between two molecules the inter-PAE values between every residue or atom in one molecule to another is averaged.

A more full description of the architecture, including hyper-parameters like number of layers and embedding dimension for the different tracks, can be found in [71].

4.3 Training Regime

4.3.1 Data

This subsection will discuss the protein-small molecule complexes used to train the model. A full description of the other datasets used and their sampling regimes is available in [71].

Each entry in the protein-small molecule dataset consists of a query ligand and accompanying partner chains, both protein and small molecule. For every given assembly in the PDB, we iterate through every non-polymer chain and treat it as a potential query ligand. A protein chain is considered in contact with that query ligand if any Cα is within 30Å of any atom of the query ligand. A non-polymer chain is considered in contact with that query ligand if all atoms are within the same 30Å radius. All protein partner chains are collected and sorted based on number of Cα atoms within 10å of any atom to the query ligand, and the primary protein chain is designated as the protein chain with the most number of contacts. If there are no protein partner chains the entry is discarded. We further filter out a list of ligands that we find to be non-biological, e.g. representing solvents or buffer agents. A full description of which ligands we filter out can be found in [71].

We cluster each entry in the dataset based on the primary protein partner down to the 30% sequence identity level using mmseqs [52]. At training time, we first sample a protein sequence cluster, and then a unique protein chain partner belonging to that cluster. Finally, we select uniformly at random a non-polymer molecule bound to that protein chain. To avoid out of memory issues, we crop the example around the query ligand using a breadth-first search initiated at a randomly chosen atom in the query ligand. Edge weights for the search are determined by bondedness and spatial proximity, resulting in crops that are both roughly around the ligand but also contiguous sequentially for protein examples.

4.3.2 Losses

At training time, we apply a variety of losses to the model. The primary loss is Frame Aligned Point Error (FAPE), which was first introduced in [64]. FAPE relies on a rigid body frame at each residue, which is defined using the N-Cα-C backbone. For arbitrary small molecules, we

derive a deterministic, path-based procedure for determining frames for losses. The procedure iterates through every path of length 3 containing a given atom and picks the path that comes first in lexicographic ordering as the three-body determining that atom's frame. FAPE is then applied as usual with the given small molecule frames, with the understanding that the frames are updated as the coordinates are.

The remainder of the losses follow from the original RoseTTAFold 2 architecture [12]. We mask out 15% of the tokens in the MSA and predict them using a masked language modeling loss, and predict the torsion angles of each protein side chain using an MSE loss with respect to the ground truth torsion angle. We predict a distogram loss that bins the orientations and distances of pairs of residues as a projection from the 2D track. We introduce a new bond loss which is an L1 loss penalty on bond lengths, angles, and distances in small molecules that are in rigid groups like aromatic rings. We additionally add a small clash loss that penalizes overlapping van der waals radii. All of these losses are described in detail in [71].

4.4 Ligand Docking

Ligand docking is the general problem of determining the plausible interacting structures that a protein and a small molecule ligand may jointly adopt, and ranking those structures according to how stable they are biophysically [121]. There are many approaches and simplifications of the general problem: in this work we will consider the problem of predicting the crystal structure of a protein-ligand complex, which is often the most stable structure that complex can form. A common-metric to evaluate ligand docking is kabsch-aligned root mean squared distance (RMSD), which is calculated by aligning the predicted and crystal protein structures and then measuring the root mean squared distance of the predicted and crystal ligand structures. To simplify the metric, we will designate the model's prediction a success if the RMSD is < 2Å, a common threshold under which predictions may be useful for biological discovery. We will care particularly about the fraction of protein-ligand complexes that the model predicts successfully.

4.4.1 Benchmarks

We evaluate our model on three different publically available benchmarks. The purpose of these benchmarks is three-fold: first, to understand how well our model performs at protein-ligand co-folding. Second, to compare our model to existing deep learning models that have been previously published. Third, to understand how well calibrated our model is: that is, to determine if the model can predict when it has made a good or bad prediction. The model is trained to predict its own error via a predicted alignment error (PAE) auxilarry network that predicts the aggregate value of the alignment (FAPE) loss between every pair of residues. Theoretically, a well-calibrated model should know when it has made a bad prediction by outputting correspondingly high PAE values between the protein and the ligand being predicted.

CASP15

The Critical Assessment of protein Structure Prediction (CASP) is a biannual blind structure prediction challenge in which participants submit predicted structures for proteins whose structures have been determined experimentally but have yet been released to the public. The latest edition of CASP, CASP15, included a protein-ligand complex structure prediction portion which we retro-actively evaluated our model on.

We downloaded the sequence data from the CASP15 targets marked as ligand targets. For simplicity, we omit all heteromeric targets and RNA targets (H1114, H1135, H1171v1, H1171v2, H1172v1, H1172v2, H1172v3, H1172v4, and R1117v2). We also omit target T1181 because the prediction of the full trimer did not fit in GPU memory.

For every other target, we predict the full protein ligand complex providing the correct stoichiometry for both ligands and protein chains. We model target T1152 as recommended by the CASP15 organizers (as a monomer). For target T1170, we omit the DNA sequence from the prediction as it was not provided at input time to the participants of the challenge, and its structure was not necessary for proper docking of the ligands in that target.

For each target, we make 10 predictions with different random seeds and then select the

prediction with the lowest inter-chain protein-ligand predicted alignment error (PAE Interaction), including early stopping over 20 recycles. For multimers, we found that first scoring on plddt and then choosing the recycle with the minimum PAE Interaction selected more plausible structures since there were some cases where the network made poor protein predictions that were distracting from the correct ligand placement. We score the predicted pose of each ligand using the ligand RMSD metric mentioned above, and recompute this metric for each of the servers that submitted to CASP15 during the competition for the same ligand poses in each target. We score only the top-ranked pose and model (e.g. model 1, pose 1) from each server for a fair comparison between methods.

Panel A of Figure 4.1 shows the results of our model against three of the top competitors from CASP15 at ligand docking. We note that we are approximately on par with the existing methods - in fact, the RFAA model would have placed second in the CASP15 challenge had it been entered. In addition to this, competing methods made heavy use of manual intervention and multi-stage pipelines that may be untenable in real-world applications. By contrast, RFAA is a fully automated tool to produce structure from sequence and chemical graph inputs.

CAMEO

The CAMEO BETA challenge (`https://beta.cameo3d.org/`) is an online, weekly, continuous evaluation of the most recent depositions into the PDB. We registered the RFAA model as a server for homomeric and ligand targets (excluding RNA, DNA and heteromeric targets for simplicity). The RFAA server searches for MSAs and templates for each protein sequence as described above. The server makes 5 different predictions with different random seeds for each target and then submits the prediction with the lowest interchain PAE. If there are multiple ligands in a given target, the interchain PAE is calculated over all ligands in the target.

We ran the RFAA CAMEO server (Server 2) from 04/08/2023 to 09/02/2023. Ligand pose baselines were introduced from 08/12/2023 onward (`https://beta.cameo3d.org/cameong_servers/`). The CAMEO organizers returns ligand RMSD poses scores that we then plot for our evaluations. We do not compute our own RMSD metrics for the CAMEO challenge.

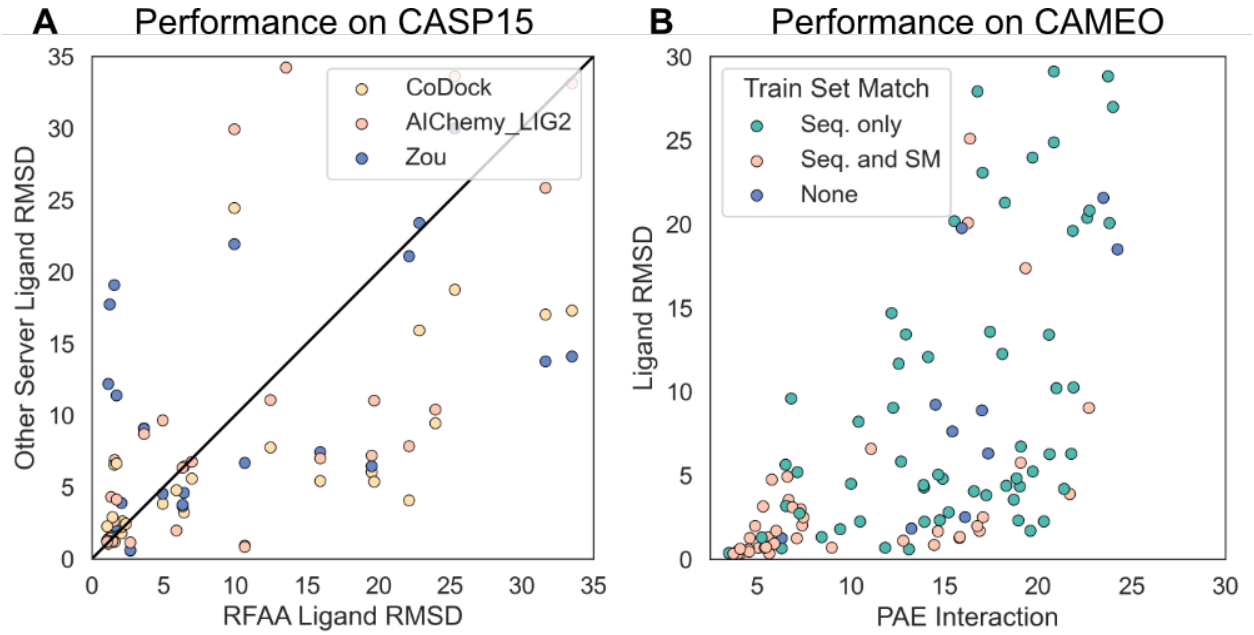


Figure 4.1: A) Ligand docking performance relative to the top three methods that competed in CASP15. B) Ligand docking performance relative to the model's error estimates on the online, blind CAMEO challenge. The colors indicate whether or not the example is close (>%30 percent sequence identity) in sequence space to anything seen in training, as well as whether or not the ligand binding partner is similar to ligands (> 0.7 tanimoto similarity) seen in training to bind to similar sequences.

The CAMEO server does not provide stoichiometry information for either protein or ligand in a given protein ligand complex. In order to determine protein stoichiometry, we check the symmetry group of each template of the input protein. We make a list of the top 20 most similar templates to the input protein by sequence similarity, and then filter that list based on a coverage threshold of 0.75 and a match score of 0.95. For the passing templates, if at least half of them form a dimeric or a trimeric complex, we then attempt to predict a dimeric or trimeric complex with the input protein, respectively. If the predicted alignment error between the symmetric subunits is less than 10 and the computed clash score between the subunits is less than 1, we model the protein as either a dimer or a trimer. Otherwise, we default to modeling the protein as monomeric.

We only make predictions with a single copy of each ligand in each target and do not duplicate

ligands in the case of dimer or trimer-forming protein chains. We found it difficult to distinguish between cases where there are two binding pockets for two different copies of the ligand in a dimer compared to a single copy of the ligand sitting at the dimeric interface. We also do not model symmetry groups above C3 because they often do not fit in GPU memory on the allocated GPUs.

Panel B of Figure 4.1 demonstrates the model's performance (ligand RMSD) relative to its protein-ligand PAE, which is the mean of the PAE matrix between the protein and ligand and indicative of how well the model thinks it has place the ligand relative to the protein. The model shows strong correlation between error estimates and true performance. This phenomenon, known as calibration, is important in biological discovery as it allows users to determine how reliable a given prediction is on a complex with unknown structure.

Each target from CAMEO was compared to the training set of the model using BLASTp [5] and considered *similar* to something seen in training if BLASTp found at least one sequence with >%30 sequence identity over at least %50 bidirectional coverage. The ligand partner of the target was compared to every ligand partner seen in training to bind to those similar protein sequences via the tanimoto coefficient [97]. Ligands were considered similar if the tanimoto coefficient was greater than 0.7. The scatter plot indicates that although the model is strongest at those things seen in training, it is able to predict some novel complexes well. More importantly, the model also remains well-calibrated on unseen examples.

PoseBusters

PoseBusters is an offline test set curated from the PDB that was designed to test whether or not deep learning methods trained for protein-ligand docking generate chemically-plausible structures, e.g. that satisfy planarity constraints or roughly ideal bond length distributions [18]. Each item in the PoseBusters dataset consists of a PDB entry and a ligand identifier specifying the ligand to be docked. At the time of our evaluations, there were 428 protein-ligand complexes with non-redundant protein chains and non-redundant, drug-like ligands. Note that future versions of PoseBusters updated this number to 308 entries by filtering out entries that have ligands at symmetric protein interfaces.

We preprocess the dataset by parsing the raw cif files of each PDB entry indicated in the

PoseBusters list. For the given query ligand identifier, if it appears multiple times in that cif file, we sort the ligands by chain ID and pick the first such ligand. We find the protein chain in that entry that has the most number of atoms within 10Å of any atom in the designated *query ligand* and call that protein chain the *primary* protein partner. We also search for any cofactor in the assembly that has at least one atom within 5Å within any atom of the query ligand. We record these contacting cofactors and the primary protein partner in a list of partners to be predicted.

For each item in the PoseBusters evaluation set, we predict the entire complex - query ligand, primary protein partner, and contacting cofactors - simultaneously. We superimpose the primary protein partner onto the crystal structure and extract the crystal pose of the query ligand. We then run the posebusters suite on the predicted pose of the molecule to determine chemically validity of the predicted ligand pose. We also use the posebusters suite to compute ligand RMSD between crystal and predicted ligand pose after protein superimposition, and to compute validity metrics between predicted ligand pose and predicted protein structure (and predicted cofactors).

We note that our model is not a docking method and does not take as input the crystal structure of the protein nor the binding site of the query ligand, making our task (prediction from sequence information alone) strictly harder than blind docking or local pose estimation. This implies that it does not make sense to compute validity metrics, such as minumum distance clashes, between predicted ligand pose and crystal protein structure, which is why we use the model's predicted protein structure instead.

Finally, we also note that although our model is only trained on data from the PDB up to April 2020 and the PoseBusters dataset is only defined on data from 2021 onwards, our training data is defined on the deposition date of the PDB entries, while the PoseBusters dataset is defined on the entry release date. In almost all cases the dates of deposition and release are similar enough that the items in the PoseBusters evaluation set are not in our training set. However, a single entry (6VTA, AKN) was deposited before April 2020 but released in 2021. We exclude this item from the evaluation set for all methods.

The results of the PoseBusters evaluation are in Figure 4.2. Panel A demonstrates that RosettaFold All Atom has superior success rates in docking under 2Å than other publically-available

deep learning methods. Panel B demonstrates the metrics about validity of the chemical structure producted by the method. We note that the minimum distance to protein filter is a highly stringent filter that considers a structure chemically invalid if an atom in the protein and an atom in the ligand are within 0.75 times the sum of the pairs of van der Waals radii of those atoms. Anecdotally, we see many examples of good predictions with a slightly shortened hydrogen bond or non-polar interaction which end up failing this particular test.

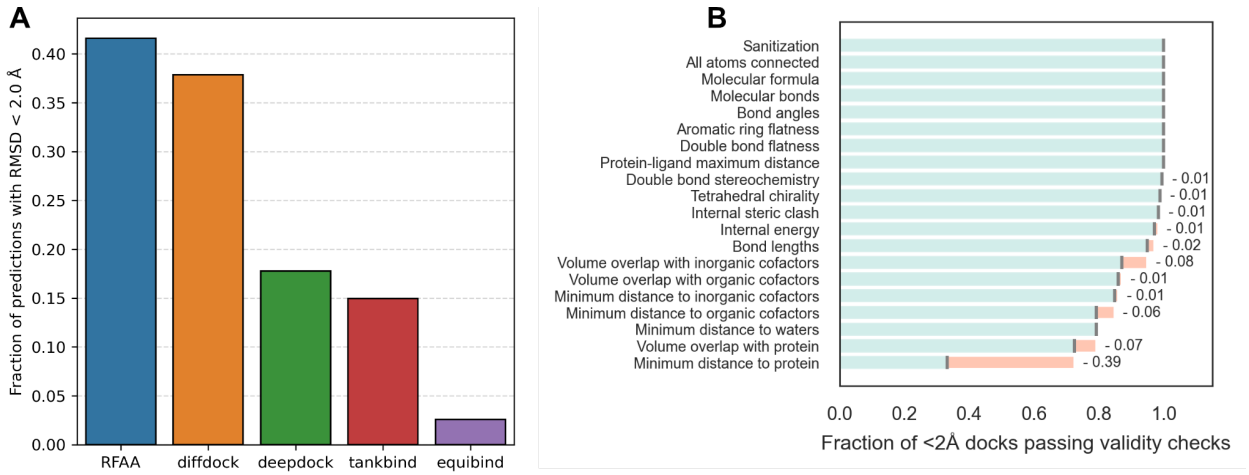


Figure 4.2: A) Docking performance on Posebusters relative to other publically-available deep learning methods. RFAA outperforms all other such methods at docking on this test set. B) Validity checks of RFAA docks as determined by Posebusters. The RFAA docks generally pass most of the tests, except for the minimum distance violation between the protein and the ligand.

4.4.2 Emergent Properties of a Co-Folding Model

Although docking is in of itself an important application, a structure prediction network that can reason over small molecule atoms and bond graphs opens up a host of broader applications in both biological discovery and de-novo design. These applications are partially due to the fact that proteins may behave differently around small molecule binding partners than they do in isolation, and the RoseTTAFold network can, theoretically, model a protein both with a binding partner and

without it. We mention two of the most important such applications in the following subsections.

Chemically-Induced Dimerization

Chemically-Induced Dimers (CIDs) are pairs of proteins, either homo or heteromeric, that only form a complex in the context of a small molecule. These systems have evolved in biology in the context of gene regulation and degredation, and also have a wide variety of potential engineering and therapeutic applications, e.g. as chemical switches of desired therapeutic events [122]. The RFAA model is not explicitly trained to model chemically-induced dimerization, particularly because the number of known CID crystal structures is very small. However, it is trained both to model protein-ligand complexes and to model protein-protein complexes. It is a reasonable hypothesis to expect that it has learned something about ligands at protein-protein interfaces and if a ligand improves the interface between two proteins.

To test this hypothesis, we modeled three different systems known to be CIDs:

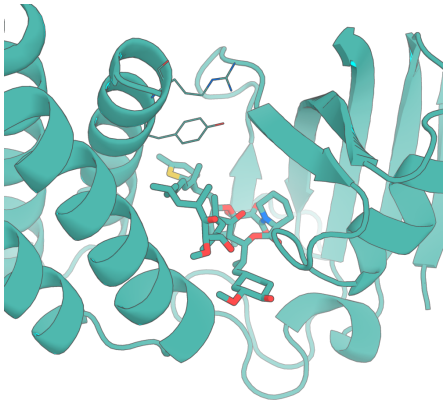
- FKBP1A and mTOR in complex with rapamycin (3fap)
- Cereblon and Casein kinase I isoform alpha in complex with S-Lenalidomide (5fqd)
- A denovo-designed cannibidol sensing system (7te8)

We predicted each complex with and without the ligand and measured the protein-protein PAE in both predictions. The results are in Table 4.1. We note first off that both 3fap and 5fqd were part of the training set - an alternative and reasonable hypothesis would be that the RFAA model would learn to memorize the protein-protein complex position regardless of whether or not the binding ligand was also present. However, this appears from the table not to be the case - in all three examples the model has signficantly lower predicted error with the bound ligand present in the input. We note that this is not just a product of have a ligand present in the complex - for example, repredicting the structure of 3fap with heme as opposed to rapamycin still causes the inter-protein PAE to jump, up to 8.7. Although more work needs to be done both to validate the model's ability to

Complex PDB ID	PAE with ligand	PAE without ligand
3fap	2.8	11.7
5fqd	11.0	19.2
7te8	13.6	18.3

Table 4.1: Prediction of CID complexes with and without their ligand binding partner.

screen entirely unseen CID systems, and potentially fine-tune the model to design them, the RFAA model as is already shows promising ability to understand binding with and without ligand context. The predictions of 3fap with and without the ligand are shown in Figure 4.3. The model seems to show some space-filling ability - without the ligand, it attempts to fill the empty space between the proteins with TYR194 and ARG194 in mTOR.



(a) Predicted structure of FKBP1A and mTOR in complex with rapamycin

(b) Predicted structure of just FKBP1A and mTOR without rapamycin

Figure 4.3: Predictions of 3fap

Co-Folding Over Folding and then Docking

Many previous methods approached ligand-docking as a two stage problem: first predict the structure of the protein, and then keep the structure (mostly) fixed while determining where the ligand docks to the protein. The drawback of this approach is that the two problems are actually inter-related - how the ligand docks to the protein may affect the side-chain positions or even the backbone structure of the protein. We hypothesize that a model that can fold the protein in the context of the ligand may better predict the structure of the portions of the protein that form ligand-binding pockets, because the model is better aware of the type of interactions that may form the structure of those pockets.

We develop a set of 594 of proteins with ligand binding partners that are predicted confidently (protein pLDDT > 80) by both the original RoseTTAFold 2 model and RFAA without the ligand binding partner. For each item, we make 3 predictions from both RFAA and RF2 to remove any artifacts that come from random seed, and pick the prediction that either minimizes inter-chain PAE or maximizes pLDDT, respectively.

We measure the all-atom RMSD of the predicted protein structure relative to the crystal after kabsch alignment on the backbone atoms. We also measure the *ligand pocket* RMSD, where the ligand pocket is defined as the set of residues that have at least one atom within 5Å of the ligand in the crystal structure. The ligand pocket RMSD is then computed as the all-atom RMSD of said residues after kabsch alignment on the backbone atoms of the same residues.

We observe a statistically significant difference (paired t-test) between the RMSD values from RFAA and RF2, depicted in Figure 4.4. We depict several examples of structures that RFAA predicts better than RF2, likely due to the added ligand context (Fig. S4B-D).

We also note that we observe anecdotal evidence that the model has some capacity to understand conformational shifts upon binding. Some proteins exhibit one structure in isolation and a slightly (or vastly) different structure in the presence of a ligand binding partner, and a model like RFAA that is capable of predicting structures both in the presence of and in the absence of a known binding partner may be able to reflect such conformational shifts. Although we have not rigorously

benchmarked performance on any type of conformational prediction task, it seems like a promising avenue for future investigation.

4.5 Applications in Virtual Screening

Virtual screening is the problem of determining whether or not a protein and a small molecule (usually a drug) will bind, and if so with what affinity [140]. Often during drug discovery, the goal is to find a single or a small number of candidate molecules out of a pool of millions to bind to a single protein target. The target's general structure may or may not be known in advance, as well as the intended binding pocket the small molecule is supposed to bind to. Given that screening is done on a pool of chemicals, it suffices to provide relative ranking scores rather than absolute binding affinity estimates, although models exists that do both. Given the high-throughput nature of the problem, it is important that any tool be able to scale well with respect to the number of screened compounds.

4.5.1 Discrimination using the PAE Head

The most natural question given the architecture of the RFAA network is to ask if it is already capable of virtual screening through use of the predicted error. It is a reasonable hypothesis to suppose that the model's PAE on protein-ligand pairs that do actually show binding affinity will be lower than protein-ligand pairs that do not actually bind. This is mainly because protein-ligand pairs that show no affinity will almost surely not have any spatial arrangement that resembles protein-ligand complexes that the model has been trained on, and therefore the model will be unable to place the so called *decoy ligands* in any reasonable conformation relative to the protein.

In order to test this, we make use of the published PoseBusters test set [18]. This test set was originally developed to benchmark deep learning-based docking methods and contains 428 unique protein-ligand complexes. For each target, we perform a "cross-docking" procedure in which we dock random *decoy* ligands to each target that are unlikely to bind to that target and measure if the PAE is higher or lower for the true ligands or the decoy ligands.

In more detail, for each of the 428 protein targets in that set, we find the ligand in the remaining 427 ligands that has the highest tanimoto similarity to the original "true" ligand for that target, as well as 5 randomly selected ligands from the remaining 426 ligands, and model the protein target with each of the 6 chosen decoy ligands using RFAA and 10 recycle iterations. We measure the model's inter-chain protein-ligand PAE for each protein-ligand complex and plot the result in the aforementioned figure.

The model's predicted error tends to go up as the Tanimoto similarity between true and decoy ligands decreases, which implies that the RFAA model has some ability to discriminate between the true binding ligand and ligands that do not bind or fit in the same pocket. In particular, 75% of decoy docks have higher predicted error than the original predicted protein-ligand complex. This is demonstrated in Figure 4.5, which plots the PAE as a function of the similarity between the decoy and known binding ligands.

As a caveat to this experiment, it is important to note that random cross-docked decoys may actually bind to given protein targets - the PAE difference is only indicative of a general trend.

4.5.2 Fine-tuning for Virtual Screening

Although using the PAE as a discriminator shows some signal, it is not specifically trained for the task of virtual screening. This begs the question - is it possible to improve performance by specifically training for this task? For the purposes of this problem we take a contrastive learning perspective, in which the learning task is to rank ligands within a batch relative to each other rather than predict absolute affinities.

In brief, contrastive learning is a framework in machine learning in which the task is to have a single network discriminate between similar and dissimilar elements in pairs. In particular, any two samples may be assigned a positive or negative relationship and the goal of the framework is to produce a score (either via metric learning, standard classification, or a host of other losses) that predicts the pairwise label. The motivation behind contrastive learning is that it is easier to determine at data curation time whether or not two inputs belong to a similar or disssimilar group than it is to absolutely classify those inputs. The field of contrastive learning has a long history and

has been applied extensively in both vision and biological applications; for a review see [74].

The specific motivation for adopting this framework when it comes to virtual screening are several-fold. Measurements of protein-ligand binding affinities are heterogeneous and often difficult to compare across conditions and targets, making absolute prediction an inherently noisy task [65]. Predicting absolute affinities is a more difficult learning task than what is necessary to succeed in virtual screening - the model only needs relative ranking scores to determine the starting points for lead optimization. Pairwise or batch-wise ranking is by contrast an easier learning task, more stable to train, and more easy to generate artificial data for. It is intractable to generate reliable computational data with absolute binding affinity measurements, but relatively easy to model whether the addition or deletion of a group on a small molecule would make it a relatively worse binder, e.g. because it induces a steric clash or deletes a hydrogen donor or acceptor.

We also take a structural perspective on virtual screening - while some methods are sequence only, docking and binding are inherently structural phenomena. It is not clear that any model that is not explicitly modeling the structure of the ligand and the associated binding pocket will learn principles that generalize to novel targets or compounds that are unlike those seen in training. In contrast to sequence-only models, the RFAA is explicitly structural.

We define a training set that makes use of the RosettaFold All-Atom training set for known protein-ligand complexes with crystal structures, which was scraped from the Protein Data Bank [49]. We generate negative examples in-silico using the following procedure:

- Pool together each ligand from the existing positive training set.
- For each protein-ligand complex, search the query ligand against the pool of existing ligands from the training set.
- Record each ligand that has tanimoto similarity less than 0.7 to the query ligand.

At training time, for each positive protein-ligand pair, we then sample uniformly at random one of the searched decoys and provide the protein + true binder + negative decoy as input to the model.

An extra attention pooling layer and two feed-forward layers are then attached to end of the 2D track of the network: these layers downsample the inter-protein-ligand pair features per-ligand to produce a single scalar-valued score that is trained via cross entropy to classify both the true binder and the negative decoy independently.

In addition, we modify the architecture slightly so that the model never computes inter-attention values between ligands. We reason that the standard base model is trained to form complexes and so the inter-ligand attention weights are used to position ligands relative to each other - however for the virtual screening task we desire the model to reason separately about each ligand and enforcing this extra inductive bias may help the model generalize beyond ligands seen in training. We note that although we have not performed extensive experiments with computational time, not computing inter-ligand attentions allows for linear rather than quadratic scaling in the number of ligands, which may enable more high-throughput screening.

We develop a held-out set for evaluation by clustering the PDB using MMseqs [52]. We select two representative examples from 64 held-out clusters such that no examples from the held-out clusters have bound ligands that were seen in training, and fix decoys deterministically for each of the 128 examples such that each decoy is no more than 0.7 tanimoto score close to the original bound ligand. We run the un-finetuned model on each protein-ligand complex (decoy and true binder) separately and record the PAE between the protein and the ligand as an estimate of binding score, and then run the fine-tuned model on the joint protein-decoy-true complex and record the per-ligand scores for each. We measure the per-target accuracy of both models, defined as the fraction of protein targets for each either model ranks the true binding ligand higher than the decoy ligand. We find that fine-tuning even for only 20k iterations improves the per-target accuracy from 0.71 to 0.79.

We then investigate the question of whether the model can generalize to more than two targets simultaneously. Being able to simultaneously rank a large batch of ligands is a key step in enabling high-throughput screening, and there is reason to belive that the permutation-invariant transformer architecture can scale well beyond the length of items seen during training. We use the same set of known protein-ligand complexes in the previous experiment, and develop a set of 16 decoys per

protein ligand complex in the same manner. For each target, we feed in the protein target, the true bound ligand, and the 16 decoys simultaneously during the forward pass, and the model produces ranking scores per-ligand.

We plot the results in Figure 4.6. The left plot shows an enrichment curve: the value at k is the fraction of protein targets (of which there are 128 total) such that the true bound ligand is ranked among the highest k candidates predicted to bind to that protein target. A perfect model would have an area under the curve of 1 and a model that ranked them randomly would have an area under the curve of 0.5 (e.g., it would track the diagonal line). We see moderate enrichment in the curve, which indicates that the model has learned to generalize to large batches without ever having seen them in training. The right plot is a histogram of the rank (out of 17) that the true ligand appears in. The plot demonstrates that the model identifies the true binder perfectly in over 25 of the protein targets, and generally enriches for the true ligand in the top half of the batch.

4.5.3 Fine-tuning for Binder Design Screening

The model trained in the section above shows potential to be useful for drug discovery - however when doing de novo protein design the emphasis shifts to screening many protein targets against a single ligand rather than a pool of ligands against a single protein target. In this problem framework, it is important that a model be able to distinguish between similar proteins as to which one best binds a given ligand. Since proteins are much larger than small molecules, we abandon the framework of contrastive learning and instead fine-tune a model that takes in a single protein-ligand complex and predicts a single, absolute score for that complex. The architecture of the prediction head is similar to that described above, except that it pools the entire pairwise state feature matrix and produces only a single score.

In addition to the decoy generation procedure above, the training procedure also involves generating artificial decoys by mutating key residues in the binding site of the protein, with the dual intuition that the right mutations will destabilize binding interactions and that a model designed for screening de-novo designed proteins should have an understanding of which residues cause destabilizing mutations. The binding site is identified from the known crystal structure as any

residue that has at least one atom within 5Å of any atom on the bound ligand. A number between 1 and 3 is sampled uniformly at random, and then that number of residues are sampled uniformly from the number of binding residues to be mutated. The residues are mutated with probability inversely proportional to their BLOSUM62 score, which is a measure of mutation rates observable across evolutionary history [35]. That is, residues are mutated in such a way that would be unlikely to occur in actual evolution, implying that such mutations would destabilize key binding interactions.

In addition to the above training procedure, we also train the model by templating the crystal structure of the protein, reasoning that at inference time a reasonable approximation of the protein structure can be produced via a separate protein folding model like AlphaFold 2. This allows this version of the fine-tuned model to focus more on protein-ligand interactions and less on protein structure determination, which is a phenomenon mostly learned from multiple sequence alignments which are not available for de novo designed sequences.

As an evaluation set, we gather retrospective data on two design campains screened in the wet lab. The first campaign is was a campaign to design protein binders to the drug Apixaban, in which each protein was de novo designed but based on the structural backbone of the NTF2 family [19, 95]. The campaign consisted of 4,229 designed proteins, of which 8 demonstrated binding affinity at the nanomolar concentration level. The second campaign was a campaign to design enzymes to catalyze the retro-aldol reaction [91]. 540 enzymes were designed and screened, and normalized activity levels were computed for each enzyme. To binarize the problem, we denote the top 5 percent of enzymes as highly active and the remainder as not. Unlike in the Apixiban campaign, which were designed small molecule binding proteins, this campaign features enzymes which are not intended to bind to their substrates but rather catalyze a reaction with them repeatedly. We reason, however, that the screening model should still show some signal because enzymes still must make some interactions with their substrates during catalysis to stabilize the intermediate transition states.

We predict the structure of each designed protein using AlphaFold 2 in single sequence mode, and then feed the predicted structure as a template in addition to the sequence and the chemical make up of the small molecule to the fine-tuned model, using 10 recycles and again in single sequence mode. The model produces a single binary score and we label a design as positive if it is

in the 8 designs that showed nanomolar binding affinity or was in the top 5 percent of normalized activity, respectively. We then compute the receiver operating characteristic curve, a measure of binary classification accuracy that indicates rank-ordering performance of positive vs. negative examples. As a control, we also use the un-finetuned base model and take the protein-ligand PAE as a prediction of binding affinity or enzymatic activity, respectively.

The results are shown in Figure 4.7. In both cases, the fine-tuned model significantly increased discrimation performance relative to the base model. Both models show higher performance at small molecule discrimination rather than determination of high activity, which is unsurprising given that enzymatic activity is governed by more factors than just binding strength. In particular, the curves indicate that the enrichment is high for both campaigns - the performance of the model at extremely low false positive rates. This is important because wet lab screening is an expensive process and the goal of a screening model is to filter down a large pool of designs into a small enough number to screen in vitro.

4.6 Improving Docking Performance

Shortly after our paper was published, the AlphaFold 3 paper was published [1]. AlphaFold 3's result were impressive, particular on ligand docking. Their performance relative to our method is in Figure 4.8. Despite this, the truth is that much of the innovation in that paper were things that we had already anticipated or were actively working towards. The two main things that came up in the context of my work post-publication were better dataset curation and a move toward more stochasticity.

The latter came up as a result of chance: I noticed that many of the ligand docks that our method got wrong (e.g. RMSD > 2Å) were in the right pocket, and generally in the right place. The problem was that they were either flipped along a symmetric axis or along an axis of symmetry that preserved space-fitting constraints, but was slightly less favorable a fit than the crystal dock. This implied that the model often made predictions that were "nearly" right but not exactly, and the difference between nearly right and successful was energetically small and difficult to distinguish geometrically. In one particularly salient example, shown in Figure 4.9, I observed that randomly masking out atoms in

the input ligand structure allowed the model to flip a ligand dock from incorrect to correct along a symmetric axis. More importantly, the correctness of the dock was understood by the model in its PAE prediction: when the model predicts the correct dock it also has a much lower predicted PAE. This highlights both that the model is well-calibrated and also that the correct dock is learned in the model: it just happens that for some stochastic input patterns, the dock with the wrong flip is more favorably predicted because the binding pocket allows for that symmetry spatially.

Inspired by this, we applied the following random masking procedure on PoseBusters: for each of five models for a given input, randomly mask between one and two atoms in the given query ligand and remake the prediction. Then, pick the prediction with the lowest inter protein-ligand PAE and pick that as the best model. Comparing this against the standard procedure (single model, no masking) lead to a performance increase of 3% (from 42% success rate to 45%), which implies there is a non-trivial number of examples for which the model has learned the correct dock but doesn't always produce it correctly due to the dock potentially having multiple spatial fits or low energy minima. This directly implied the future use of generative modeling in ligand docking: if it is the case that such stochasticity helps in structure determination, the process should be generative from the start. This is exactly the observation made in AlphaFold 3, which turned structure prediction into a generative process.

The second part of the picture is better dataset preprocessing. The original version of RosettaFold All-Atom used a very hacky and bespoke preprocessing of the PDB, which split protein small molecule complexes into several different subcategories, left out a large portion of those complexes for validation or filtering reasons, and sampled them sub-optimally only by protein sequence cluster. Many of these design decisions were done for convenience rather than optimality, and we had long known that one of the best ways to improve the model was to improve the underlying dataset code. I spend many weeks re-parsing the PDB and improving our datasets to include missing clusters, sample protein ligand complexes by joint clustering, and remove the filtering steps so that more of the data was represented. Preliminary results are in Figure 4.10. Sadly, I didn't have time to train a full run to completion with the new data, but it seems pretty clear from initial trends that improvements to data translate directly to improvements in modeling performance.

All of these improvements, plus a variety of improvements in speed and memory usage, including in the way we parse and deal with MSAs and templates, made it in to the newest version of the RFAA pipeline in the lab, which is still under active development.

4.7 Discussion

A lot of the work I did for this particular project was the less glamorous side of research: work in data loading and data processing, as well as code and performance optimizations for a large and slow model. Nonetheless, the evaluations I did became a central portion of the protein small molecule complex prediction results that made it into the paper: results implying that not only had RosettaFold All Atom learned at least something meaningful about docking, but also that in doing so it had learned something about movement in binding pockets, screening different ligands against the same pocket, and predicting ligands at protein-protein interfaces. All of this implies a bright future for RFAA: as it gets better at generalized structure prediction, so too does its performance improve at a variety of relevant downstream tasks that unlock new applications in de novo design.

The importance of this work lies in carefully understanding the underlying data: how to deal with large, heterogeneous structural databases like the PDB, how to cluster and sample diverse molecule complexes, and how to represent such complexes to a machine learning model. In the time since AlphaFold 3 and RosettaFold All Atom have been published, we have made great strides towards improving the model and I am optimistic that we will continue to do so in the future, supported by the insights gained from this chapter.

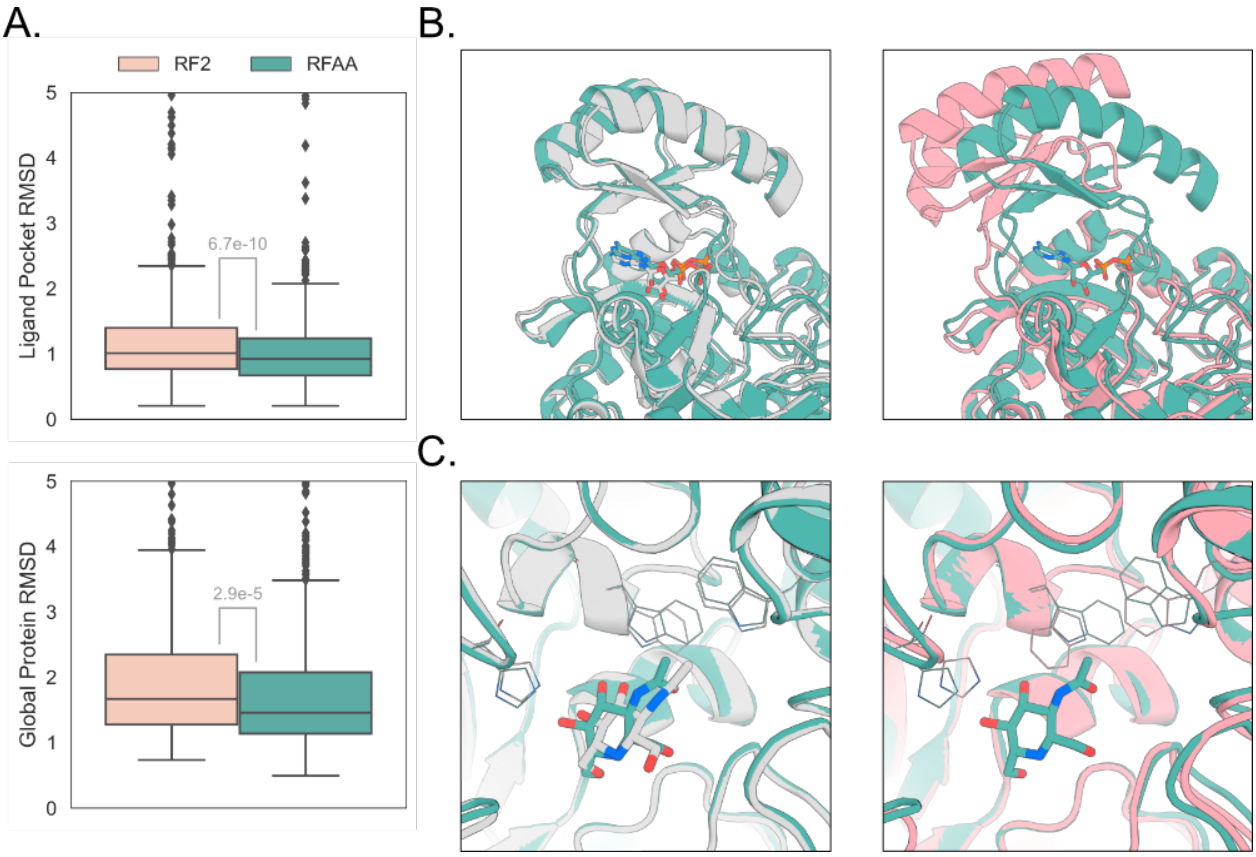


Figure 4.4: A) Ligand context allows a small but statistically significant performance boost at protein prediction, both globally and in ligand binding pocket regions. B) Ligand-aware protein folding allows better positioning of relative protein domains. The crystal structure is gray, the RFAA prediction is green, and the RF2 prediction is pink (PDB ID: 7kct). C) Ligand-aware protein folding enables more accurate side-chain predictions in a binding pocket (PDB ID: 7kg7).

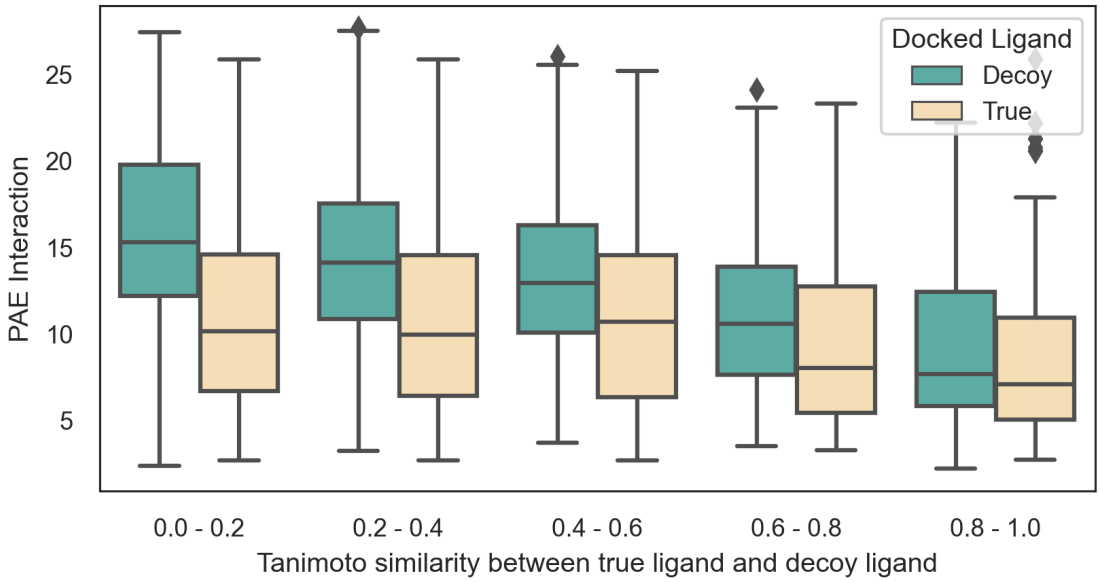


Figure 4.5: Cross-docking decoys on PoseBusters. Decoy molecules generaly have higher PAE than known binders, and the difference is exacerbated the farther away the decoys are from the known binders in chemical space.

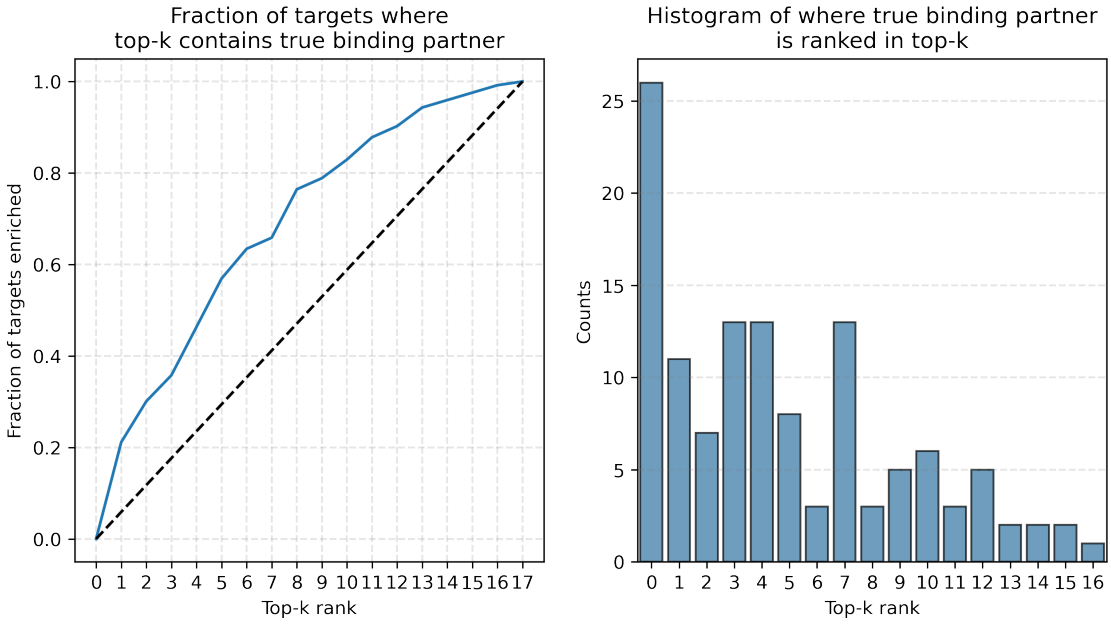


Figure 4.6: The generalization of a pairwise contrastive model to 16 simultaneous ligands. Even though the model has only been trained to see two ligands at once, it can still enrich batches of 16 decoy ligands with a single known binder. This opens the door for high-throughput virtual screening in-silico with large structure-based networks.

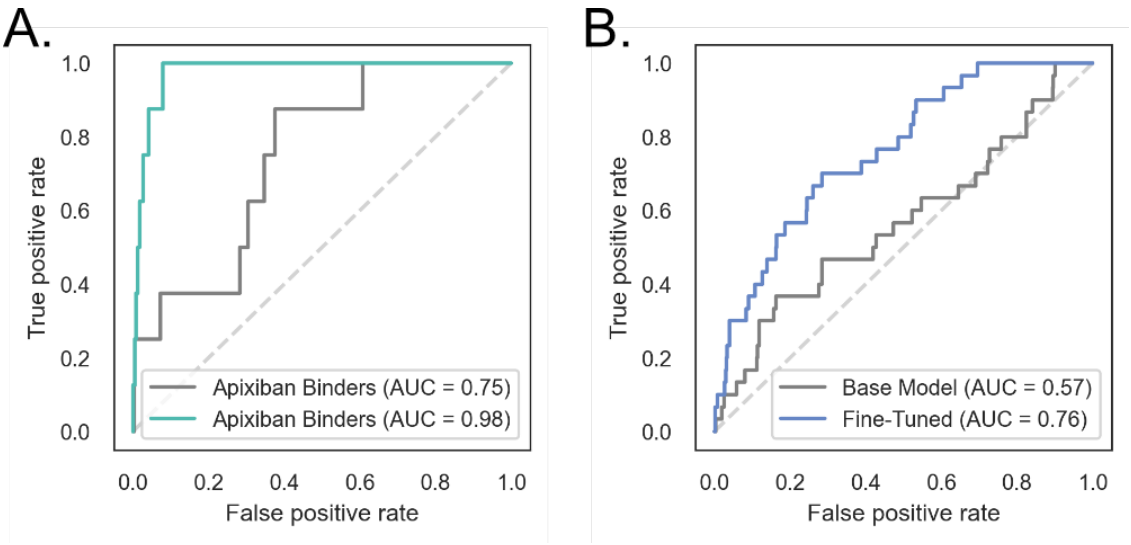


Figure 4.7: The retro-active performance of a model specifically fine-tuned for discrimination of proteins that bind to ligands on experimental data gathered by collaborators in the wet-lab on two de-novo designed campaigns. A) The model can highly enrich for de novo designed binders using the NTF2 scaffold to the drug inhibitor Apixaban. B) By modeling the subsrates of a reaction binding to the protein, the model can also screen for highly active enzymes on a de novo design campaign of retro-aldolases.

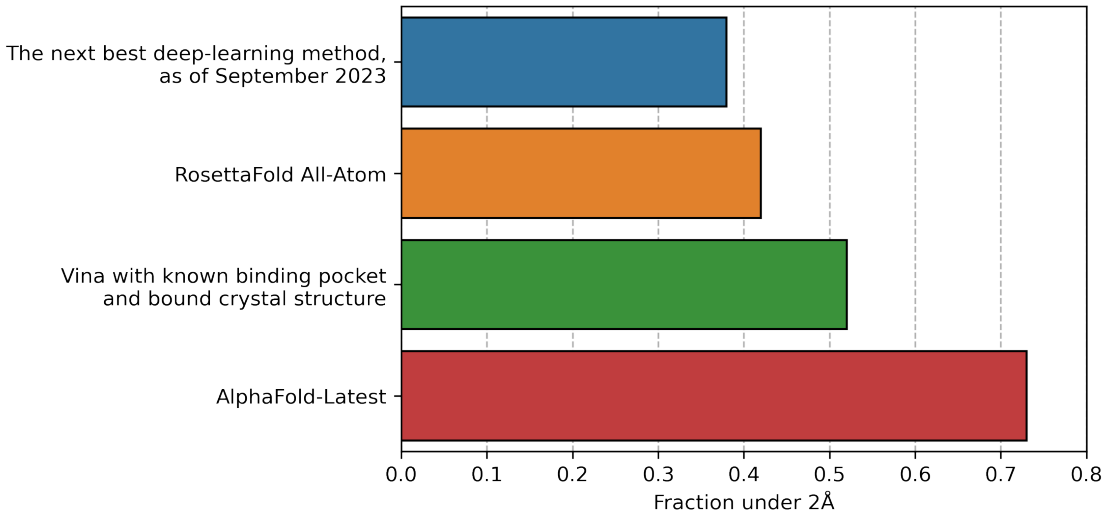


Figure 4.8: The performance of RFAA relative to AlphaFold 3 at ligand docking.

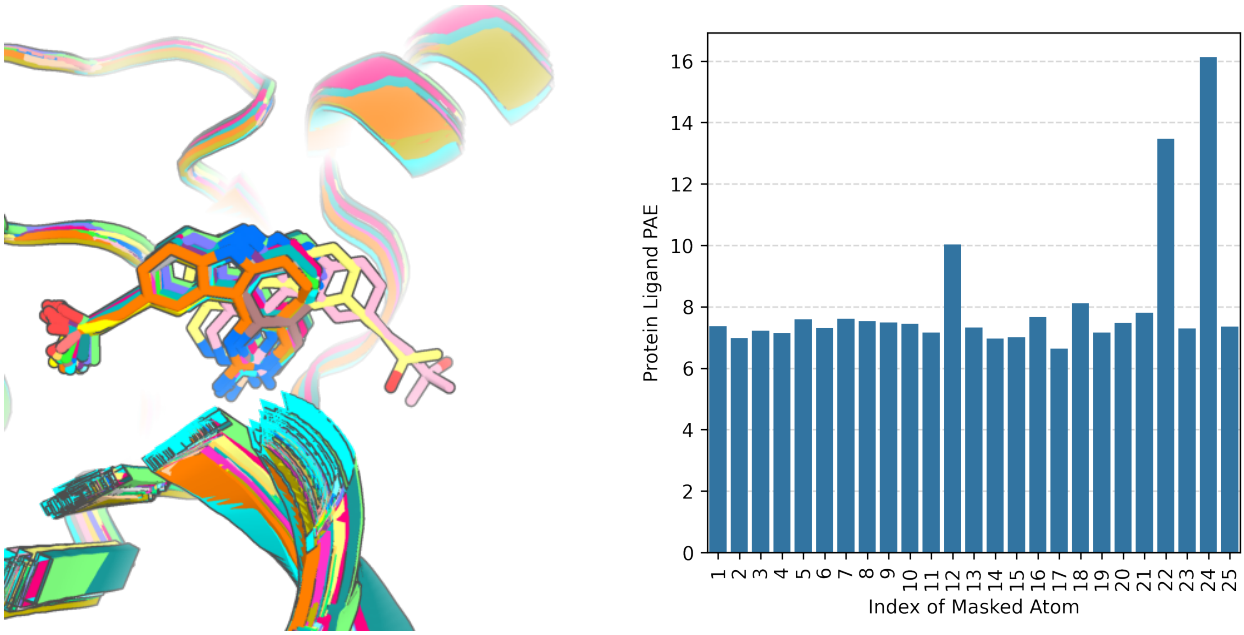


Figure 4.9: Masking random ligand atoms can lead to improved docks, highlighting the role of stochasticity in helping resolve partially symmetric or ambiguous docks. Left: a predicted dock (7jxx) with all possible single-atom masks applied. Right: the inter protein-ligand PAE of the prediction with the given atom masked.

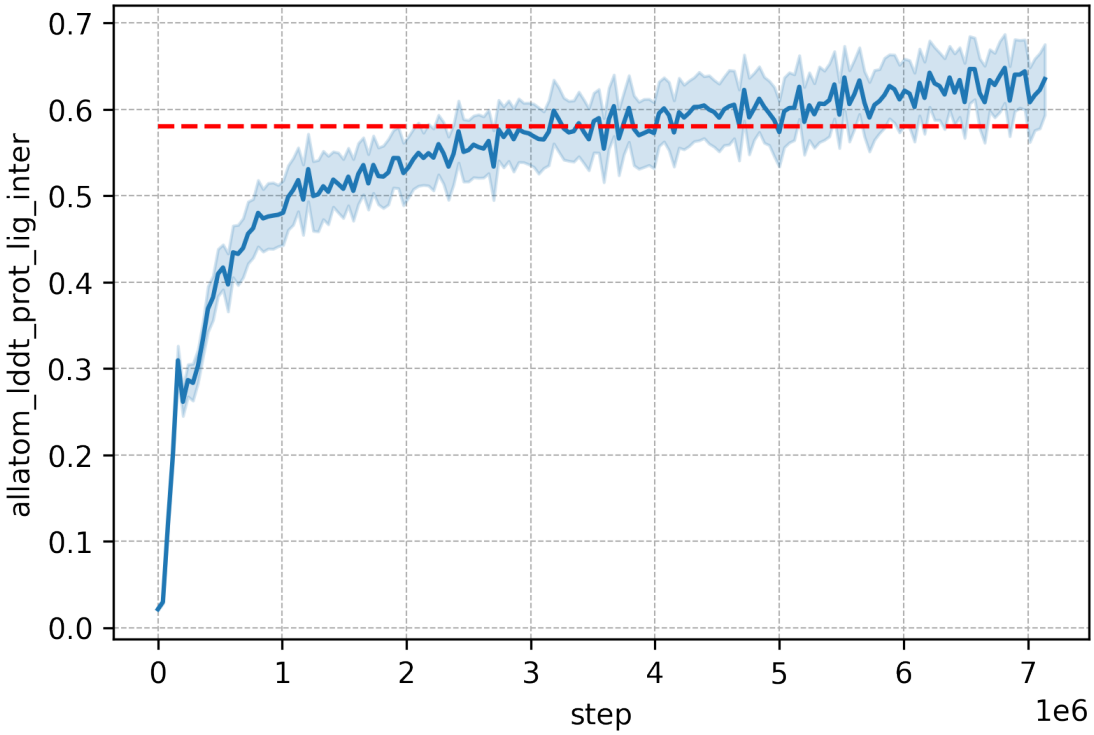


Figure 4.10: New protein small molecule datasets and improved sampling for RFAA leads to improved docking performance. The dashed red line is the validation performance of the model released in the paper on the same validation set.

Chapter 5

CONCLUSIONS

I draw three main conclusions from the work done during my PhD. The first is that any method to interpret or explain a model is by necessity incomplete. The goal of such methodology is not to perfectly represent exactly what a model is doing in all instances. Instead, it is to derive enough insight, with some reasonable guarantees, that a practitioner can learn something about the underlying data or domain. My work on feature interaction and training explainable models reflects this. Certainly, the methods proposed in this thesis and in related papers have flaws: they scale poorly to large models and require smoothness approximations to compute. They require approximating intractable integrals and having a meaningful notion of missingness in models not design to handle such notions. However, even with these caveats we can derive insights: from language, images, and a variety of biological and scientific datasets.

The second is that methods in biology can benefit from biological domain knowledge. It might sound obvious to say this out loud, but the trend in machine learning has been increasingly towards large scale datasets with fewer and fewer inductive biases based on the underlying data. Although this approach has been successful, to some degree, in image and natural language, biology offers inherently different and in many ways more challenging problems. The advances in language and vision modeling, while powerful, need thought before being blindly applied to applications in structural biology and drug discovery. Architectures, training objectives, and data representations that worked best for the former domains may not be optimal for the latter, and it behooves those who work at the intersection of the domains to carefully consider the rich structure that has been developed in more classical bioinformatics.

The final is that structural data is both incredibly rich and incredibly challenging to work with. Most of my time working on structural modeling was spent wrestling with the diversity of

available publicly-available structural data and all of its glorious edge cases. The protein data bank is an amazingly forward-thinking project - but due to its heterogeneity, it is also, at times, not as standardized as most machine learning practitioners wish it was. Understanding how to deal with that heterogeneity was probably one of the most valuable skills I learned in my PhD. It feels like rather anticlimactic conclusion to a PhD to say this, but some of the most useful work I did in the last couple years was to better understand and represent messy data. It's not nearly as flashy or exciting as inventing new architectures, but I'm pretty sure that good data parsing will outlive every neural network architecture currently in use today.

BIBLIOGRAPHY

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024.

[2] Rebecca F Alford, Andrew Leaver-Fay, Jeliazko R Jeliazkov, Matthew J O'Meara, Frank P DiMaio, Hahnbeom Park, Maxim V Shapovalov, P Douglas Renfrew, Vikram K Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, 13(6):3031–3048, 2017.

[3] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, page 589333, 2019.

[4] Mohammed AlQuraishi. Proteinnet: a standardized data set for machine learning of protein structure. *BMC bioinformatics*, 20(1):1–10, 2019.

[5] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[6] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[7] Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[8] Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, et al. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, 2021.

[9] Adam P Arkin and Douglas C Youvan. An algorithm for protein engineering: simulations of recursive ensemble mutagenesis. *Proceedings of the National Academy of Sciences*, 89(16):7811–7815, 1992.

[10] Robert J Aumann and Lloyd S Shapley. *Values of non-atomic games*. Princeton University Press, 2015.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[12] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[13] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. In *International Conference on Learning Representations*, 2018.

[14] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*, 2019.

[15] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In *International Conference on Artificial Neural Networks*, pages 63–71. Springer, 2016.

[16] Gino Brunner, Yang Liu, Damián Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. *arXiv preprint*, 2019.

[17] Drew H Bryant, Ali Bashir, Sam Sinai, Nina K Jain, Pierce J Ogden, Patrick F Riley, George M Church, Lucy J Colwell, and Eric D Kelsic. Deep diversification of an aav capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, 2021.

[18] Martin Buttenschoen, Garrett M Morris, and Charlotte M Deane. Posebusters: Ai-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 2024.

[19] Wonkyung Byon, Samira Garonzik, Rebecca A Boyd, and Charles E Frost. Apixaban: a clinical pharmacokinetic and pharmacodynamic review. *Clinical pharmacokinetics*, 58:1265–1279, 2019.

[20] Maria Chatzou, Cedrik Magis, Jia-Ming Chang, Carsten Kemena, Giovanni Bussotti, Ionas Erb, and Cedric Notredame. Multiple sequence alignment modeling: methods and applications. *Briefings in bioinformatics*, 17(6):1009–1023, 2016.

[21] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[22] The UniProt Consortium. Uniprot: the universal protein knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 2023.

[23] Florence Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, 16(22):10881–10890, 1988.

[24] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776*, 2022.

[25] Brian Coventry. *Learning How to Make Mini-Proteins that Bind to Specific Target Proteins*. University of Washington, 2021.

[26] James A Cuff and Geoffrey J Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 34(4):508–519, 1999.

[27] James A Cuff and Geoffrey J Barton. Application of multiple sequence alignment profiles to improve protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 40(3):502–511, 2000.

[28] Tianyu Cui, Pekka Marttinen, and Samuel Kaski. Recovering pairwise interactions using neural networks. *arXiv preprint arXiv:1901.08361*, 2019.

[29] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, page eadd2187, 2022.

[30] 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.

[31] 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.

[32] Kedar Dhamdhere, Ashish Agarwal, and Mukund Sundararajan. The shapley taylor interaction index. *arXiv preprint arXiv:1902.05622*, 2019.

[33] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, pages 13567–13578, 2019.

[34] Sean R. Eddy. Profile hidden markov models. *Bioinformatics (Oxford, England)*, 14(9):755–763, 1998.

[35] Sean R Eddy. Where did the blosum62 alignment score matrix come from? *Nature biotechnology*, 22(8):1035–1036, 2004.

[36] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373, 2006.

[37] Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, et al. The pfam protein families database in 2019. *Nucleic acids research*, 47(D1):D427–D432, 2019.

[38] Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

[39] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[40] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Debsindhu Bhowmik, et al. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *arXiv preprint arXiv:2007.06225*, 2020.

[41] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

[42] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott M Lundberg, and Su-In Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature machine intelligence*, 3(7):620–631, 2021.

[43] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

[44] Robert D Finn, Jody Clements, and Sean R Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2):W29–W37, 2011.

[45] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[46] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

[47] Reza Ghaeini, Xiaoli Z Fern, and Prasad Tadepalli. Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*, 2018.

[48] Vladimir Gligorijevic, Daniel Berenberg, Stephen Ra, Andrew Watkins, Simon Kelow, Kyunghyun Cho, and Richard Bonneau. Function-guided protein design by deep manifold sampling. *bioRxiv*, 2021.

[49] David S Goodsell, Christine Zardecki, Luigi Di Costanzo, Jose M Duarte, Brian P Hudson, Irina Persikova, Joan Segura, Chenghua Shao, Maria Voigt, John D Westbrook, et al. Rcsb protein data bank: Enabling biomedical research and drug discovery. *Protein Science*, 29(1):52–65, 2020.

[50] Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of game theory*, 28(4):547–565, 1999.

[51] Peyton Greenside, Tyler Shimko, Polly Fordyce, and Anshul Kundaje. Discovering epistatic feature interactions from neural network models of regulatory dna sequences. *Bioinformatics*, 34(17):i629–i637, 2018.

[52] Maria Hauser, Martin Steinegger, and Johannes Söding. Mmseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics*, 32(9):1323–1330, 2016.

[53] Brian L Hie, Duo Xu, Varun R Shanker, Theodora UJ Bruun, Payton A Weidenbacher, Shaogeng Tang, and Peter S Kim. Efficient evolution of human antibodies from general protein language models and sequence information alone. *bioRxiv*, 2022.

[54] Desmond G Higgins, Alan J Bleasby, and Rainer Fuchs. Clustal v: improved software for multiple sequence alignment. *Bioinformatics*, 8(2):189–191, 1992.

[55] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[56] Jie Hou, Badri Adhikari, and Jianlin Cheng. Deepsf: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

[57] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. *bioRxiv*, 2022.

[58] Ruth Huey, Garrett M Morris, and Stefano Forli. Using autodock 4 and autodock vina with autodocktools: a tutorial. *The Scripps Research Institute Molecular Graphics Laboratory*, 10550(92037):1000, 2012.

[59] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

[60] Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[61] Joseph D Janizek, Pascal Sturmfels, and Su-In Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *Journal of Machine Learning Research*, 22(104):1–54, 2021.

[62] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. Fastshap: Real-time shapley value estimation. In *International Conference on Learning Representations*, 2021.

[63] David T Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*, 292(2):195–202, 1999.

[64] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[65] Ganesh Chandan Kanakala, Rishal Aggarwal, Divya Nayar, and U Deva Priyakumar. Latent biases in machine learning models for predicting binding affinities using popular data sets. *ACS omega*, 8(2):2389–2397, 2023.

[66] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Segment integrated gradients: Better attributions through regions. *arXiv preprint arXiv:1906.02825*, 2019.

[67] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[68] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[69] Michael Schantz Klausen, Martin Closter Jespersen, Henrik Nielsen, Kamilla Kjærgaard Jensen, Vanessa Isabell Jurtz, Casper Kaae Sønderby, Morten Otto Alexander Sommer, Ole Winther, Morten Nielsen, Bent Petersen, et al. Netsurfp-2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins: Structure, Function, and Bioinformatics*, 87(6):520–527, 2019.

[70] Peter K Koo, Praveen Anand, Steffan B Paul, and Sean R Eddy. Inferring sequence-structure preferences of rna-binding proteins with convolutional residual networks. *bioRxiv*, page 418459, 2018.

[71] Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S Morey-Burrows, Ivan Anishchenko, Ian R Humphreys, et al. Generalized biomolecular modeling and design with rosettafold all-atom. *Science*, 384(6693):eadl2528, 2024.

[72] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[73] Vivian Lai, Jon Z Cai, and Chenhao Tan. Many faces of feature importance: Comparing built-in and post-hoc feature importance in text classification. *arXiv preprint arXiv:1910.08534*, 2019.

[74] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934, 2020.

[75] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2:18, 2010.

[76] Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing: System Demonstrations, pages 121–126, 2017.

[77] 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.

[78] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[79] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[80] Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. *arXiv preprint arXiv:1906.08286*, 2019.

[81] Jia Liu and T Ashton Cropp. Rational protein sequence diversification by multi-codon scanning mutagenesis. In *Enzyme Engineering*, pages 217–228. Springer, 2013.

[82] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[83] Amy X. Lu, Haoran Zhang, Marzyeh Ghassemi, and Alan Moses. Self-supervised contrastive learning of protein representations by mutual information maximization. *bioRxiv*, 2020.

[84] Wei Lu, Qifeng Wu, Jixian Zhang, Jiahua Rao, Chengtao Li, and Shuangjia Zheng. Tankbind: Trigonometry-aware neural networks for drug-protein binding structure prediction. *Advances in neural information processing systems*, 35:7236–7249, 2022.

[85] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020.

[86] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[87] Jianzhu Ma, Sheng Wang, Zhiyong Wang, and Jinbo Xu. Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning. *Bioinformatics*, 31(21):3506–3513, 2015.

[88] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Deep neural language modeling enables functional protein generation across families. *bioRxiv*, 2021.

[89] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

[90] Fábio Madeira, Matt Pearce, Adrian RN Tivey, Prasad Basutkar, Joon Lee, Ossama Edbali, Nandana Madhusoodanan, Anton Kolesnikov, and Rodrigo Lopez. Search and sequence analysis tools services from embl-ebi in 2022. *Nucleic acids research*, 50(W1):W276–W279, 2022.

[91] Rainer Mahrwald and David Evans. *Modern aldol reactions*, volume 1. Wiley Online Library, 2004.

[92] Alex CW May. Percent sequence identity: the need to be explicit. *Structure*, 12(5):737–738, 2004.

[93] Jeremy Minshull, Sridhar Govindarajan, Tony Cox, Jon E Ness, and Claes Gustafsson. Engineered protein function by selective amino acid diversification. *Methods*, 32(4):416–427, 2004.

[94] W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *arXiv preprint arXiv:1801.05453*, 2018.

[95] Bryce M Paschal and Larry Gerace. Identification of ntf2, a cytosolic factor for nuclear import that interacts with nuclear pore complex protein p62. *The Journal of cell biology*, 129(4):925–937, 1995.

[96] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[97] Anita Rácz, Dávid Bajusz, and Károly Héberger. Life beyond the tanimoto coefficient: similarity measures for interaction fingerprints. *Journal of cheminformatics*, 10(1):1–12, 2018.

[98] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, pages 9689–9701, 2019.

[99] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[100] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019.

[101] Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, page 622803, 2019.

[102] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[103] Xavier Robin, Juergen Haas, Rafal Gumienny, Anna Smolinski, Gerardo Tauriello, and Torsten Schwede. Continuous automated model evaluation (cameo)—perspectives on the future of fully automated evaluation of structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1977–1986, 2021.

[104] Gabriel J Rocklin, Tamuka M Chidyausiku, Inna Goreshnik, Alex Ford, Scott Houliston, Alexander Lemak, Lauren Carter, Rashmi Ravichandran, Vikram K Mulligan, Aaron Chevalier, et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, 357(6347):168–175, 2017.

[105] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[106] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

[107] Burkhard Rost. Protein secondary structure prediction continues to rise. *Journal of structural biology*, 134(2-3):204–218, 2001.

[108] Anna J Ruff, Alexander Dennig, and Ulrich Schwaneberg. To get what we aim for–progress in diversity generation methods. *The FEBS journal*, 280(13):2961–2978, 2013.

[109] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[110] Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

[111] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[112] Sofia Serrano and Noah A Smith. Is attention interpretable? *arXiv preprint arXiv:1906.03731*, 2019.

[113] Lloyd S Shapley et al. A value for n-person games. *Contributions to the Theory of Games*, 1953.

[114] Amol V Shivange, Jan Marienhagen, Hemanshu Mundhada, Alexander Schenk, and Ulrich Schwaneberg. Advances in generating functional diversity for directed protein evolution. *Current opinion in chemical biology*, 13(1):19–25, 2009.

[115] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.

[116] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539, 2011.

[117] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[118] Chandan Singh, W James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. *arXiv preprint arXiv:1806.05337*, 2018.

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

[120] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[121] Sergio Filipe Sousa, Pedro Alexandrino Fernandes, and Maria Joao Ramos. Protein–ligand docking: current status and future challenges. *Proteins: Structure, Function, and Bioinformatics*, 65(1):15–26, 2006.

[122] Benjamin Z Stanton, Emma J Chory, and Gerald R Crabtree. Chemically induced proximity in biology and medicine. *Science*, 359(6380):eaao5902, 2018.

[123] Hannes Stärk, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In *International conference on machine learning*, pages 20503–20521. PMLR, 2022.

[124] Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J Haunsberger, and Johannes Söding. Hh-suite3 for fast remote homology detection and deep protein annotation. *BMC bioinformatics*, 20(1):1–15, 2019.

[125] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

[126] Pascal Sturmfels, Roshan Rao, Robert Verkuil, Zeming Lin, Ori Kabeli, Tom Sercu, Adam Lerer, and Alexander Rives. Seq2msa: A language model for protein sequence diversification. *arXiv preprint arXiv:2012.00195*, 2022.

[127] Pascal Sturmfels, Jesse Vig, Ali Madani, and Nazneen Fatema Rajani. Profile prediction: An alignment-based pre-training task for protein sequence models. *arXiv preprint arXiv:2012.00195*, 2020.

[128] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *arXiv preprint arXiv:1908.08474*, 2019.

[129] Mukund Sundararajan and Ankur Taly. A note about: Local explanation methods for deep neural networks lack sensitivity to parameter values. *arXiv preprint arXiv:1806.04205*, 2018.

[130] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[131] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[132] Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[133] Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.

[134] Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? interpretable attribution for feature interactions. *arXiv preprint arXiv:2006.10965*, 2020.

[135] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[136] Mihaly Varadi and Sameer Velankar. The impact of alphafold protein structure database on the fields of life sciences. *Proteomics*, 23(17):2200128, 2023.

[137] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[138] Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. Bertology meets biology: Interpreting attention in protein language models. *arXiv preprint arXiv:2006.15222*, 2020.

[139] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[140] W Patrick Walters, Matthew T Stahl, and Mark A Murcko. Virtual screening—an overview. *Drug discovery today*, 3(4):160–178, 1998.

[141] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[142] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[143] Tuck S Wong, Daria Zhurina, and Ulrich Schwaneberg. The diversity challenge in directed protein evolution. *Combinatorial chemistry & high throughput screening*, 9(4):271–288, 2006.

[144] Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, et al. High-resolution de novo structure prediction from primary sequence. *BioRxiv*, 2022.

[145] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

[146] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.