Browsing in the Library of Babel: Leveraging Evolutionary Information to Improve Protein Modeling

Neil Thomas

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-27
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-27.html

May 1, 2024

Browsing in the Library of Babel:
Leveraging Evolutionary Information to Improve Protein Modeling

by

Neil Thomas

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Yun S. Song, Chair
Professor Ian Holmes
Professor Nilah Ioannidis

Fall 2022

Browsing in the Library of Babel:
Leveraging Evolutionary Information to Improve Protein Modeling

Abstract

Browsing in the Library of Babel:
Leveraging Evolutionary Information to Improve Protein Modeling

by

Neil Thomas

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Yun S. Song, Chair

Proteins are the molecular machines that perform the vast majority of natural biological functions. Discovering proteins to perform novel functions or optimizing them for an existing function are central goals of synthetic biology. Doing so is challenging primarily because for most proteins there is limited understanding of how they function, let alone how to modify them; experimental characterization and crystal structures are expensive and time-consuming to collect. For a given protein, however, genes performing related functions can be found in the the genomes of diverse organisms – the natural result of the process of evolution. With improved techniques for genetic sequencing, an abundance of data deposited in protein sequence databases has become available. This presents a tantalizing modeling opportunity: models that can understand protein function through the observation of related sequences can reduce the reliance on experimental characterization and unlock new possibilities for protein discovery and optimization. Building such models has been a goal of bioinformatics research, and has more recently emerged as a goal of machine learning research. In particular, "protein language models," models trained to learn a distribution over sequence data, have shown promise in predicting functional properties of proteins.

This work leverages the information in protein sequence databases to the following ends. First, it presents a benchmark for the effectiveness of protein language models using a suite of protein prediction tasks. Second, it draws a connection between a well-established graphical model of protein families and the neural network architecture of protein language models. Third, it presents a framework for deriving synthetic protein fitness landscapes from evolutionary data that can be used to evaluate strategies for model-guided protein design *in silico*.

For David, Griffin,
Тевье и Циле,

> Instinct and study; love and hate;
> Audacity — reverence. These must mate,
> And fuse with Jacob's mystic heart,
> To wrestle with the angel — Art.
>
> *Herman Melville*

Contents

List of Figures

List of Tables

Acknowledgments

I have been incredibly lucky to be supported by a large circle of collaborators, friends and family. Without them, none of this would be possible.

First, I have to thank my advisor, Yun S. Song. Yun's Stat 134 course that I took as an undergraduate catalyzed my transition from the chemistry program into applied mathematics and statistics. He has watched me grow from a young undergraduate trying to understand the connection between entropy in statistical mechanics and informational entropy to a graduate student who... is still trying to figure it out. Yun embodies the consummate scientist; he is concerned first and foremost with interesting problems, and lets his boyish curiosity guide him into new fields. He will dive as deeply as necessary into anything, whether it be archaic genomes or ribosomes, or proteins. I feel very lucky to have had him as my advisor.

I can't imagine graduate school without Nick Bhattacharya. Nick and I have worked so closely together that some people struggle to tell us apart. We spent countless hours together working late in the lab, pilfering Yosef Lab Thai food. During the pandemic, there were multiple week stretches where our Zoom cameras didn't turn off, and served as portals into each other's universes. Nick has a great sense of humor, and is incredibly patient; he will also wake you up in the morning saying he rewrote the entire paper overnight. For Nick, any concept is just one afternoon with a textbook away from being grokked. It's such a refreshing approach to research, and to life. He has tried many times to remind me that it's true for me as well, but I always seem to need an extra afternoon. I'm so glad we found each other.

I was really lucky to meet Roshan Rao when I did. Fate brought us together when my housemate, Philippe Laban, told me that he had a labmate also interested in proteins. Roshan is a man who knows how to enjoy the good things in life, like red wine and homemade paneer. He brings the calm confidence of a man who knows he's about to execute. His code is as aesthetic as his homecooked meals, which I'm surprised aren't in the Michelin guide. I continue to learn so much from working with him, and I forgive him for every time he called me Nick.

When the pandemic hit, a lot of things became further away, but one of the bright spots was our completely grassroots Proteinbert research group with Sergey Ovchinnikov and Sam Petti at Harvard, Justas Dauparas at University of Washington, and Peter Koo at Cold Spring Harbor Laboratory. This was an incredibly productive and positive group, and I look forward to remaining colleagues. Sergey is that rare form of wizard where if you ask him a half-baked question, he somehow already has 100 slides on it and two alternative theories of his own.

I spent most of my 4th year working on an enzyme design project as an intern at Google, and I was blessed with amazing mentors: Lucy Colwell, David Belanger, Atish Agarwala, and Abi Ramanan. These are folks who are excited about the work that they do, and always willing to lift each other up. Lucy would spin PyMol structures around with me at 1 AM before telling me I should go rest. David writes the best code reviews I've ever received. Atish was generous with his ideas and let me reinvigorate a project that he had mostly set aside.

Abi has an infectious energy and is an amazing manager. I'm glad I get to keep working with them.

The Song Lab is an inspiring, though sometimes insane, place to work where everyone works on something different. Everyone is incredibly sharp while somehow maintaining an incredible humility. I have to thank Geno Guerra, who mentored me for a summer of undergraduate research. Jeff Spence offered me guidance in the office and never held back with his legs when cycling Morgan Territory. Sanjit Singh Batra and I joined the lab at the same time and have navigated much of the journey of grad school together. So many other folks contributed to the wonderful research environment in Stanley Hall: Jonathan Fischer, Jane Yu, Đan Daniel Erdmann-Pham, Jeff Chan, Gonzalo Benegas, Yutong Wang, Alan Aw, Sebastian Prillo, Yun Deng, Antoine Koehl, Milind Jagota, Carlos Albors, and Junhao (Bear) Xiong.

I was lucky to meet many other folks in my own and adjacent departments that ended up becoming friends: Chloe Hsu, Micah Carroll, Philippe Laban, Chenling Xu, Nicholas Carlini, David Brookes, Alyssa Morrow, Romain Lopez, Michael Levy, Stephen Martis, Jordi Silvestre-Ryan, Clara Wong-Fannjiang, Akosua Busia, Hunter Nisonoff, Michael Chang, Nilesh Tripuraneni, Armin Askari, Adam Gayoso, Deirdre Quillen, Lucas Spangher, Subu Subramanian, Ali Madani, Jacob West-Roberts, and Alex Crits-Christoph.

I have to thank the Pomodobros, aka the Work from Homies: Nick Salazar, Elizabeth McCarthy, and Sanjit Singh Batra. These folks would log on with me every morning for weeks in the era of remote work and help keep me on task through 25 minute Pomodoro work sessions. It sounds so simple, but it works.

I'm blessed with amazing friends: Nat Hendel was always there to riff or get a pastrami Reuben at Saul's. Camille Moore taught me almost everything I know about structural biology and poetry, and saw me through generous eyes when I couldn't see myself. Many other friends supported me, whether it was through a shared meal, bike ride, camping trip, long phone call, or general shenanigans: Gianni Glick, Bill Connacher, Lukas Whaley-Mayda, Kate Boden, Hannah-Laura Hagen, Dannielle McCarthy, Cameron Hearne, AJ Kantor, Serena Gupta, Jacob Dehovitz, Carlee Jensen, Max Tamahori, Alvin Bui, Clea DeCrane, Emily Efland, Travis Hairfield, Kiran Singh, Joe Cackler, Fah Sathirapongsasuti, Karl Heilbron, Hilary Vance, Bo Lopker, Hoai Truong and Sasha Aravkin.

For almost the entirety of my PhD I lived in the Berkeley Student Cooperative at 25-person communal home we call Convent. This was such a beautiful, supportive, and vibrant community, and I'm so lucky to have met so many wonderful humans who are passionate about what they do and willing to share that passion with me over a homecooked meal. Saalem Adera, Paula Salazar, Karl Lindemann, Ali Ohringer, Nikka Singh, Spike Jackson, Wyeth Binder, Sarina Patel, Kat Yarbrough, Holly Burns, Tanner Frank, Scarlett Saunders, Vanessa Ehrenpreis, Mike Shore, Maura Liévano, Vincent Porras, Mohammad Parsa, and many other nuns all brought joy and support to my life.

The Cal Ultimate team was a wonderful (and intense) addition to the PhD during my first two years. It was a way for me to let out some emotions, as well as a way for me to let my shoulder out of its socket, and a variety of tendons out of their insertion points. I want

to thank Rajiv Sambharya, Jesus Avila, Dylan Maxwell, Tommy Lin, as well as many other talented players I got to play with and the dedicated coaching staff. For many nights, the image of a floating disc thrown by Tommy soothed me to sleep. I also want to thank my surgeons Drs. Robert Eppley and William Brown, as well as my physical therapists Ellen DeNeef and Jocelyn Chang for putting me back together, Humpty-Dumpty style.

The PhD journey is so much more about emotional regulation than anyone could have adequately warned me. Thankfully, I had the Grad Men's group, facilitated by Dr. Richard Chiovarelli. Every week a group of graduate students met to offer each other support. This was an important grounding force where I tried to learn by example how to generously offer support to others. This group has made me a better researcher, partner, and person.

Of course, I have to thank my family. My father, Michael, gave up his own PhD aspirations to raise me. He's been my professor since I was old enough to listen to A Shropshire Lad. My mother, Bella, has always modeled a vitality that I've tried to draw on during graduate school. My uncles, Mark and Jack, the ambassadors to my left and right brains (respectively), were always there to cheer me on. So was my extended family: my cousins Sheila, Victoria, Lev, Tom, and Dema, as well as Adek and Natalia Hanukai.

In case they ever see this, I want to thank the light at the top of the Campanile, and the oxidized bust of Athena at the entrance to Doe Library. These two totems have watched over me for my entire journey at Berkeley. I hope I've made you proud.

And to everyone I have forgotten, thank you as well.

Chapter 1

Introduction

1.1 The Library of Babel

In his 1941 short story, The Library of Babel, Jorge Luis Borges imagines a fictional library that contains all permutations of the orthographic symbols [1]:

> *... its bookshelves contain all possible combinations of the twenty-two orthographic symbols (a number which, though unimaginably vast, is not infinite) – that is, all that is able to be expressed, in every language.*

The library is larger than the universe, and contains all possible books. Those who live in the library seek meaning in the books they pull off the shelves and often go mad doing so. Every book that they could ever want is in the library, if they could only find it.

Replace the orthographic symbols with the 20 genetically-encoded amino acids and we are presented with our own protein Library of Babel. The number of possible proteins with 100 amino acids is $20^{100} \approx 10^{130}$, far greater than the number of atoms in the observable universe. Frances Arnold, in her Nobel Prize acceptance speech in 2018 for her work on Directed Evolution [2], referenced Borges's short story to emphasize the difficulty of searching through protein space.

Finding meaningful proteins in this immense combinatorial space is impactful across diverse fields such as green energy, therapeutics, and basic science. Proteins are molecular workhorses that we can think of as "solutions" to biochemical problems. Finding meaningful proteins means enabling new molecular tools like CRISPR gene editing [3], new enzymes for bioremediation [4], and antibodies against emerging viral threats [5].

How do we search the space of all possible proteins to find anything meaningful? For this, we turn to evolution.

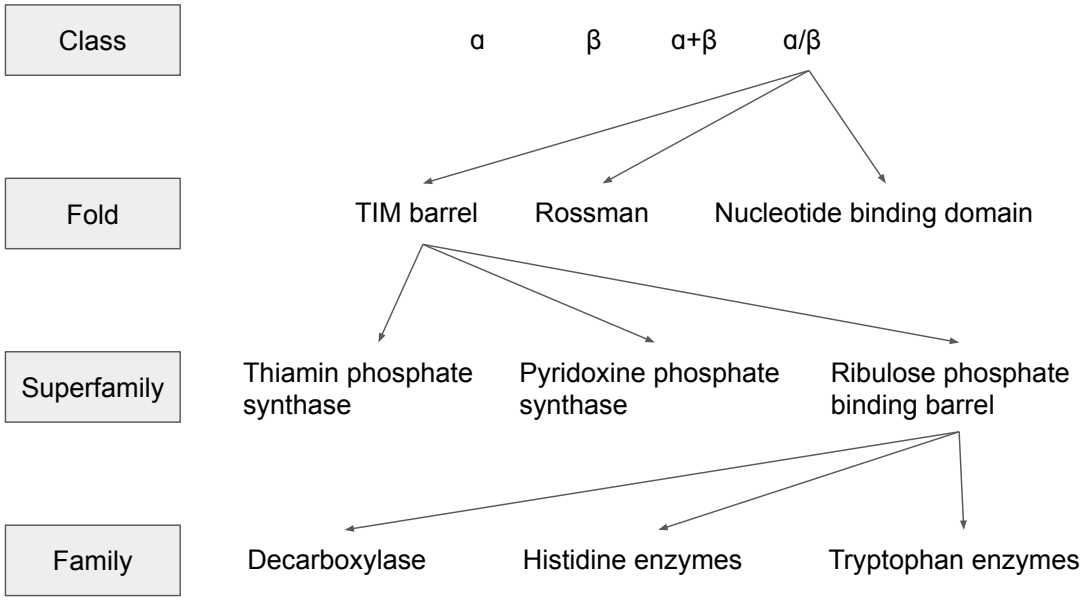


Figure 1.1: An illustration of the Structural Classification Of Proteins (SCOP), showing the hierarchical relationship between related structures.

1.2 Information in Evolution

Nature has been incredibly successful in exploring the vast combinatorial space of possible proteins. "Nature is a tinkerer, not an inventor" [6], and every gene that we discover is related to similar genes in other organisms, derived from common ancestral genes by the natural process of mutation, recombination, and selection. We call these related genes *homologs*. In the same way that human arms and bat wings exhibit homologous bone structures of the humerus and ulna, homologous proteins exhibit common structural folds. The Structural Classification Of Proteins (SCOP) [7] classifies proteins according to a structural taxonomy (see Figure 1.1). At the highest level, proteins in the same *class* share secondary structure composition and order. Proteins in the same *fold* share gross structural similarity. Proteins in the same *superfamily* are hypothesized to have a common ancestor, and proteins within the same *family* have a clear evolutionary relationship and high sequence similarity. We can utilize these relationships to understand novel proteins.

For a given protein, related proteins can be found in sequence databases like UniProt, UniParc, or BFD [8, 9]. These sequence databases growing exponentially as the cost and ease of genetic sequencing decreases. New genes and the proteins they encode are constantly being discovered and deposited, some from entirely new phyla of life [10]. These raw, unannotated protein sequences we think of as *unlabeled* data. We have many orders of magnitude more unlabeled sequences than *labeled* sequences. By way of comparison, there are 500 *million* sequences in UniParc compared to 200 thousand protein crystal structures in PDB [11].

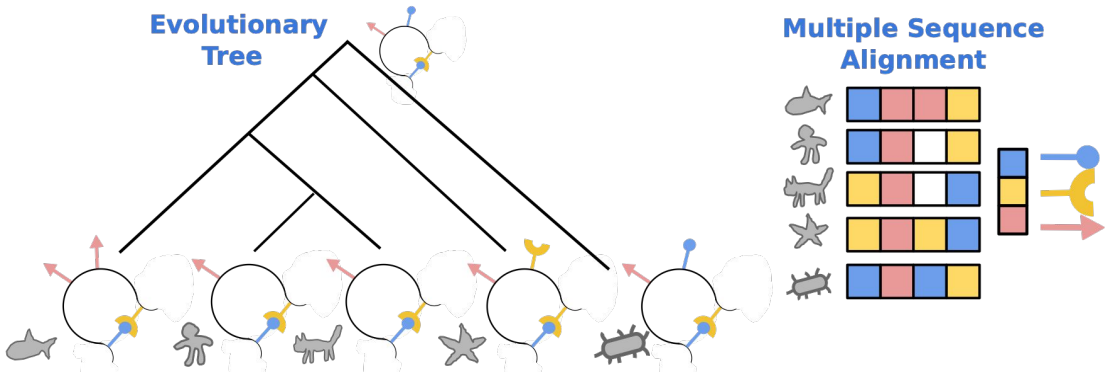


Figure 1.2: Left: An evolutionary tree for a cartoon loop protein with three possible amino acids (blue, yellow, red) at each of four positions. Substitutions, insertions, and deletions accumulate across evolutionary time, resulting in the genes at the leaves of the tree. Right: the multiple sequence alignment derived from this collection of homologous proteins.

1.3 Retrieval and Storage Methods

Reuse is pervasive in biology, and there are hundreds of millions of sequences from the natural process of evolution from which to learn these patterns of reuse. Broadly, there are two approaches to leveraging evolutionary information: *retrieval* and *storage*. In the retrieval approach, a query sequence is scanned against a database to construct a multiple sequence alignment full of homologous sequences, from which we can then extract conservation and coevolution information. In contrast, in the storage approach, a model (such as a pretrained neural network) stores evolutionary information in its parameters. Instead of querying a database, one can query the model directly. The following sections go into slightly more detail on each approach.

Retrieval Methods: Potts and Independent Sites

For a novel protein sequence, the established method for classifying its function and understanding the effect of mutations is to use a *multiple sequence alignment* (MSA). Given a query sequence that we want to understand better, we can use a search tool such as BLAST [12] or HMMer [13] to search a database like UniRef to return a set sequences that are evolutionarily related across potentially disparate species, along with their likely alignment (see Figure 1.2). For a classic overview of sequence alignment methods, see Durbin *et al.* [14]. If the query returns a high-likelihood hit to a sequence with a known fold or function, we can confidently declare our query sequence to have the same fold or function. This is the simplest way to do transfer learning from labeled sequences to unlabeled sequences.

From this multiple sequence alignment, we can derive a further understanding of the function of the protein family. We can derive *conservation* information using a Position-

Specific Scoring Matrix (PSSM). A position is conserved if it rarely changes across evolutionary homologs, i.e., if the position has low entropy in the derived PSSM. The PSSM models each amino acid in the protein as an independent draw from a per-position distribution parameterized by θ_i. For a sequence x of length L, where each amino acid at position i is given by x_i, the likelihood of the sequence under the independent sites model is given by:

$$\mathcal{L}_{PSSM}(x;\theta) = \prod_{i}^{L} p(x_i|\theta_i).$$

We fit the parameters $\theta = \{\theta_1, ..., \theta_L\}$ of this model by maximizing the likelihood of the aligned sequences, which results in matching the first-order statistics (i.e., amino acid occurrence frequencies) in the alignment. Conservation on its own is a powerful feature, telling us which positions have been preserved across evolutionary time. High conservation indicates the position is essential for function, like a charged residue in the active site of an enzyme. Introducing a mutation that drastically reduces the model likelihood by breaking a conserved site will likely destroy the function of the protein.

We can further derive *coevolutionary* information, (i.e., which positions evolve together), from an MSA using a Potts model [15, 16, 17, 18]. A Potts model assigns scores to coupling interactions. A pair of positions are inferred to be co-evolving if the interaction has a high coupling score in the fitted Potts model. It has been shown that high coupling scores are correlated with positions that are close together, or in *contact*, in the 3D structure of the protein [15, 18]. In a Potts model, the likelihood of a sequence is given by a Boltzmann distribution where the statistical energy is given explicitly by the sum of the marginal terms h and coupling scores J:

$$\mathcal{L}_{Potts}(x;h,J) \propto \exp\left[-\sum_{i}^{L} h_i(x_i) - \sum_{i=1}^{L}\sum_{j=1}^{L} J_{ij}(x_i,x_j)\right].$$

Multiple approaches to fitting this model to match the first and second-order statistics of an aligned set of sequences have been explored in the literature, including pseudolikelihood maximization and Boltzmann-machine learning [15, 18, 19, 20].

Potts and PSSMs are examples of alignment-based, single-family models. We call these "retrieval" methods because they rely on a database lookup to retrieve an alignment. Potts models in particular have been shown to be effective at predicting protein contacts [15, 18], predicting variant effects [21, 22], improving downstream structure prediction [23, 24], and generating novel functional sequences [20].

Storage Methods: Protein Language Models

How do we take advantage of the broader patterns of reuse across protein folds and functions? Is there a way to leverage information from more distantly related sequences to help with modeling on sequences for which no good quality alignments exist?

To model multiple families simultaneously and capture higher order interactions, we can use more flexible model classes like neural networks. Since proteins can be encoded as strings of characters, where each character is an amino acid, multiple groups have tried to leverage techniques for learning from documents in natural language processing (NLP) to learn from protein sequences

The amount of unannotated protein sequence data has an analogy in the ever-increasing amount of raw text available on the web. NLP models like GPT-3 [25] and BERT [26] leverage this raw text by combining a pretraining procedure with the Transformer architecture [27], and learn how to perform natural language tasks like question-answering. By pretraining on hundreds of billions of words, scraped from all across the web, these models are able to recapitulate linguistic patterns – syntax and semantics. Pretraining allowed models like GPT-3 to achieve state-of-the-art performance on few-shot learning tasks, (i.e., where labeled data is scarce), such as translations into a target language or sentence parsing.

The key aspect of a *protein* language model is how it is *pretrained*. In order to learn from unlabeled data, unlabeled data must be used for some sort of predictive task. For protein language models, this means training using *masked language modeling*. For millions of protein sequences, a sequence is drawn, a set of positions are masked, and the model is tasked with predicting what the amino acid at the masked positions should be. For a model parameterized by θ, the masked language modeling objective for an input sequence x with masked position indexes M and unmasked positions $x_{\setminus M} = \{x_i | i \notin M\}$ is given by:

$$\mathcal{L}_{MLM}(\theta; x, M) = \sum_{i \in M} \log p_\theta(x_i | x_{\setminus M}).$$

Maximizing this objective is a way of learning a probability distribution over a set of sequences, without necessarily relying on any insights from linguistics.

At each position, prior to outputting the distribution over amino acids, the neural network produces an internal per-position representation of some dimension, which can then be used for other tasks. During pretraining, this representation is used to do masked language modeling, but we can take this same representation and transfer it to some specific task where we have a smaller amount of labeled data.

This approach is part of a framework called *semi-supervised learning*, which combines labeled and unlabeled data to build predictive models. By observing unlabeled data, a model can learn to embed proteins in a high dimensional space where certain manifolds corresponding to function or fold become more obvious. A classifier which takes protein embeddings as input can use the manifolds in the embedding space to make better predictions.

Protein language models, pretrained on large databases of raw sequence data, have shown incredible promise for their ability to predict protein contacts [28], predict variant effects [29], improve downstream structure prediction [30], and generate novel functional sequences [31].

1.4 Dissertation Outline

This dissertation leverages evolutionary information to improve protein modeling, with the goal of enabling applications in structure prediction, functional annotation, and protein design.

Chapter 2 presents a benchmark of protein representations in the semi-supervised setting. It introduces Tasks Assessing Protein Embeddings (TAPE), a set of five biologically relevant tasks spread across different domains of protein biology. It benchmarks a range of approaches to semi-supervised protein representation learning, and finds that, while self-supervised pretraining is helpful for almost all models on all tasks, features from simple retrieval methods such as PSSMs are better for structure prediction tasks..

After the development of this benchmark, an insight from Vig *et al.* [32] revealed how to extract structural contacts directly from protein language models and enabled benchmarking on contact extraction in a fully unsupervised fashion. Building on this insight, Chapter 3 argues that the key operation in neural network architectures for protein language models, that is, *attention* [27], is grounded in real properties of protein family data. It introduces an energy-based attention layer, *factored attention*, which recovers a Potts model in a certain limit. This model is then used to contrast Potts and Transformers. Chapter 3 explores the extent to which the Transformer leverages signal across protein families and shows that Transformer contact prediction performance beats Potts when the number of sequences in the MSA is small.

AlphaFold2's [33] dominant performance at the CASP14 structure prediction competition created additional opportunities for problems in protein function prediction and protein design. To further applications in protein design, Chapter 4 develops a framework for constructing synthetic landscapes for benchmarking pipelines for model-guided protein design. These synthetic landscapes are based on Potts models, are biologically-motivated, and exhibit tunable difficulty. Chapter 4 establishes that without a tuning framework, Potts models are easy to optimize, while landscapes constructed with the tuning framework are sufficiently challenging to benchmark pipelines for model-guided protein design.

Chapter 2

Evaluating Protein Transfer Learning with TAPE

This chapter is based on collaborative work done with Roshan Rao, Nick Bhattacharya, Yan Duan, Xi Chen, John Canny, Pieter Abbeel and Yun S. Song, which was published in the conference proceedings of NeurIPS 2019 [34].

2.1 Introduction

New sequencing technologies have led to an explosion in the size of protein databases over the past decades. These databases have seen exponential growth, with the total number of sequences doubling every two years [8]. Obtaining meaningful labels and annotations for these sequences requires significant investment of experimental resources, as well as scientific expertise, resulting in an exponentially growing gap between the size of protein sequence datasets and the size of annotated subsets. Billions of years of evolution have sampled the portions of protein sequence space that are relevant to life, so large unlabeled datasets of protein sequences are expected to contain significant biological information.

Advances in natural language processing (NLP) have shown that self-supervised learning is a powerful tool for extracting information from unlabeled sequences [35, 26, 25], which raises a tantalizing question: can we adapt NLP-based techniques to extract useful biological information from massive sequence datasets?

To help answer this question, we introduce the Tasks Assessing Protein Embeddings (TAPE), which to our knowledge is the first attempt at systematically evaluating semi-supervised learning on protein sequences. TAPE includes a set of five biologically relevant supervised tasks that evaluate the performance of learned protein embeddings across diverse aspects of protein understanding.

We choose our tasks to highlight three major areas of protein biology where self-supervision can facilitate scientific advances: structure prediction, detection of remote homologs, and protein engineering. We constructed data splits to simulate biologically relevant generalization,

such as a model's ability to generalize to entirely unseen portions of sequence space, or to finely resolve small portions of sequence space. Improvement on these tasks range in application, including designing new antibodies [36] and finding new antimicrobial genes hiding in the so-called "Dark Proteome": tens of millions of sequences with no labels where existing techniques for determining protein similarity fail [37].

We assess the performance of three representative models (recurrent, convolutional, and attention-based) that have performed well for sequence modeling in other fields to determine their potential for protein learning. We also compare two recently proposed semi-supervised models (Bepler et al. [38], Alley et al. [39]). With our benchmarking framework, these models can be compared directly to one another for the first time.

We show that self-supervised pretraining improves performance for almost all models on all downstream tasks. Interestingly, performance for each architecture varies significantly across tasks, highlighting the need for a multi-task benchmark such as ours. We also show that non-deep alignment-based features [40, 41, 42] outperform features learned via self-supervision on secondary structure and contact prediction, while learned features perform significantly better on remote homology detection.

Our results demonstrate that self-supervision for proteins is promising but considerable improvements need to be made before self-supervised models can achieve breakthrough performance. All code and data for TAPE are publicly available[1], and we encourage members of the machine learning community to participate in these exciting problems.

2.2 Background

Protein Terminology

Proteins are linear chains of amino acids connected by covalent bonds. We encode amino acids in the standard 25-character alphabet, with 20 characters for the standard amino acids, 2 for the non-standard amino acids selenocysteine and pyrrolysine, 2 for ambiguous amino acids, and 1 for when the amino acid is unknown [43, 8]. Throughout this paper, we represent a protein x of length L as a sequence of discrete amino acid characters $(x_1, x_2, \ldots, x_L)$ in this fixed alphabet.

Beyond its encoding as a sequence $(x_1, \ldots, x_L)$, a protein has a 3D molecular structure. The different levels of protein structure include *primary* (amino acid sequence), *secondary* (local features), and *tertiary* (global features). Understanding how primary sequence folds into tertiary structure is a fundamental goal of biochemistry [44]. Proteins are often made up of a few large *protein domains*, sequences that are evolutionarily conserved, and as such have a well-defined fold and function.

Evolutionary relationships between proteins arise because organisms must maintain certain functions, such as replicating DNA, as they evolve. Evolution has selected for proteins that are well-suited to these functions. Though structure is constrained by evolutionary pressures,

[1] `https://github.com/songlab-cal/tape`

sequence-level variation can be high, with very different sequences having similar structure. Two proteins that share a common evolutionary ancestor are called *homologs*. Homologous proteins may have very different sequences if they diverged in the distant past.

Quantifying these evolutionary relationships is very important for preventing undesired information leakage between data splits. We mainly rely on *sequence identity*, which measures the percentage of exact amino acid matches between aligned subsequences of proteins [45]. For example, filtering at a 25% sequence identity threshold means that no two proteins in the training and test set have greater than 25% exact amino acid matches. Other approaches besides sequence identity filtering also exist, depending on the generalization the task attempts to test [46].

Modeling Evolutionary Relationships with Sequence Alignments

The key technique for modeling sequence relationships in computational biology is alignment [12, 42, 40, 14]. Given a database of proteins and a query protein at test-time, an alignment-based method uses either carefully designed scoring systems [12] or Hidden Markov Models (HMMs) [42] to align the query protein against all proteins in the database. Good alignments give information about local perturbations to the protein sequence that may preserve, for example, function or structure. The distribution of aligned residues at each position is also an informative representation of each residue that can be fed into downstream models.

Semi-supervised Learning

The fields of computer vision and natural language processing have been dealing with the question of how to learn from unlabeled data for years [47]. Images and text found on the internet generally lack accompanying annotations, yet still contain significant structure. Semi-supervised learning tries to jointly leverage information in the unlabeled and labeled data, with the goal of maximizing performance on the supervised task. One successful approach to learning from unlabeled examples is *self-supervised learning*, which in NLP has taken the form of next token prediction [35], masked token prediction [26], and next sentence classification [26]. Analogously, there is good reason to believe that unlabelled protein sequences contain significant information about their structure and function [44]. Since proteins can be modeled as sequences of discrete tokens, we test both next token and masked token prediction for self-supervised learning.

2.3 Related Work

The most well-known protein modeling benchmark is the Critical Assessment of Structure Prediction (CASP) [48], which focuses on structure modeling. Each time CASP is held, the test set consists of new experimentally validated structures which are held under embargo until the competition ends. This prevents information leakage and overfitting to the test

set. The recently released ProteinNet [49] provides easy to use, curated train/validation/test splits for machine learning researchers where test sets are taken from the CASP competition and sequence identity filtering is already performed. We take the contact prediction task from ProteinNet. However, we believe that structure prediction alone is not a sufficient benchmark for protein models, so we also use tasks not included in the CASP competition to give our benchmark a broader focus.

Semi-supervised learning for protein problems has been explored for decades, with lots of work on kernel-based pretraining [50, 51]. These methods demonstrated that semi-supervised learning improved performance on protein network prediction and homolog detection, but couldn't scale beyond hundreds of thousands of unlabeled examples. Recent work in protein representation learning has proposed a variety of methods that apply NLP-based techniques for transfer learning to biological sequences [52, 38, 39, 53, 54]. In a related line of work, Riesselman et al. [55] trained Variational Auto Encoders on aligned families of proteins to predict the functional impact of mutations. Alley et al. [39] also try to combine self-supervision with alignment in their work by using alignment-based querying to build task-specific pretraining sets.

Due to the relative infancy of protein representation learning as a field, the methods described above share few, if any, benchmarks. For example, both Rives et al. [54] and Bepler et al. [38] report transfer learning results on secondary structure prediction and contact prediction, but they differ significantly in test set creation and data-splitting strategies. Other self-supervised work such as Alley et al. [39] and Yang et al. [56] report protein engineering results, but on different tasks and datasets. With such varied task evaluation, it is challenging to assess the relative merits of different self-supervised modeling approaches, hindering efficient progress.

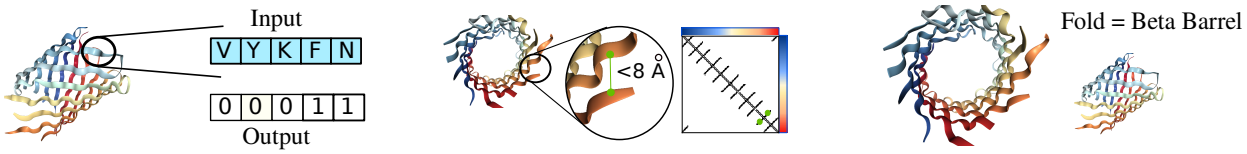
2.4 Datasets

Here we describe our unsupervised pretraining and supervised benchmark datasets. To create benchmarks that test generalization across large evolutionary distances and are useful in real-life scenarios, we curate specific training, validation, and test splits for each dataset. Producing the data for these tasks requires significant effort by experimentalists, database managers, and others. Following similar benchmarking efforts in NLP [57], we describe a set of citation guidelines in our repository[2] to ensure these efforts are properly acknowledged.

Unlabeled Sequence Dataset

We use Pfam [58], a database of thirty-one million protein domains used extensively in bioinformatics, as the pretraining corpus for TAPE . Sequences in Pfam are clustered into evolutionarily-related groups called *families*. We leverage this structure by constructing a test set of fully heldout families (see Appendix A.5 for details on the selected families), about 1%

[2] `https://github.com/songlab-cal/tape#citation-guidelines`



(a) Secondary Structure (b) Contact Prediction (c) Remote Homology

Figure 2.1: Structure and Annotation Tasks on protein KgdM Porin (pdbid: 4FQE). (a) Viewing this Porin from the side, we show secondary structure, with the input amino acids for a segment (blue) and corresponding secondary structure labels (yellow and white). (b) Viewing this Porin from the front, we show a contact map, where entry i, j in the matrix indicates whether amino acids at positions i, j in the sequence are within 8 angstroms of each other. In green is a contact between two non-consecutive amino acids. (c) The fold-level remote homology class for this protein.

of the data. For the remaining data we construct training and test sets using a random 95/5% split. Perplexity on the uniform random split test set measures in-distribution generalization, while perplexity on the heldout families test set measures out-of-distribution generalization to proteins that are less evolutionarily related to the training set.

Supervised Datasets

We provide five biologically relevant downstream prediction tasks to serve as benchmarks. We categorize these into structure prediction, evolutionary understanding, and protein engineering tasks. The datasets vary in size between 8 thousand and 50 thousand training examples (see Table A.1 for sizes of all training, validation and test sets). Further information on data processing, splits and experimental challenges is in Appendix A.1. For each task we provide:

(Definition) A formal definition of the prediction problem, as well as the source of the data.
(Impact) The impact of improving performance on this problem.
(Generalization) The type of understanding and generalization desired.
(Metrics) The metric reported in Table 2.2 to report results and additional metrics presented in Appendix A.8

Task 1: Secondary Structure (SS) Prediction (Structure Prediction Task)
(Definition) Secondary structure prediction is a sequence-to-sequence task where each input amino acid x_i is mapped to a label $y_i \in \{\text{Helix}(H), \text{Strand}(E), \text{Other}(C)\}$. See Figure 2.1a for illustration. The data are from Klausen et al. [59].
(Impact) SS is an important feature for understanding the function of a protein, especially if the protein of interest is not evolutionarily related to proteins with known structure [59].

SS prediction tools are very commonly used to create richer input features for higher-level models [60].

(Generalization) SS prediction tests the degree to which models learn local structure. Data splits are filtered at 25% sequence identity to test for broad generalization.

(Metrics) We report accuracy on a per-amino acid basis on the CB513 [61] dataset. We further report three-way and eight-way classification accuracy for the test sets CB513, CASP12, and TS115.

Task 2: Contact Prediction (Structure Prediction Task)

(Definition) Contact prediction is a pairwise amino acid task, where each pair x_i, x_j of input amino acids from sequence x is mapped to a label $y_{ij} \in \{0, 1\}$, where the label denotes whether the amino acids are "in contact" (< 8Å apart) or not. See Figure 2.1b for illustration. The data are from the ProteinNet dataset [49].

(Impact) Accurate contact maps provide powerful global information; e.g., they facilitate robust modeling of full 3D protein structure [62]. Of particular interest are medium- and long-range contacts, which may be as few as twelve sequence positions apart, or as many as hundreds apart.

(Generalization) The abundance of medium- and long-range contacts makes contact prediction an ideal task for measuring a model's understanding of global protein context. We select the data splits that was filtered at 30% sequence identity to test for broad generalization.

(Metrics) We report precision of the $L/5$ most likely contacts for medium- and long-range contacts on the ProteinNet CASP12 test set, which is a standard metric reported in CASP [48]. We further report Area under PR Curve and Precision at L, $L/2$, and $L/5$ for short-range, medium-range and long-range contacts in Appendix A.

Task 3: Remote Homology Detection (Evolutionary Understanding Task)

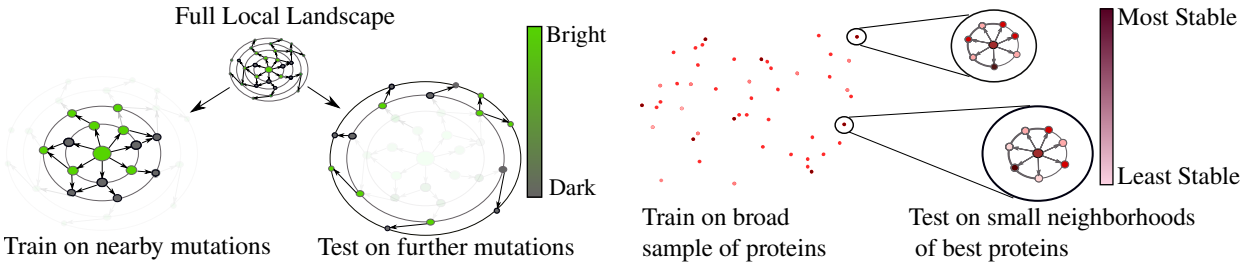
(Definition) This is a sequence classification task where each input protein x is mapped to a label $y \in \{1, \ldots, 1195\}$, representing different possible protein folds. See Figure 2.1c for illustration. The data are from Hou et al. [63].

(Impact) Detection of remote homologs is of great interest in microbiology and medicine; e.g., for detection of emerging antibiotic resistant genes [64] and discovery of new CAS enzymes [3].

(Generalization) Remote homology detection measures a model's ability to detect structural similarity across distantly related inputs. We hold out entire evolutionary groups from the training set, forcing models to generalize across large evolutionary gaps.

(Metrics) We report overall classification accuracy on the fold-level heldout set from Hou et al. [63]. We further report top-one and top-five accuracy for fold-level, superfamily-level and family-level holdout sets in Appendix A.

Task 4: Fluorescence Landscape Prediction (Protein Engineering Task)



(a) Fluorescence (b) Stability

Figure 2.2: Protein Engineering Tasks. In both tasks, a parent protein p is mutated to explore the local landscape. As such, dots represent proteins and directed arrow $x \to y$ denotes that y has exactly one more mutation than x away from parent p. (a) The Fluorescence task consists of training on small neighborhood of the parent green fluorescent protein (GFP) and then testing on a more distant proteins. (b) The Stability task consists of training on a broad sample of proteins, followed by testing on one-mutation neighborhoods of the most promising sampled proteins.

(Definition) This is a regression task where each input protein x is mapped to a label $y \in \mathbb{R}$, corresponding to the log-fluorescence intensity of x. See Figure 2.2a for illustration. The data are from Sarkisyan et al. [65].

(Impact) For a protein of length L, the number of possible sequences m mutations away is $O(L^m)$, a prohibitively large space for exhaustive search via experiment, even if m is modest. Moreover, due to epistasis (second- and higher-order interactions between mutations at different positions), greedy optimization approaches are unlikely to succeed. Accurate computational predictions could allow significantly more efficient exploration of the landscape, resulting in better optima. Machine learning methods have already seen some success in related protein engineering tasks [66].

(Generalization) The fluorescence prediction task tests the model's ability to distinguish between very similar inputs, as well as its ability to generalize to unseen combinations of mutations. The train set is a Hamming distance-3 neighborhood of the parent green fluorescent protein (GFP), while the test set has variants with four or more mutations. Hamming distance is measured at the amino acid level.

The choice of Hamming distance between amino acids does not always reflect evolution, since not all proteins at the same Hamming distance correspond to equal "evolutionary" distance in the sense of number of nucleotide substitutions. Since we are trying to highlight the protein engineering setting, we believe that this is an important feature of the Fluorescence task. Our goal is to test the models' ability to accurately predict phenotype as a function of an input molecule (e.g. one presented by a protein designer)

(Metrics) We report Spearman's ρ (rank correlation coefficient) on the test set. We further

	Random Families			Heldout Families			Heldout Clans		
	Acc	Perp	ECE	Acc	Perp	ECE	Acc	Perp	ECE
Transformer	**0.45**	**8.89**	**6.01**	**0.35**	**11.77**	**8.87**	**0.28**	**13.54**	10.76
LSTM	0.40	**8.89**	6.94	0.24	13.03	12.73	0.13	15.36	16.94
ResNet	0.41	10.16	6.86	0.31	13.19	9.77	**0.28**	13.72	**10.62**
Bepler et al. [38]	0.28	11.62	10.17	0.19	14.44	14.32	0.12	15.62	17.05
Alley et al. [39]	0.32	11.29	9.08	0.16	15.53	15.49	0.11	16.69	17.68
Random	0.04	25	25	0.04	25	25	0.04	25	25

Table 2.1: Language modeling metrics: Language Modeling Accuracy (Acc), Perplexity (Perp) and Exponentiated Cross-Entropy (ECE).

report MSE and Spearman's ρ for the full test set, only bright proteins, and only dark proteins in Appendix A.

Task 5: Stability Landscape Prediction (Protein Engineering Task)

(Definition) This is a regression task where each input protein x is mapped to a label $y \in \mathbb{R}$ measuring the most extreme circumstances in which protein x maintains its fold above a concentration threshold (a proxy for intrinsic stability). See Figure 2.2b for illustration. The data are from Rocklin et al. [67].

(Impact) Designing stable proteins is important to ensure, for example, that drugs are delivered before they are degraded. More generally, given a broad sample of protein measurements, finding better refinements of top candidates is useful for maximizing yield from expensive protein engineering experiments.

(Generalization) This task tests a model's ability to generalize from a broad sampling of relevant sequences and to localize this information in a neighborhood of a few sequences, inverting the test-case for fluorescence above. The train set consists of proteins from four rounds of experimental design, while the test set contains Hamming distance-1 neighbors of top candidate proteins.

(Metrics) We report Spearman's ρ on the test set. In Appendix A we also assess classification of a mutation as stabilizing or non-stabilizing. We report Spearman's ρ and accuracy for this task broken down by protein topology in Appendix A.

2.5 Models and Experimental Setup

Losses: We examine two self-supervised losses that have seen success in NLP. The first is *next-token prediction*, which models $p(x_i \mid x_1, \ldots, x_{i-1})$. Since many protein tasks are sequence-to-sequence and require bidirectional context, we apply a variant of next-token prediction which additionally trains the reverse model, $p(x_i \mid x_{i+1}, \ldots, x_L)$, providing full

context at each position (assuming a Markov sequence). The second is *masked-token prediction* [26], which models $p(x_{\text{masked}} \mid x_{\text{unmasked}})$ by replacing the value of tokens at multiple positions with alternate tokens.

Protein-specific loss: In addition to self-supervised algorithms, we explore another protein-specific training procedure proposed by Bepler et al. [38]. They suggest that further *supervised* pretraining of models can provide significant benefits. In particular, they propose supervised pretraining on contact prediction and remote homology detection, and show it increases performance on secondary structure prediction. Similar work in computer vision has shown that supervised pretraining can transfer well to other tasks, making this a promising avenue of exploration [68].

Architectures and Training: We implement three architectures: an LSTM [69], a Transformer [27], and a dilated residual network (ResNet) [70]. We use a 12-layer Transformer with a hidden size of 512 units and 8 attention heads, leading to a 38M-parameter model. Hyperparameters for the other models were chosen to approximately match the number of parameters in the Transformer. Our LSTM consists of two three-layer LSTMs with 1024 hidden units corresponding to the forward and backward language models, whose outputs are concatenated in the final layer, similar to ELMo [35]. For the ResNet we use 35 residual blocks, each containing two convolutional layers with 256 filters, kernel size 9, and dilation rate 2. We chose these hyperparameters based on common choices from the literature. Our supervised tasks are of similar size to most of those in the GLUE [71] benchmark, which has been instrumental in demonstrating the success of self-supervision in NLP. Since the models that were applied to GLUE have tens to hundreds of millions of parameters, we chose to make our models roughly the same size. See Appendix A.7 for model size ablation experiments. See Appendix A.2 for details of how these pretrained models are fed into downstream tasks.

In addition, we benchmark two previously proposed architectures that differ significantly from the three above. The first, proposed by Bepler et al. [38], is a two-layer bidirectional language model, similar to the LSTM discussed above, followed by three 512 hidden unit bidirectional LSTMs. The second, proposed by Alley et al. [39], is a unidirectional mLSTM [72] with 1900 hidden units. Details on implementing and training these architectures can be found in the original papers.

The Transformer and ResNet are trained with masked-token prediction, while the LSTM is trained with next-token prediction. Both Alley et al. and Bepler et al. are trained with next-token prediction. All self-supervised models are trained on four NVIDIA V100 GPUs for one week.

Baselines: We evaluate learned features against two baseline featurizations. The first is a one-hot encoding of the input amino acid sequence, which provides a simple baseline. Most current state-of-the-art algorithms for protein classification and regression take advantage of alignment or HMM-based inputs (see Section 2.2). Alignments can be transformed into

various features, such as mutation probabilities [63] or the HMM state-transition probabilities [59] for each amino acid position. These are concatenated to the one-hot encoding of the amino acid to form another baseline featurization. For our baselines we use alignment-based inputs that vary per task depending on the inputs used by the current state-of-the-art method. See Appendix A.3 for details on the alignment-based features used for each task. We do not use alignment-based inputs for protein engineering tasks. Proteins in the engineering datasets differ by only a single amino acid, while alignment-based methods search for proteins with high sequence identity, so alignment-based methods return the same set of features for all proteins we wish to distinguish between.

Experimental Setup: The goal of our experimental setup is to systematically compare all featurizations. For each task we select a particular supervised architecture, drawing from state-of-the-art where available, and make sure that fine-tuning on all language models is identical. See Appendix A.3 for details on supervised architectures and training.

2.6 Results

Table 2.1 contains accuracy, perplexity, and exponentiated cross entropy (ECE) on the language modeling task for the five architectures we trained with self-supervision as well as a random model baseline. We report metrics on both the random split and the fully heldout families. Supervised LSTM metrics are reported after language modeling pretraining, but before supervised pretraining. Heldout family accuracy is consistently lower than random-split accuracy, demonstrating a drop in the out-of-distribution generalization ability. Note that although some models have lower perplexity than others on both random-split and heldout sets, this lower perplexity does not necessarily correspond to better performance on downstream tasks. This replicates the finding in Rives et al. [54].

Table 2.2 contains results for all benchmarked architectures and training procedures on all downstream tasks in TAPE . We report accuracy, precision, or Spearman's ρ, depending on the task, so higher is always better and each metric has a maximum value of 1.0. See Section 2.4 for the metric reported in each task. Detailed results and metrics for each task are in Appendix A.8.

We see from Table 2.2 that self-supervised pretraining improves overall performance across almost all models and all tasks. Further analysis reveals aspects of these tasks with room for significant improvement. In the fluorescence task, the distribution is bimodal with a mode of bright proteins and a mode of dark proteins (see Figure 2.3). Since one goal of using machine learning models in protein engineering is to screen potential variants, it is important for these methods to successfully distinguish between beneficial and deleterious mutations. Figure 2.3 shows that the model does successfully perform some clustering of fluorescent proteins, but that many proteins are still misclassified.

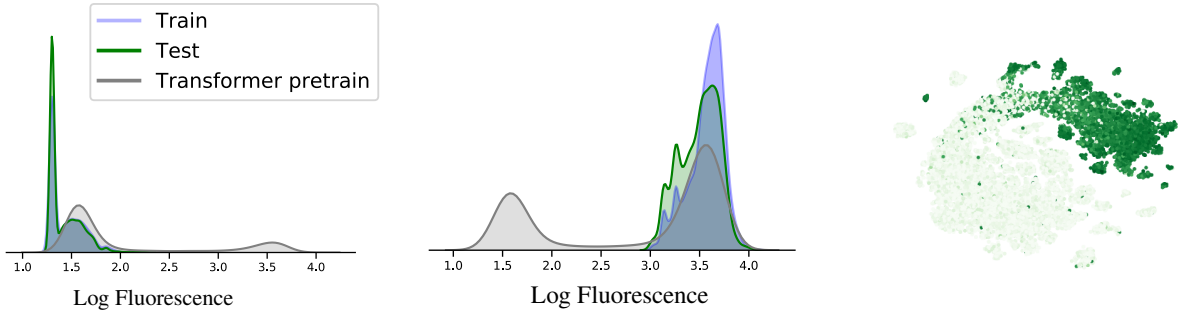
For the stability task, to identify which mutations a model believes are beneficial, we use the parent protein as a decision boundary and label a mutation as beneficial if its predicted

Method		Structure		Evolutionary	Engineering	
		SS	Contact	Homology	Fluorescence	Stability
No Pretrain	Transformer	0.70	0.32	0.09	0.22	-0.06
	LSTM	0.71	0.19	0.12	0.21	0.28
	ResNet	0.70	0.20	0.10	-0.28	0.61
Pretrain	Transformer	0.73	0.36	0.21	**0.68**	**0.73**
	LSTM	0.75	0.39	**0.26**	0.67	0.69
	ResNet	0.75	0.29	0.17	0.21	**0.73**
	Bepler [38]	0.73	0.40	0.17	0.33	0.64
	Alley [39]	0.73	0.34	0.23	0.67	**0.73**
Baseline	One-hot	0.69	0.29	0.09	0.14	0.19
	Alignment	**0.80**	**0.64**	0.09	N/A	N/A

Table 2.2: Results on downstream supervised tasks

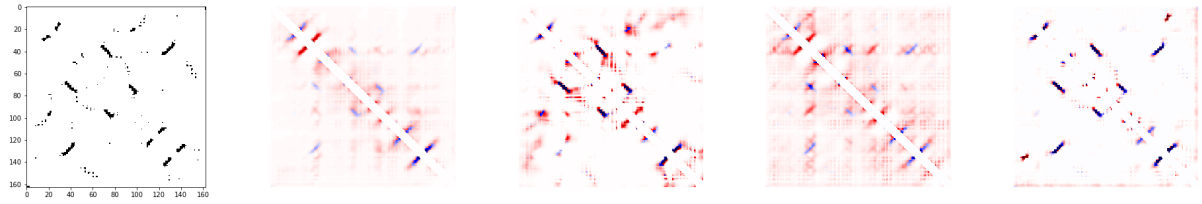
stability is higher than the parent's predicted stability. We find that our best pretrained model achieves 70% accuracy in making this prediction while our best non-pretrained model achieves 68% accuracy (see Table A.9 for full results). Improving the ability to distinguish beneficial from deleterious mutations would make these models much more useful in real protein engineering experiments.

In the contact prediction task, long-range contacts are of particular interest and can be hundreds of positions apart. Figure 2.4 shows the predictions of several models on a protein where the longest range contact occurs between the 8th and 136th amino acids. Pretraining



(a) Dark Mode (b) Bright Mode (c) Embedding t-SNE

Figure 2.3: Distribution of training, test, and pretrained Transformer predictions on the dark and bright modes, along with t-SNE of pretrained Transformer protein embeddings colored by log-fluorescence.



(a) True Contacts (b) LSTM (c) LSTM Pretrain (d) One Hot (e) Alignment

Figure 2.4: Predicted contacts for chain 1A of a Bacterioferritin comigratory protein (pdbid: 3GKN). Blue indicates true positive contacts while red indicates false positive contacts. Darker colors represent more certainty from the model.

helps the model capture more long-range information and improves the overall resolution of the predicted map. However, the hand-engineered alignment features result in a much sharper map, accurately resolving more long-range contacts. This increased specificity is highly relevant in structure prediction pipelines [23, 62] and highlights a clear challenge for pretraining.

2.7 Discussion

Comparison to state of the art. As shown in Table 2.2, alignment-based inputs can provide a powerful signal that outperforms current self-supervised models on multiple tasks. Current state-of-the-art prediction methods for secondary structure prediction, contact prediction, and remote homology classification all take in alignment-based inputs. These methods combine alignment-based inputs with other techniques (e.g. multi-task training, kernel regularization) to achieve an additional boost in performance. For comparison, NetSurfP-2.0 [59] achieves 85% accuracy on the CB513 [61] secondary structure dataset, compared to our best model's 75% accuracy, RaptorX [73] achieves 0.69 precision at $L/5$ on CASP12 contact prediction, compared to our best model's 0.49, and DeepSF [63] achieves 41% accuracy on remote homology detection compared to our best model's 26%.

Need for multiple benchmark tasks. Our results support our hypothesis that multiple tasks are required to appropriately benchmark performance of a given method. Our Transformer, which performs worst of the three models in secondary structure prediction, performs best on the fluorescence and stability tasks. The reverse is true of our ResNet, which ties the LSTM in secondary structure prediction but performs far worse for the fluorescence task, with a Spearman's ρ of 0.21 compared to the LSTM's 0.67. This shows that performance on a single task does not capture the full extent of a trained model's knowledge and biases, creating the need for multi-task benchmarks such as TAPE .

Chapter 3

Interpreting Potts and Transformer Protein Models Through the Lens of Simplified Attention

This chapter is based on collaborative work done with Nick Bhattacharya, Roshan Rao, Justas Dauparas, David Baker, Peter Koo, Yun S. Song, and Sergey Ovchinnikov, which was published in the conference proceedings of the 2022 Pacific Symposium on Biocomputing [74].

3.1 Introduction

Inferring protein structure from sequence is a longstanding problem in computational biochemistry. Potts models, a particular kind of Markov Random Field (MRF), are the predominant unsupervised method for modeling interactions between amino acids. Potts models are trained to maximize pseudolikelihood on alignments of evolutionarily related proteins [75, 76, 77]. Features derived from Potts models were the main drivers of improved performance at the CASP11 competition [78]. Potts models were subsequently used as input features for top performing supervised neural network models in CASP13 [79, 80, 24].

Inspired by the success of BERT [26], GPT [25] and related unsupervised models in NLP, a line of work has emerged that learns features of proteins through self-supervised pretraining [54, 81, 34, 31, 82]. This new approach trains Transformer [27] models on large datasets of protein sequences. Pretrained model performance raises questions about the importance of data and model scale [83, 81], whether neural features compete with evolutionary features extracted by established bioinformatic methods [34], and the benefits of transfer learning [84, 22, 85].

In CASP14, Alphafold2 achieved breakthrough performance by replacing the Potts model with an attention-based model that directly used the MSA as input [33]. This approach was adapted subsequently in RoseTTAFold [86]. The performance of these methods established attention as state-of-the-art for extracting features from MSAs. This raises a natural question

of how Potts models and attention mechanisms are related.

In this paper, we investigate the ways in which attention-based models and Potts models trained on alignments can learn meaningful interactions in biological sequence data. To do so, we introduce a simplified energy-based attention model trained on alignments, *factored attention*, which interpolates between the standard attention mechanism and Potts models. We show that factored attention can successfully share parameters across positions within a family or share amino acid features across hundreds of families.

3.2 Background

Proteins are polymers composed of amino acids and are commonly represented as strings. Along with this 1D sequence representation, each protein folds into a 3D physical structure. Physical distance between positions in 3D is often a much better indicator of functional interaction than proximity in sequence. One representation of physical distance is a *contact map* C, a symmetric matrix in which entry $C_{ij} = 1$ if the beta carbons[1] of i and j are within 8Å of one another, and 0 otherwise.

Multiple Sequence Alignments. To understand structure and function of a protein sequence, one typically assembles a set of its evolutionary relatives and looks for patterns within the set. A set of related sequences is referred to as a *protein family*, commonly represented by a Multiple Sequence Alignment (MSA). Gaps in aligned sequences correspond to insertions from an alignment algorithm [87, 41], ensuring that positions with similar structure and function line up for all members of the family. After aligning, sequence position carries significant evolutionary, structural, and functional information.

Coevolutionary Analysis of Protein Families. The observation that statistical patterns in MSAs can be used to predict couplings has been widely used to infer structure and function from protein families [88, 89, 90, 91].

3.3 Methods

To explore how attention and Potts models learn interactions in protein sequence data, we compare a number of unsupervised methods which learn contacts with sequence-modeling objectives. Many of these methods are based on the formalism of Markov Random Fields (MRFs). We do not extend our analysis to *supervised* contact prediction models which take MRF features as input, as these are outside the scope of this work.

Throughout this section, $x = (x_1, \ldots, x_L)$ is a sequence of length L from an alphabet of size A. This sequence is part of an MSA of length L with N total sequences. Recall that a

[1] In the case of glycine, the alpha carbon is used.

fully-connected Pairwise MRF over p variables $X_1, \ldots, X_p$ specifies a distribution

$$p_\theta(x_1, \ldots, x_p) = \frac{1}{Z} \exp\left(\sum_{i<j} E_\theta(x_i, x_j) \right), \tag{3.1}$$

where Z is the partition function and $E_\theta(x_i, x_j)$ is an arbitrary function of i, j, x_i and x_j. For all models below, we can introduce an explicit functional $E_\theta(x_i)$ to capture the marginal distribution of X_i. When introduced, we parametrize the marginal with $E_\theta(x_i) = b_{i,x_i}$ for $b \in \mathbb{R}^{L \times A}$.

Potts Models

A Potts model is a fully-connected pairwise MRF with L variables, each representing a position in the MSA. An edge (i, j) is parametrized with a matrix $W^{ij} \in \mathbb{R}^{A \times A}$. These matrices are organized into an order-4 tensor which form the parameters of a Potts model. Note that $W^{ij} = W^{ji}$. The energy functional of a Potts model is given through lookups, namely

$$E_\theta(x_i, x_j) = W^{ij}(x_i, x_j). \tag{3.2}$$

Factored Attention

Factored attention has two advantages over Potts for modeling protein families: it shares a pool of amino acid feature matrices across all positions and it estimates $\mathcal{O}(L)$ parameters instead of $\mathcal{O}(L^2)$.

Sharing amino acid features. Many contacts in a protein are driven by similar interactions between amino acids, such as many types of weakly polar interactions [92, 93]. If two pairs of positions (i, j) and (l, m) are both in contact due to the same interaction, a Potts model must estimate completely separate amino acid features W^{ij} and W^{lm}. In order to share amino acid features, we want to compute all energies from one pool of $A \times A$ feature matrices. The simplest way to accomplish this is by associating an $L \times L$ matrix $\mathcal{A}$ to every $A \times A$ feature matrix W_V. For H such pairs $(\mathcal{A}, W_V)$, we could introduce a factorized MRF:

$$E_\theta(x_i, x_j) = \sum_{h=1}^{H} \operatorname{symm}\left(\operatorname{softmax}\left(\mathcal{A}^h\right)\right)_{ij} W_V^h(x_i, x_j). \tag{3.3}$$

A row-wise softmax is taken to encourage sparse interactions and aid in normalization. This model allows the pairs (i, j) and (l, m) to reuse a single feature W_V^h, assuming $\mathcal{A}_{ij}^h$ and $\mathcal{A}_{lm}^h$ are both large.

Scaling linearly in length. Both Potts and the factorized model in Equation 3.3 have $\mathcal{O}(L^2)$ parameters. However, contacts are observed to grow linearly over the wide range of protein structures currently available [94, 95]. Given that the number of interactions

we wish to estimate grows linearly in length, the quadratic scaling of these models can be greatly improved. One way to fix this is by introducing the factorization $\mathcal{A} = W_Q W_K^T$, where $W_Q, W_K \in \mathbb{R}^{L\times d}$. We use the subscripts Q, K, and V in analogy with the "Query", "Key", and "Value" nomenclature from the attention literature [27]. As before, we employ a row-wise softmax for sparsity and normalization. Combining feature sharing with linear length scaling leads to *factored attention*, defined in Equation 3.4.

Like Potts, factored attention is a fully-connected pairwise MRF with L variables. The parameters of this model consist of H triples (W_Q, W_K, W_V), where $W_Q, W_K \in \mathbb{R}^{L\times d}$; $W_V \in \mathbb{R}^{A\times A}$; and d is a hyperparameter. Each such triple is called a *head* and d is the *head size*. Unlike a Potts model, the parameters for each edge (i, j) are tied through the use of heads. The energy functional is

$$E_\theta(x_i, x_j) = \sum_{h=1}^{H} \operatorname{symm}\left(\operatorname{softmax}\left(W_Q^h W_K^{h\,T}\right)\right)_{ij} W_V^h(x_i, x_j), \tag{3.4}$$

where $\operatorname{symm}(M) = (M + M^T)/2$ ensures the positional interactions are symmetric.

Adding sequence-dependent interactions leads to standard attention, see Appendix B.1.

Single-layer attention

Our *single-layer attention* model consists of a single Transformer encoder layer: an attention layer followed by a dense layer, with layer normalization [96] to aid in optimization. Transformer implementations typically use a sine/cosine positional encoding [27] or learned Gaussian positional encoding [97], rather than the one-hot positional encoding used in our single-layer models.

Self-Supervised Losses. Given an MSA, many standard methods estimate Potts model parameters through pseudolikelihood maximization [95, 76]. On the other hand, BERT-like attention-based models are typically trained with variants of masked language modeling [26]. Pseudolikelihood is challenging to compute efficiently for generic models, unlike the masked language modeling loss. Both of these losses require computing conditionals of the form $p_\theta(x_i|x_{\setminus M})$, where M is a subset of $\{1, \ldots, L\}$ containing i. The losses $\mathcal{L}_{PL}$ and $\mathcal{L}_{MLM}$ for pseudolikelihood and masked language modeling, respectively, are

$$\mathcal{L}_{PL}(\theta; x) = \sum_{i=1}^{L} \log p_\theta(x_i|x_{\setminus i}), \qquad \mathcal{L}_{MLM}(\theta; x, M) = \sum_{i\in M} \log p_\theta(x_i|x_{\setminus M}).$$

Regularization for Potts and factored attention are both based on MRF edge parameters, while single-layer attention is penalized using weight decay. More details can be found in Appendix B.2.

Pretraining on Sequence Databases

All single-layer models are trained on a set of evolutionarily related sequences. Given a large database of protein sequences such as UniRef100 [98] or BFD [9, 99], these models cannot be trained until significant preprocessing has been done: clustering, dereplication of highly related sequences, and alignment to generate an MSA for each cluster. In contrast, the self-supervised approach taken by works such as Refs. [81, 54, 34, 31] applies BERT-style pretraining directly on the database of proteins with minimal preprocessing.

Given a new sequence of interest and a database of sequences, single-family models require more steps for inference than pretrained Transformers. To apply a single-family model, one must query the database for related sequences, dereplicate the set, align sequences into an MSA, then train a model to learn contacts. On the other hand, a Transformer pretrained on the database simply computes a forward pass for the sequence of interest and its attention activations are used to predict contacts. No explicit querying or aligning is performed.

Extracting Contacts

Potts. We follow standard practice and extract a contact map $\widehat{C} \in \mathbb{R}^{L \times L}$ from the order-4 interaction tensor W by setting $\widehat{C}_{ij} = \|W^{ij}\|_F$.

Factored Attention. Since factored attention is a pairwise MRF, we can compute its order-4 interaction tensor W and use the same procedure as Potts. See Equation B.2.

Single-Layer Attention. To produce contacts for an MSA, we compute attention maps from *only* the positional encoding (without sequence) and average attention maps from all heads. Each single-layer attention model is trained on one MSA, so the positional encoding is a feature shared by all sequences in the MSA.

ProtBERT-BFD. We extract contacts from ProtBERT by averaging a subset of attention maps for an input sequence x. Of the 16 heads in 30 layers, we selected six whose attention maps had the top individual contact precisions over 500 families randomly selected from the Yang *et al.* [80] dataset. Predicted contacts for x are given by averaging the $L \times L$ attention maps from these six heads, then symmetrizing additively. See Appendix Table B.1.

Average Product Correction (APC). Empirically, Potts models trained with Frobenius norm regularization have artifacts in the outputs $\widehat{C}$. These are removed with the Average Product Correction (APC) [100]. Unless otherwise stated, we apply APC to all extracted contacts.

3.4 Results

Experimental Setup. We use a set of 748protein families from Ref. [80] to evaluate all models. For Potts models and single attention layers, we train separate models on each individual MSA. ProtBERT-BFD is frozen for all experiments. We train models using PyTorch Lightning [101] and Weights and Biases [102]. We extract contacts from each model

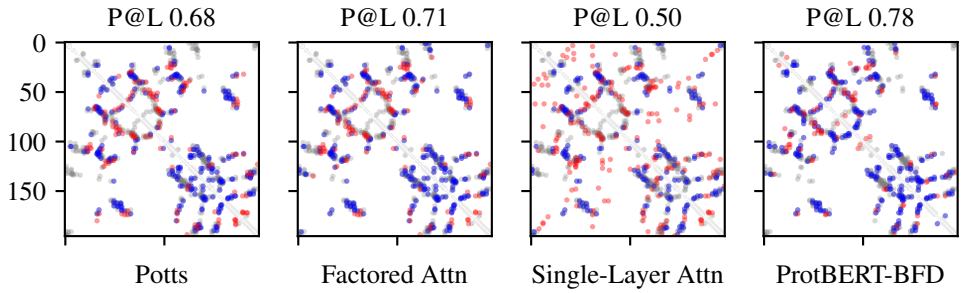


Figure 3.1: Predicted contact maps and Precision at L for each model on PDB entry *2BFW*. Blue indicates a true positive, red indicates a false positive, and grey indicates a false negative.

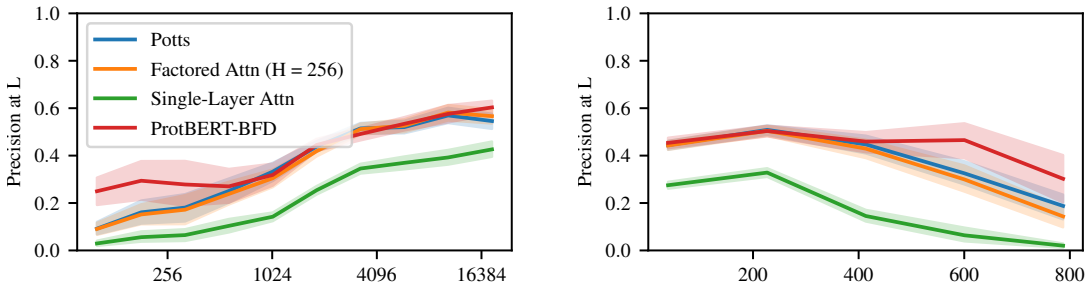
following the procedure outlined in Appendix B.6. We compare predicted contact maps $\widehat{C}$ to true contact maps C using standard metrics based on precision. A particularly important metric is *precision at* L, where L is the length of the sequence [103, 104]. This is computed by masking $\widehat{C}$ to only consider positions ≥ 6 apart, predicting the top L entries to be contacts, and computing precision. We provide more information on data and metrics in Appendix B.4 and on model hyperparameters in Appendix B.8.

Attention assumptions reflected in 15,051 protein structures. We examine all 15,051 structures in the dataset in Ref. [80] for evidence of two key properties useful for single-layer attention models: few contacts per residue and the number of contacts scaling linearly in length. In Appendix Figure B.2, we see that 80% of the 3,747,101 million residues in these structures have 4 or fewer contacts. Only 1.8% of residues have more than ten contacts. This shows that the row-wise softmax, which encourages each residue to attend to only a few other residues per-head, reflects structure found in the data.

Factored attention matches Potts performance on 748 families. Figure 3.1 shows a representative sample of good quality contact maps extracted from all models. Figure 3.2a summarizes the performance of all models over the set of 748protein families. Factored attention, Potts, and ProtBERT-BFD have comparable overall performance, with median precision at L of 0.46, 0.47, and 0.48, respectively. Stratifying by number of sequences reveals that ProtBERT-BFD has higher precision on MSAs with fewer than 256 sequences. For MSAs with greater than 1024 sequences, Potts, factored attention, and ProtBERT-BFD have comparable performance. Single-layer attention is uniformly worse over all MSA depths.

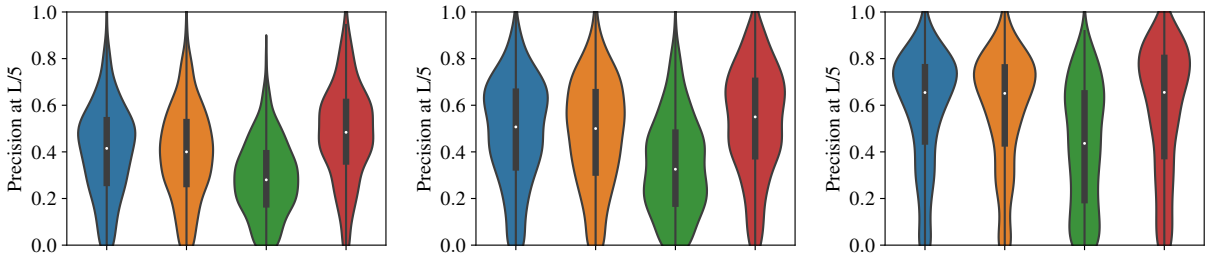
Next, we evaluate the impact of sequence length on performance. Figure 3.2b shows that factored attention and Potts achieve similar precision at L over the whole range of family lengths, despite factored attention having far fewer parameters for long families. This shows that factored attention can successfully leverage sparsity assumptions where they are most useful.

Long-range contacts are particularly important for downstream structure-prediction algorithms – long-range precision at $L/5$ is reported in both CASP12 and CASP13 [103, 104]. Figure 3.3 breaks down contact precisions based on position separation into short



(a) MSA depth (b) Reference length

Figure 3.2: Model performance evaluated on MSA depth and reference length. ProtBERT-BFD has higher precision on MSAs with fewer than 256 sequences. For larger MSAs, Potts, Factored Attention, and ProtBERT-BFD perform comparably. Across a variety of protein lengths, Factored Attention performs comparably to Potts with substantially fewer parameters.

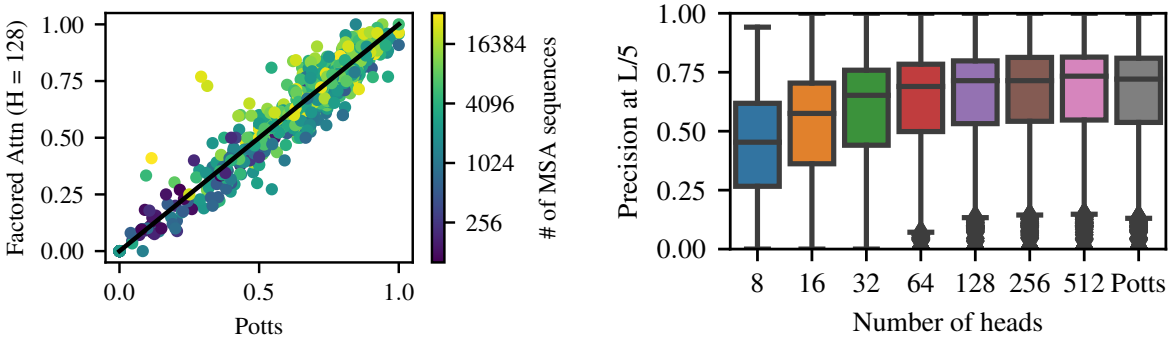


(a) Short-range (b) Medium-range (c) Long-range

Figure 3.3: Contact precision for all models stratified by the range of the interaction, with the same color correspondence as in Figure 3.2a. Potts, Factored Attention, and ProtBERT-BFD perform comparably for long and medium-range contacts, while ProtBERT-BFD has slightly better precision on short-range contacts.

$(6 \leq \text{sep} < 12)$, medium $(12 \leq \text{sep} < 24)$, and long $(24 \leq \text{sep})$. We see that ProtBERT-BFD performs best on short-range contacts, with a median increase of 0.068 precision at $L/5$. On long-range contacts, there is no appreciable difference in performance to Potts and factored attention. Across the range of contact bins, factored attention and Potts perform very similarly.

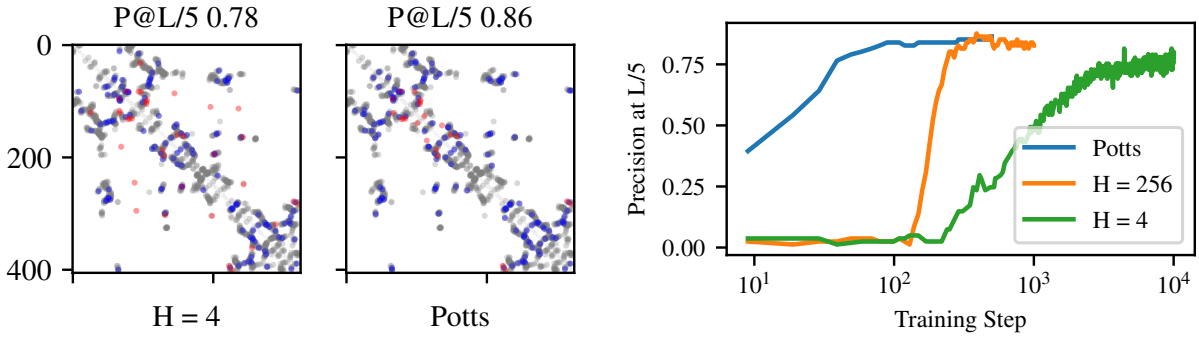
Fewer heads can match Potts on $L/5$ contacts. We probe the limits of parameter sharing by lowering the number of heads in factored attention and evaluating whether fewer heads can be used to precisely estimate contacts. Figure 3.4a shows that 128 heads can be used to estimate $L/5$ contacts as precisely as Potts over the full set of 748 families. In Figure 3.4b, we see that factored attention with 32 and 64 heads is still able to achieve reasonable



(a) 128 heads versus Potts P@$L/5$

(b) P@$L/5$ for varying number of heads.

Figure 3.4: Examining impact of number of heads on precision at $L/5$. Left: Comparing performance of Potts and 128 heads over each family shows comparable performance. Right: Precision at $L/5$ drops off slowly until 32 heads, then steeply declines beyond that.



(a) Top $L/5$ contacts on *3n2a*.

(b) Training dynamics of models on *3n2a*.

Figure 3.5: Factored attention with 4 heads can learn the top $L/5$ contacts on PDB *3n2a*.

overall performance compared to Potts. 32 and 64 heads have precision at $L/5$ at least as high as Potts for 329 and 348 families, respectively. If we wish to recover the top L contacts, 256 heads are required to match Potts across all families, as seen in Appendix Figure B.3. Having more heads than 256 does not further increase performance. Intriguingly, Appendix Figure B.4 demonstrates that both Spearman and Pearson correlation between the order-4 interaction tensors of factored attention and Potts improve even when increasing to 512 heads. We do not observe the same trends for increasing head size, as shown in Appendix Figure B.5

For some families, the number of heads can be reduced even further. We show an example on the MSA built for PDB entry *3n2a*. In Figure 3.5a, we see that merely 4 heads are required to recover $L/5$ contacts nearly identical to those recovered by Potts. This shows

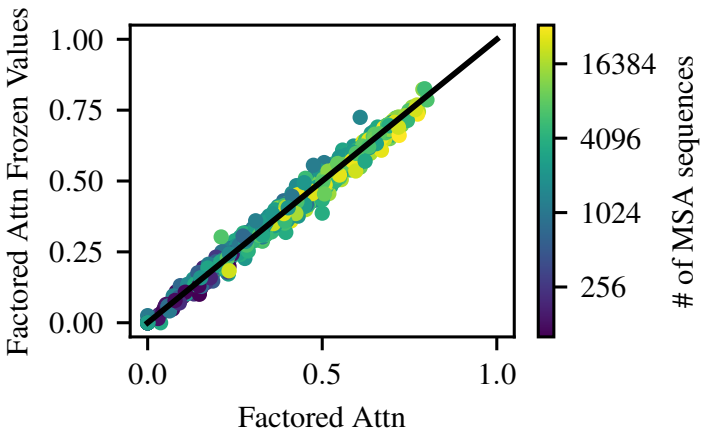


Figure 3.6: Precision at L comparison, which illustrates that a single set of frozen value matrices can be used for all families.

that shared amino acid features and interaction parameters can enable identical performance with a 300× reduction in parameters. The training dynamics of these models are shown in Figure 3.5b. Both factored attention with 256 heads and Potts converge after roughly 100 gradient steps, whereas factored attention with 4 heads requires nearly 10,000 steps to converge. In Appendix Figure B.6, we show that the top L contacts are significantly worse for 4 heads compared to Potts.

One set of amino acid features can be used for all families. Thus far we have only examined models that share parameters within single protein families. Since ProtBERT is trained on an entire database, it can leverage feature sharing across families to attain greater parameter efficiency and improved performance on small MSAs.

To explore the possibility that attention can share parameters across families, we train factored attention using a single set of frozen value matrices. We first train factored attention normally on *3n2a* with 256 heads, then freeze the learned value matrices for the remaining 747 families. The query and key parameters are trained normally. In Figure 3.6, we compare the precision at L of factored attention with frozen *3n2a* features to that of factored attention trained normally. Using a single frozen set of features results in only 6 families seeing precision at L decrease by more than 0.05, with a maximum drop of 0.11. This suggests that, even for a single-layer model, a single set of value matrices can capture amino acid features across functionally and structurally distinct protein families.

Factored attention reduces total parameters estimated. For an MSA of length L with alphabet size A, Potts models require $\binom{L}{2}A^2$ parameters. Factored attention with H heads and head size d requires $H(2Ld + A^2)$ parameters. In Figure B.7, we plot number of parameters versus length for various values of H and $d = 32$. Potts requires a total of 12 billion parameters to model all 748families. Factored attention with 256 heads and head size

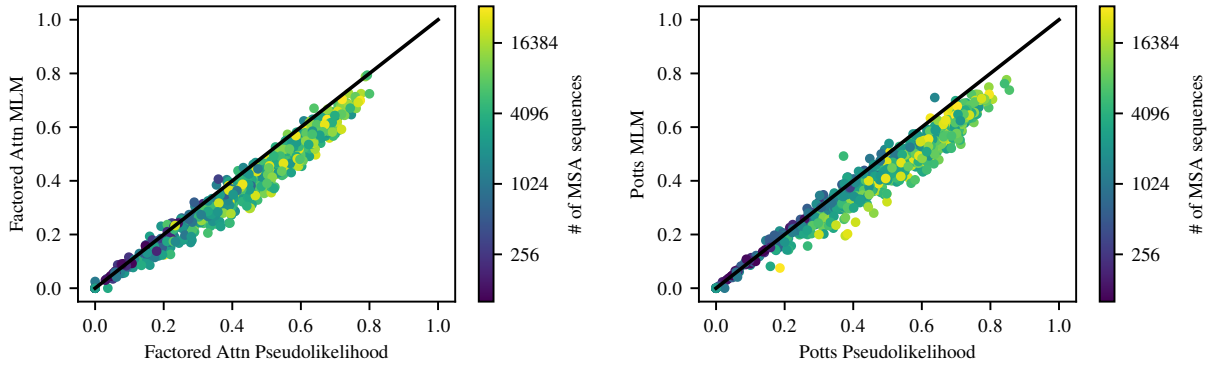


Figure 3.7: Effect of loss on precision at L over many families. Pseudolikelihood has a uniform but small benefit over masked language modeling for both models.

32 has 3.2 billion parameters; lowering to 128 heads reduces this to 790 million. Half of this reduction comes from 107 families of length greater than 400. ProtBERT-BFD is the most efficient, with 420 million parameters.

Impact of training loss function. The choice of loss function had a uniform but small impact for factored attention and Potts. As seen in Figure 3.7, pseudolikelihood training slightly improves contact accuracy over masked language modeling training.

Ablations. APC has a considerable impact on both Potts and factored attention, creating a median increase in precision at L of 0.1 and 0.07, respectively. The effect of APC is negligible for single-layer attention and ProtBERT. Addition of the single-site potential b_i increases performance slightly for attention layers, but not enough to change overall trends. To compare to ProtBERT-BFD, we train our single-layer attention models on unaligned families and found that performance degrades significantly. See Appendix Figures B.8-B.10.

3.5 Discussion

We have shown that single-layer factored attention models and the ProtBert-BFD Transformer achieve performance comparable to Potts models on unsupervised contact extraction. We have also shown that the assumptions encoded by attention reflect important properties of protein families. These results suggest that attention has a natural role in protein representation learning, without analogy to attention's success in the domain of NLP.

Our results also show that hierarchical signal within and across families can be captured by even simple attention models. The MSA Transformer [105] explicitly ties weights within families to achieve improved results on contact extraction, showing that modeling of hierarchical structure is beneficial for larger models trained on entire databases. There have been extensive efforts to organize the relationships between protein families and folds, most

notably the SCOP [106] and CATH [107] hierarchies. Further leveraging such rich structure will be essential to the development of powerful protein representations.

Chapter 4

Tuned Fitness Landscapes for Benchmarking Model Guided Protein Design

This chapter is based on collaborative work with Atish Agarwala, David Belanger, Lucy Colwell and Yun S. Song, and is available as a preprint on bioRxiv [108].

4.1 Introduction

Directed evolution (DE) [109, 110, 2] has revolutionized bioengineering, enabling the development of proteins with novel function across industries including food, chemicals, and therapeutics [110, 111, 112]. Part of the power of directed evolution is its simplicity: synthesize a set of variants, screen them for the desired function, induce mutagenesis in the top variants that passed the screen and repeat. This selective pressure pushes the variants towards the desired activity without requiring any prior knowledge of how mutations affect function. We can view DE as a genetic algorithm exploring a high-dimensional *landscape*, where higher points correspond to greater levels of desired activities. DE works under the assumption that functional landscapes are relatively smooth [2, 113], that is, combining high-performing mutations results in further improved variants.

With sufficient experimental throughput, DE can successfully climb smooth fitness landscapes. However, testing a set of variants for activity is a costly and time-consuming process, requiring specialized laboratory expertise. For example, experimental throughput to test the turnover rate of an enzyme can be on the order of tens of variants per round over a 10-round campaign [114]. Each inactive design comes at a great cost, not only because of the cost of the single negative result, but also comes at an opportunity cost as the inactive variant cannot be leveraged for designs in future rounds of DE. Thus, a practitioner wants to make efficient use of their experimental budget by carefully curating the designed variants. Guiding the designs can be done with *rational design*, which uses an understanding of the protein's structural and

functional properties to avoid poor designs. However, these properties are often unknown: an arbitrary wildtype starting sequence may not have any associated structure [37], or any characterization besides its homology to other sequences. To gain the understanding that would enable rational design would require experimental characterization which could be even more costly than the rest of the protein design campaign.

Effectively exploring rough (i.e., not smooth) landscapes with limited experimental throughput is a challenge for DE [115, 116, 117]. Thankfully, the confluence of advancements in DNA synthesis (bespoke oligonucleotide sequences), DNA sequencing (high throughput screens), and machine learning have enabled a new paradigm of machine learning guided directed evolution (MLDE) [118, 119, 120, 121] which can address the challenge of exploring rough landscapes with limited data. In each round of MLDE, a set of (variant, activity) pairs are collected, which a practitioner can use to train a genotype-phenotype model to predict the effect of variants. In the next round, that model can be used to propose candidates (see Figure 1 in [118]). MLDE has been successfully applied to designing proteins, such as AAV capsids with preferential delivery to specific organs in the body [122, 123], or increasing the fluorescence of GFP [124].

Despite early successes, there is no consensus on best practices for MLDE. There are many decisions involved in the design of an MLDE pipeline. What data should be collected? Which model class should be used? What optimization objective should be used for model training? How should models be selected? How should proposals be generated using a trained model? How does one trade off between "exploiting" model confidence and "exploring" regions of model uncertainty when synthesizing the next round of designs [125, 126]? Each choice interacts nonlinearly with every other choice, complicating the meta-optimization problem.

Running experiments is expensive, so it is essential to test the pipeline before using it in a new design campaign. One approach is to use *empirical landscapes* from publicly available experimental datasets. However, these datasets are limited by the prohibitive cost of producing enough high-quality data to benchmark an MLDE pipeline across multiple rounds. Datasets like [127, 128, 119, 123, 129] curate the highest quality protein function datasets from Deep Mutational Scanning [130], but focus on one- or two-mutation regions around a wildtype sequence [22]. Other landscapes contain all possible multi-mutants, but only cover a small portion of the overall protein (4 positions of the binding domain (B1) of protein G [131]), or test a limited allele vocabulary [132].

In parallel with deeply characterized empirical landscapes, *synthetic landscapes* are being explored as testing grounds for MLDE [133, 126]. Synthetic landscapes are defined by a software function that can be queried for any sequence of interest [134, 135, 133, 136, 125, 116, 137]. In this paper, we propose a specific synthetic landscape with two key properties:

- Properties **grounded** in the statistics of real protein families.
- **Tunable, interpretable** difficulty to match a range of plausible optimization landscapes.

To obtain these properties, we introduce and validate an MLDE benchmarking framework called SLIP: "Synthetic Landscape Inference for Proteins." SLIP is a set of synthetic fitness landscapes based on Potts models [138, 139, 15, 17, 18], combined with utilities for tuning the landscape difficulty. SLIP is open-source and is available at `https://github.com/google-research/slip`.

4.2 Background and Related Work

We define a *fitness landscape* $\mathcal{F}$ as a scalar-valued function over sequences $\mathbf{x}$ of length L with A alleles at each position. In practice, "fitness" is used to refer to a molecular phenotype (e.g., fluorescence) or organismal fitness (e.g., reproductive rate). Empirical landscapes measure the underlying fitness through a noisy observation process. Our synthetic fitness landscapes refer to the underlying, noiseless quantity.

We will consider a sequence design process which starts at "wildtype" $\mathbf{x}_0$, which has allele a_i at site i. The goal of the design process is to *maximize* the fitness (as opposed to the minimization of the loss function that occurs in traditional supervised learning). Therefore, we focus on the fitness *gain* $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0)$. A successfully designed sequence will, at a minimum, display $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0) > 0$.

Epistasis and Landscape Ruggedness

In order to test an MLDE pipeline, we need landscapes that cannot be effectively navigated without guidance from an ML model. Thus it is useful to tune the nonlinearity (or "difficulty") of a landscape. Two basic quantities we can use to understand the difficulty of task are 1) the *single-mutant effects* and 2) *pairwise epistasis*.

Single-mutant effects.

Letting $\mathbf{x}_{i\beta}$ denote the sequence obtained by mutating the wildtype allele at site i to allele β, the single mutant effect $s_{i\beta}$ is defined as:

$$s_{i\beta} = \mathcal{F}(\mathbf{x}_{i\beta}) - \mathcal{F}(\mathbf{x}_0). \tag{4.1}$$

Landscapes with many *adaptive* single mutations (site-allele pairs where $s_{i\beta} > 0$) tend to be easier to optimize.

Pairwise epistasis.

If the fitness of multi-mutants was given by the sum of single-mutant effects, then the landscape would be linear and easy to optimize – simply combining adaptive single mutants would lead to good sequences $\mathbf{x}$ with high values of $\mathcal{F}(\mathbf{x})$. Nonlinear interactions between mutations make optimization more difficult. These nonlinear effects are known as *epistasis*, a

pervasive property in empirical landscapes [115, 140, 141, 142, 143, 144]. For two mutations $a_i \to \beta$ at site i and $a_j \to \gamma$ at site j, we can define the *pairwise epistasis* $\epsilon_{i\beta,j\gamma}$ by

$$\epsilon_{i\beta,j\gamma} = \mathcal{F}(\mathbf{x}_{i\beta,j\gamma}) - \mathcal{F}(\mathbf{x}_0) - (s_{i\beta} + s_{j\gamma})\,. \tag{4.2}$$

In other words, pairwise epistasis is the part of the fitness difference between $\mathbf{x}_{i\beta,j\gamma}$ and $\mathbf{x}_0$ which cannot be explained by the single mutants. Deleterious epistatic interactions between adaptive single mutants (known as *reciprocal sign epistasis*) make it more difficult to combine good individual mutations to obtain good multi-mutants, confounding a linear model of the landscape.

Note that both pairwise epistasis and the single-mutant fitness differences must be defined relative to a reference sequence - in our case, the wildtype $\mathbf{x}_0$.

Synthetic Landscapes

Synthetic landscapes are designed so that $\mathcal{F}$ can be evaluated quickly for arbitrary sequences, even if computing all L^A values is prohibitive (in memory or time). Many of these landscapes were originally designed to study evolutionary processes [145, 116, 137].

More recently, synthetic landscapes have been applied to benchmark sequence design algorithms [133, 126, 136]. We can divide synthetic landscapes into four broad, overlapping classes: supervised neural models, biophysical models, random models, and graphical models, briefly described below.

Supervised neural models.

Supervised neural models are trained on experimental data to predict a regression output [146, 34, 147, 148]. Of particular note is the structural score given by AlphaFold2, which can be used as an optimization objective to find sequences likely to fold into a desired structure [149, 147]. If the model is sufficiently good, model outputs can be used as a synthetic replacement for the experimental target. Neural models are fast to evaluate with a single forward pass. However, they can exhibit pathological behavior when used as optimization objectives, giving high scores to unrealistic sequence [150, 151] or giving outsize influence to irrelevant parts of the sequence [152]. While trained neural models can exhibit high levels of ruggedness [153], it is not straightforward to tune the optimization difficulty of a neural landscape.

Biophysical models.

Biophysical models explicitly model the energetic interactions in the protein. Prominent examples of biophysical models are ViennaRNA [146] and Rosetta [154], which provide a score for an input sequence representing the free energy of the folded structure at equilibrium. These models return globally characterized landscapes without unexpected pathologies. However, they require a computationally expensive optimization procedure to report the free energy minimum for each query sequence. Since physical assumptions like physical constants and

potential energy functions are baked into the model, there is no principled tuning procedure to make the landscapes more difficult to optimize.

Random models.

Random models generate a landscape by drawing a random function on the configuration space $\{1, \ldots, A\}^L$ with a particular distribution. The NK-model explicitly models order-K interactions for a sequence of length N to provide tunably rugged fitness landscapes which exhibit high-order correlations [134, 135]. The generalized NK-model, which explicitly models sparse blocks of interacting positions, has been shown to reflect the sparsity of empirical fitness function when conditioned on real structures [155]. Distance-dependent models [117] have interactions at all orders, and are defined by fixing a functional form for the covariance between sequences as a function of genetic distance. All of these models have been applied to study evolutionary dynamics, and are typically not fit to data.

Graphical models.

Graphical models explicitly represent the interactions between positions in the sequence as edges in a graph. Profile HMMs do not model epistasis, and only model first-order interactions (i.e., amino acid distributions at aligned positions). Despite this, they serve as powerful protein family classifiers [13], and HMM likelihoods have been used as synthetic optimization objectives [133]. Profile HMMs can also flexibly handle insertions and deletions, and do not require aligned sequences as input. We describe in detail below a graphical model known as a Potts model.

Potts Models of Protein Families

Definition.

For a family of proteins of length L with A possible alleles at each position, a Potts model defines a probability distribution over sequences in the family as

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(\mathcal{F}(\mathbf{x})\right), \tag{4.3}$$

where $\mathcal{F}$ is a negated statistical energy, and the partition function Z is a normalization constant such that the $p(\mathbf{x})$ sum to 1. In a Potts model, for an input one-hot encoded sequence $\mathbf{x} \in \mathbb{R}^{L \times A}$, $\mathcal{F}$ is given by the sum over the marginal effects and pairwise interactions:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{L} \sum_{\alpha=1}^{A} h_{i\alpha} x_{i\alpha} + \frac{1}{2} \sum_{i,j=1}^{L} \sum_{\alpha,\beta=1}^{A} H_{i\alpha,j\beta} x_{i\alpha} x_{j\beta} \tag{4.4}$$

where $\mathbf{h} = (h_{i\alpha})$ is a tensor of dimension $L \times A$ representing marginal terms and $\mathbf{H} = (H_{i\alpha,j\beta})$ is a symmetric tensor of dimension $L \times L \times A \times A$ representing pairwise coupling terms. The

parameters of a Potts model can be fit using a set of aligned sequences; see Appendix C.1 for details.

Modeling coevolution.

There has been extensive work establishing that Potts models learn statistics grounded in protein structure and function. Potts models are useful as unsupervised structure predictors [15, 16, 17, 18], and are competitive with neural unsupervised structure predictors [156] on families with large, diverse alignments. The statistical energy of protein variants scored by a Potts model has been shown to correlate well with empirical fitness [21]. When the statistical energy is included as an additional feature to a regression model, it has been shown to improve predictive performance on empirical landscapes [22]. Used as generative models, Potts models have been shown to propose functional variants of a given protein target [20]. Synthetic sequences evolved *in silico* on a Potts landscape have been shown to correlate with summary statistics with *in vitro* evolved sequences [157]. The parameters of the Potts model can also be used as input featurizations that improve performance on downstream tasks [158, 159].

4.3 Methods: Tuned Quadratic Landscapes

We can use the statistical energy of the Potts model as the fitness function $\mathcal{F}$ to define a synthetic landscape. In [133], for example, the authors introduced a "PDB-Ising" synthetic fitness landscape, which combined the contact map of a protein with standard pair potentials for amino acid substitution to create a simple *quadratic landscape* with pairwise interactions. We will instead derive model parameters from alignment data, which has the advantage that the resulting synthetic landscape exhibits correlations grounded in the coevolutionary couplings in that family.

While useful in their own right, Potts models derived from alignment data are not challenging synthetic landscapes, as they can be optimized by combining top mutations one at a time (akin to an *in silico* DE algorithm). We quantify these shortcomings in Section 4.5. To benchmark the performance of MLDE pipelines on more difficult optimization scenarios, we require synthetic landscapes where strategies guided by nonlinear models can substantially outperform those guided by linear models. In what follows, we develop a framework for tuning quadratic landscapes to a desired level of difficulty.

Tuning Model Statistics

For a fitness function defined by a Potts model, the single-mutant and pairwise epistasis terms can be written explicitly in terms of the model parameters $\mathbf{h}$ and $\mathbf{H}$:

$$s_{i\beta} = \mathbf{h}_{i\beta} - \mathbf{h}_{ia_i} + \sum_{j=1}^{L} \left(\mathbf{H}_{i\beta,ja_j} - \mathbf{H}_{ia_i,ja_j} \right), \tag{4.5}$$

$$\epsilon_{i\beta,j\gamma} = \mathbf{H}_{i\beta,j\gamma} - \mathbf{H}_{i\beta,ja_j} - \mathbf{H}_{ia_i,j\gamma} + \mathbf{H}_{ia_i,ja_j}, \tag{4.6}$$

where again a_i is the allele of the wildtype $\mathbf{x}_0$ at site i. Note that single-mutant effect $s_{i\beta}$ depends on both the linear and quadratic parameters of the Potts model. See Appendix C.2 for a detailed derivation.

Once we reparameterize the Potts model in terms of the single-mutant and pairwise epistasis terms, the fitness decomposes into the form

$$\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0) = \sum_{(i,\beta)\in M} s_{i\beta} + \frac{1}{2} \sum_{(i,\beta),(j,\gamma)\in M:(i,\beta)\neq(j,\gamma)} \epsilon_{i\beta,j\gamma}, \tag{4.7}$$

where M is the set of mutations in $\mathbf{x}$ encoded as site-allele pairs (i, β).

Introducing shift (μ_s, μ_ϵ) and scale $(\lambda_s, \lambda_\epsilon)$ parameters for single-mutant and epistatic terms, we can parameterize a family of tuned fitness functions $\tilde{F}$ with parameters $\tilde{s}$ and $\tilde{\epsilon}$ given by

$$\tilde{s}_{i\beta} = \lambda_s(s_{i\beta} + \mu_s) \quad \text{and} \quad \tilde{\epsilon}_{i\beta,j\gamma} = \lambda_\epsilon(\epsilon_{i\beta,j\gamma} + \mu_\epsilon). \tag{4.8}$$

The four parameters $(\mu_s, \mu_\epsilon, \lambda_s, \lambda_\epsilon)$ allow for the mean and variance of both $\tilde{s}$ and $\tilde{\epsilon}$, taken over position-allele pairs, to be independently tuned. This allows us the flexibility of changing the difficulty of the landscape (e.g., by making epistasis more negative on average) while maintaining much of the structure of the original problem (e.g., preserving the coevolutionary couplings).

Epistatic Horizon

For untuned landscapes, the single-mutant fitness effects approximate the double-mutant fitness effects well (Figure 4.1, blue), meaning the landscape is very linear. If combining random adaptive single mutants (mutations where $s_{i\beta} > 0$) is not a viable strategy, a naïve design strategy will struggle. For example, in Figure 4.1, points with $s_{i\beta} + s_{j\gamma} > 0$ but $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0) < 0$ (bottom right, in orange) would be candidate proposals by a naïve algorithm that would fail an experimental screen on a rugged landscape. One way to quantify the linearity (or non-linearity) of the landscape is to define an "epistatic horizon" K_{epi} – the number of mutations after which the linear approximation breaks down. With an eye towards tuning optimization difficulty, we define K_{epi} as follows.

Let $\bar{s}_+$ be the average of $s_{i\beta}$ over adaptive singles, and $\bar{\epsilon}_{+,+}$ the mean epistatic effect over random adaptive pairs. Then, the sequence design problem is difficult if the average

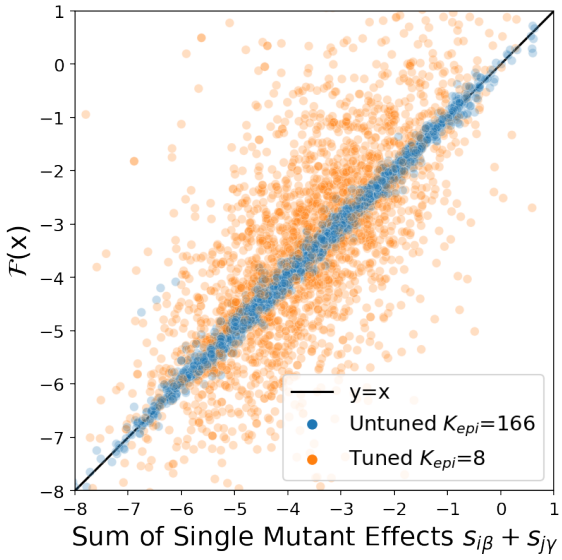


Figure 4.1: Tuning the epistatic horizon increases the ruggedness of the resulting landscape. Fitness $\mathcal{F}(\mathbf{x})$ of 5000 variants with 2 mutations (centered so that $\mathcal{F}(\mathbf{x}_0) = 0$). The untuned fitness landscape with epistatic horizon $K_{\text{epi}} = 166$ (blue) is roughly linear, while the tuned fitness landscape with epistatic horizon $K_{\text{epi}} = 8$ (orange) exhibits more ruggedness. This landscape is derived from the alignment for PDB id 3er7. Note that the untuned epistatic horizon is greater than the length of the protein, $K_{\text{epi}} = 166 > 118 = L$.

interaction between individually good mutations is negative: $\bar{\epsilon}_{+,+} < 0$. Taking K random adaptive mutations, the average change in fitness for a K-mutant $\mathbf{x}_K$ is

$$\mathbb{E}_{\mathbf{x}_K}\left[\mathcal{F}(\mathbf{x}_K) - \mathcal{F}(\mathbf{x}_0)\right] = K\bar{s}_+ + \binom{K}{2}\bar{\epsilon}_{+,+}. \tag{4.9}$$

As K increases, the relative effect of epistasis grows relative to the single-mutant effects. We can compute a crossover value when $\mathbb{E}_{\mathbf{x}_K}[\mathcal{F}(\mathbf{x}_K) - \mathcal{F}(\mathbf{x}_0)] = 0$. Motivated by this example, we define the *epistatic horizon* K_{epi} as the non-zero solution to the equation:

$$K_{\text{epi}}\bar{s}_+ + \frac{K_{\text{epi}}(K_{\text{epi}} - 1)}{2}\bar{\epsilon}_{+,+} = 0. \tag{4.10}$$

This definition suggests two ways in which synthetic landscapes can fail to be difficult for MLDE:

- $K_{\text{epi}} > L$. In this case, the landscape is relatively linear at all relevant length scales. Combining adaptive single mutants leads to adaptive multi-mutants even for a large number of mutations.

- $K_{\text{epi}} < 0$. In this case, $\bar{\epsilon}_{+,+}$ is positive; that is, on average, adaptive single-mutants combine to be *better* than the sum of their parts. In this case, combining adaptive single-mutants also leads to adaptive multi-mutants, since typical pairs will interact positively with each other.

All the untuned landscapes we studied have $K_{\text{epi}} > L$, and many also have $K_{\text{epi}} < 0$; see Figure 4.1 and Appendix Table C.1. This means that the untuned landscapes are unsuitable for testing MLDE pipelines as is.

However, we can use tuning parameters to adjust K_{epi} and generate landscapes which are more non-linear and harder to optimize (Figure 4.1, orange). In the next section, we derive a tuning procedure which adjusts K_{epi} while leaving many other statistics of the fitness landscape fixed, thereby allowing us to benchmark MLDE pipelines.

Tuning Procedure

With four free parameters (two shift and two scale parameters), we can introduce additional constraints on the fitness landscape. First, we normalize the landscape such that the single-mutant effects have unit variance by setting $\lambda_s = \sigma(s)^{-1}$, where $\sigma(s)$ is the standard deviation of the $\{s_{i\beta}\}$. Second, we preserve the fraction of adaptive single mutants $\alpha_{s_+} \equiv \#\{s_{i\beta} > 0\}/L(A-1)$ by setting $\mu_s = 0$. Finally, to preserve the relative magnitudes (i.e., the ratios) of the pairwise epistasis terms, we set $\mu_\epsilon = 0$. This leaves λ_ϵ free. By fixing a target K_{epi}, we can use Equation (4.10) to solve for λ_ϵ.

To summarize: given a target epistatic horizon $K_{\text{epi}} > 0$, the tuning parameters are given by equations:

$$\mu_s = \mu_\epsilon = 0, \tag{4.11}$$

$$\lambda_s = \sigma(s)^{-1}, \tag{4.12}$$

and

$$K_{\text{epi}}\lambda_s\bar{s}_+ + \frac{K_{\text{epi}}(K_{\text{epi}}-1)}{2}\lambda_\epsilon\bar{\epsilon}_{+,+} = 0, \tag{4.13}$$

where $\bar{s}_+$ and $\bar{\epsilon}_{+,+}$ are the values for the untuned landscape. Note that other tunings are possible for a given K_{epi}. This specific tuning scheme was chosen to preserve as much of the co-evolutionary structure of the Potts model as possible.

4.4 Methods: *In silico* Validation

To validate that our tuned synthetic landscapes are sufficiently difficult, we aim to create landscapes where nonlinear models outperform linear (naïve) models on sequence design tasks. To do so, we design an experimental framework which evaluates how effective linear and nonlinear models are at using training data to accurately rank design candidates.

We use the following procedure for each untuned landscape; see Figure 4.2 for a schematic:

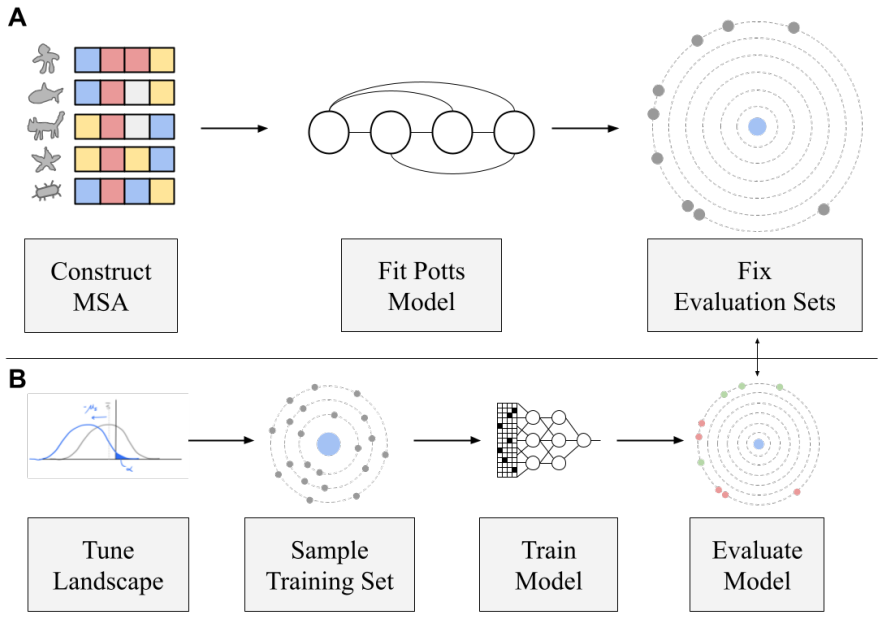


Figure 4.2: *In silico* validation workflow. Panel A shows the tasks that are performed once for each PDB ID. After training a Potts model on an aligned set of sequences, we derive a set of evaluation sequences. Panel B shows the tasks that are performed for each replicate of the regression experiment: we tune the landscape, sample a set $(x, y) \in D$ of training sequences x with their associated synthetic fitnesses y, train a model on D, and then evaluate the model predictions on the evaluation sets. The evaluation sets are fixed for all landscapes derived from the same untuned Potts initialization.

1. Tune the epistatic horizon to $K_{\text{epi}} = 2^{\ell}$ for $\ell \in \{1, 2, \ldots, 10\}$ as in Section 4.3. Center at $\mathcal{F}(\mathbf{x}_0) = 0$.

2. Sample a dataset $D = \{(\mathbf{x}, y)\}$ of sequences $\mathbf{x}$ with their associated fitness y, where $|D| = 5000$. Each sample $(\mathbf{x}, y)$ is obtained by sampling a number of mutations from the wildtype $\mathbf{x}_0$ uniformly at random from $\{1, 2, 3\}$ and then sampling a variant uniformly at random at the selected distance.

3. Train Ridge and convolutional neural network (CNN) regression models across many different hyperparameter choices.

4. For the best performing model of each type, compute a paired performance metric on the evaluation set.

Untuned Landscapes

To select a suitable set of synthetic landscapes, we first initialize $\mathbf{h}$ and $\mathbf{H}$ by training Potts models on alignments of protein sequences. We choose protein targets to span a range of functions, structural folds, and primary sequence lengths, while ensuring the resulting Potts model has excellent contact accuracy on a high resolution structure. From the 748 Potts models trained in [74], the models corresponding to the 5 PDB IDs in Figure 4.3 were selected manually from the top performing models with respect to contact prediction accuracy. See Appendix Figure C.1 for predicted contact maps. We set the wildtype sequence $\mathbf{x}_0$ to be the alignment query sequence.

Evaluation Sets

We choose an evaluation set relevant for the objective of sequence design. Combining top single mutations is a common strategy for proposing variants [112], and properly ranking these proposals is directly relevant to the objective of MLDE. For each (untuned) landscape, we construct evaluation sets at mutation distance 6 from the wildtype by taking the top 20 single mutants, combining them to construct variants at the desired distance, and then taking a random subset of the desired size (200). Because the set of top 20 single mutants does not change in response to tuning (i.e., single-mutant rankings are preserved by tuning), the evaluation set for a given PDB ID is fixed to the same set of 200 sequences (note that their fitness $\tilde{\mathcal{F}}(\mathbf{x})$ changes with tuning). See Appendix C.1 for a discussion of other evaluation sets.

Models

Ridge regression.

Our baseline linear model uses the `sklearn` implementation of Ridge regression, which has a single hyperparameter representing the L_2 penalty. The grid of hyperparameters used during model selection is reported in Appendix Table C.3. This is a strong baseline especially for landscapes where the level of epistasis is low. In addition, Hsu *et al.* [22] showed that the ridge penalty induces a powerful inductive bias that generalizes to unseen mutations by setting the effect of unseen mutations to the average effect seen at the same position. We remove the intercept term, since centering ensures $\tilde{\mathcal{F}}(\mathbf{x}_0) = 0$.

Convolutional Neural Network.

Convolutional neural networks (CNNs) have been used to great success in protein sequence modeling [123, 160, 161], so we select them as our nonlinear model class. The CNN model architecture is 3 layers of 1D convolutions, followed by a dense layer. On 3er7 ($L = 118$), a CNN architecture with 32 filters, kernel size 5, and hidden size 64 results in a model with 255,329 parameters. The CNN is trained using an Adam optimizer [162] to minimize

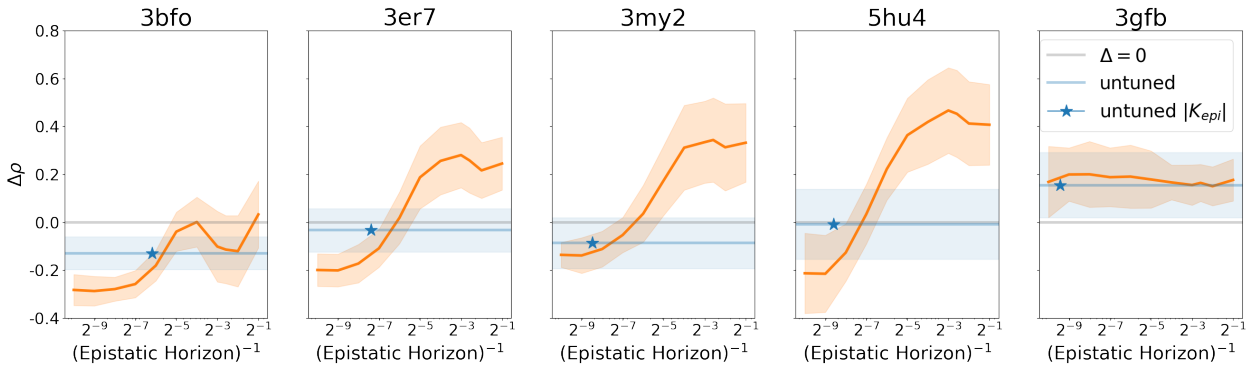


Figure 4.3: Tuning the epistatic horizon interpolates between linear and non-linear model performance on ranking combinations of 6 adaptive singles after training on 5000 examples. The grey line shows $\Delta\rho = 0$ for ease of visualization of the threshold for one model outperforming the other. The orange line shows the difference between maximum CNN Spearman's ρ and maximum Ridge Spearman's ρ, with error bands showing ± 1 standard deviation across 20 random training set replicates. The x-axis corresponds to inverse epistatic horizon, so that more linear landscapes are to the left and more non-linear landscapes are to the right. The blue band shows model performance on the untuned landscape. The blue star shows the position on the x-axis which corresponds to the magnitude of the untuned epistatic horizon $|K_{\text{epi}}|$.

MSE loss. We tune the learning rate, number of filters, batch size, and number of training epochs. See Appendix Table C.4 for tuned hyperparameters and Appendix Table C.5 for fixed hyperparameters.

Paired Performance Metrics

An important driver of variability in evaluation performance are the training and evaluation sets. By keeping these fixed while allowing the model class to vary, we can isolate performance differences due to modeling capacity. For each sampled training set, we select the best Ridge model and the best CNN model in terms of ranking the given evaluation set. We then compare the computed performance metric on the given evaluation set and take the difference. "Differential Spearman ρ" (or $\Delta\rho$) refers to the difference, given a fixed training set, between the maximum CNN Spearman ρ and the maximum Ridge Spearman ρ on the evaluation set.

4.5 Results

Untuned Landscapes are Linear

In Figure 4.1, we plot the variant fitness as a function of the sum of constituent single-mutant effects. The untuned landscape fitness (in blue) follows a linear trend, where the fitness of a double-mutant can be well predicted by the sum of constituent single-mutant effects. Compared to the tuned landscape, the untuned landscape exhibits much less ruggedness. In Figure 4.3, the blue line corresponds to the differential performance of the CNN model compare to the Ridge model on the untuned landscape. Across all untuned landscapes except 3gfb, the CNN models have a mean performance change $\Delta\rho < 0$, i.e., that the CNN does not significantly boost performance on ranking combinations of adaptive singles compared to the Ridge model. This shows that untuned Potts models do not provide landscapes useful for benchmarking sequence design guided by nonlinear models, motivating the development of our tuning framework.

Epistatic Horizon Tunes the Nonlinearity of the Landscape

We aim to validate that our tuning framework can create fitness landscapes difficult enough to require nonlinear models. In Figure 4.3, for four of the five PDB IDs, as the epistatic horizon increases (to the left in the figure), evaluation set performance skews in favor of the Ridge model, confirming that as the landscape is tuned to be more linear, linear models are preferred to nonlinear models. Conversely, as the epistatic horizon decreases (to the right in the figure) and the landscape becomes dominated by nonlinear effects, evaluation set performance shifts to favor the CNN model. The intermediate-length proteins (3er7, 3my2, 5hu4 – the middle three panels of Figure 4.3), show consistent behavior, indicating that the epistatic horizon is a generalizable metric of landscape difficulty.

On the intermediate-length proteins 3er7, 3my2, and 5hu4, Spearman's ρ improves for the nonlinear model by between 0.2 and 0.4. For example, 3er7 shows an improvement in evaluation set performance from 0.1 to 0.3 (Appendix Figure C.2). For comparison, the authors in [118] found that zero-shot predictors on the 4-position GB1 landscape with a Spearman's ρ of 0.2 are sufficient to substantially improve sequence design. Across all landscapes, all models get worse with increased tuning, indicating that decreasing the epistatic horizon increases the landscape difficulty for nonlinear as well as linear models (see Appendix Figure C.2 for unpaired model performance). For horizons $K_{\text{epi}} \gg L$, the Ridge model achieves near perfect ranking accuracy $\rho \approx 1$ (see Appendix Figure C.2).

On 3gfb (far right in Figure 4.3), differential model performance remains roughly constant around $\Delta\rho = 0.2$ across all tested tunings. On 3bfo (the first panel of Figure 4.3), the shortest protein, differential model performance favors linear models for increased epistatic horizons, but does not achieve a regime where nonlinear models significantly outperform linear models. This may be due to CNN models overfitting on the short protein.

4.6 Discussion and Future Work

Using nonlinear models to guide sequence design has made a large impact in practice [120, 119, 123, 163], but many questions still remain in regard to how to use these models as part of an MLDE pipeline. Our experimental results validate that our quadratic landscape tuning framework can generate synthetic landscapes which require nonlinear models for effective optimization. By deriving our landscapes from Potts models trained on real alignments, we ground the properties of our synthetic landscape in the structural and biochemical features of real proteins. These two properties enable tuned quadratic landscapes to be used to benchmark machine learning-guided protein design.

Our landscape tuning procedure relied on an optimization-motivated definition of the epistatic horizon K_{epi}. There are other scenarios where a more general definition for K_{epi} may be more relevant; for example, in a regression setting, the relevant crossover may be the point at which the *variance* in nonlinear fitness components is more than the variance of the linear components. Additionally, we focused on a tuning which increased the difficulty of combining adaptive mutants; there are other forms of nonlinearity which make ML-aided design more useful. One such situation is finding individually non-adaptive single mutants which combine to make adaptive mutants. Our landscape tuning framework is flexible enough to allow tuning of these types of properties.

Another frontier of ML for biological sequence design is making efficient use of labeled data with protein-specific priors provided, for example, by large language models [34, 81, 28]. There is room for model development to incorporate priors that allow sequence models such as CNNs to learn more nonlinear landscapes. By providing realistic datasets for model training, tuned quadratic landscapes are a useful sandbox environment for proposing modeling advancements that can take advantage of small datasets.

Optimizing MLDE pipelines involves more than tuning a neural network architecture. In MLDE, design choices ranging from training set curation to sequence proposal distributions can have a huge impact on the overall effectiveness of the pipeline. Benchmarking these choices against tuned quadratic landscapes would allow practitioners of MLDE to understand how to optimize their pipeline before having to collect expensive experimental data. Often, a new protein design campaign will have very specific constraints, such as assay-specific noise, or limitations on experimental throughput. Our tuned quadratic landscapes lend themselves easily to multiple extensions that allow an MLDE practitioner to impose these specific constraints and see how their pipeline performs. For a new design campaign with a sequence alignment, a bespoke synthetic landscape can be derived directly to match the structural constraints of the target. In addition, MLDE design choices can be correlated with landscape difficulty by testing across a range of tuning parameters. We hope that the benchmarks enabled by SLIP will further support the development of robust and efficient methods for biological sequence design.

Bibliography

[1] Jorge Luis Borges. *Ficciones*. New York: Grove Press, 1963.

[2] Frances H Arnold. *Nobel Lecture: Innovation by Evolution: Bringing New Chemistry to Life*. en. https://www.nobelprize.org/prizes/chemistry/2018/arnold/lecture/. Accessed: 2022-10-8. Dec. 2018.

[3] Jun-Jie Liu et al. "CasX enzymes comprise a distinct family of RNA-guided genome editors". In: *Nature* 566.7743 (2019), p. 218.

[4] Hongyuan Lu et al. "Machine learning-aided engineering of hydrolases for PET depolymerization". In: *Nature* 604.7907 (Apr. 2022), pp. 662–667.

[5] Brian Hie et al. "Learning the language of viral evolution and escape". In: *Science* 371.6526 (Jan. 2021), pp. 284–288.

[6] F Jacob. "Evolution and tinkering". In: *Science* 196.4295 (June 1977), pp. 1161–1166.

[7] Naomi K Fox, Steven E Brenner, and John-Marc Chandonia. "SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures". In: *Nucleic Acids Research* 42.D1 (2013), pp. D304–D309.

[8] The UniProt Consortium. "UniProt: a worldwide hub of protein knowledge". In: *Nucleic Acids Research* 47.D1 (2018), pp. D506–D515. ISSN: 0305-1048. DOI: 10.1093/nar/gky1049. URL: https://doi.org/10.1093/nar/gky1049.

[9] Martin Steinegger and Johannes Söding. "Clustering huge protein sequence sets in linear time". In: *Nature Communications* 9.1 (2018), pp. 1–8.

[10] Christopher T Brown et al. "Unusual biology across a group comprising more than 15% of domain Bacteria". In: *Nature* 523.7559 (July 2015), pp. 208–211.

[11] Stephen K Burley et al. "RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy". In: *Nucleic Acids Research* 47 (2019). DOI: 10.1093/nar/gky1004. URL: https://academic.oup.com/nar/article-abstract/47/D1/D464/5144139.

[12] Stephen F. Altschul et al. "Basic local alignment search tool". In: *Journal of Molecular Biology* 215.3 (1990), pp. 403–410. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2. URL: https://www.sciencedirect.com/science/article/pii/S0022283605803602?via%7B%5C%%7D3Dihub.

[13] Sean R Eddy. "Accelerated Profile HMM Searches". en. In: *PLoS Comput. Biol.* 7.10 (Oct. 2011), e1002195.

[14] Richard Durbin et al. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[15] Debora S Marks et al. "Protein 3D structure computed from evolutionary sequence variation". In: *PLoS One* 6.12 (Dec. 2011), e28766.

[16] Sivaraman Balakrishnan et al. "Learning generative models for protein fold families". In: *Proteins* 79.4 (Apr. 2011), pp. 1061–1078.

[17] Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. "Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era". In: *Proc. Natl. Acad. Sci. U. S. A.* 110.39 (Sept. 2013), pp. 15674–15679.

[18] Sergey Ovchinnikov et al. "Large-scale determination of previously unsolved protein structures using evolutionary information". In: *Elife* 4 (Sept. 2015), e09248.

[19] Matteo Figliuzzi, Pierre Barrat-Charlaix, and Martin Weigt. "How Pairwise Coevolutionary Models Capture the Collective Residue Variability in Proteins?" In: *Mol. Biol. Evol.* 35.4 (Apr. 2018), pp. 1018–1027.

[20] William P Russ et al. "An evolution-based model for designing chorismate mutase enzymes". In: *Science* 369.6502 (July 2020), pp. 440–445.

[21] Thomas A Hopf et al. "Mutation effects predicted from sequence co-variation". In: *Nat. Biotechnol.* 35.2 (Feb. 2017), pp. 128–135.

[22] Chloe Hsu et al. "Learning protein fitness models from evolutionary and assay-labeled data". In: *Nat. Biotechnol.* (Jan. 2022).

[23] Joerg Schaarschmidt et al. "Assessment of contact predictions in CASP12: Co-evolution and deep learning coming of age". In: *Proteins: Structure, Function, and Bioinformatics* 86 (2018), pp. 51–66.

[24] Andrew W. Senior et al. "Improved protein structure prediction using potentials from deep learning". In: *Nature* 577.7792 (2020), pp. 706–710. ISSN: 14764687. DOI: 10.1038/s41586-019-1923-7. URL: https://doi.org/10.1038/s41586-019-1923-7.

[25] Tom B Brown et al. "Language Models are Few-Shot Learners". In: (May 2020). arXiv: 2005.14165 [cs.CL].

[26] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv* (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[27] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[28] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *Proc. Natl. Acad. Sci. U. S. A.* 118.15 (Apr. 2021).

[29] Joshua Meier et al. "Language models enable zero-shot prediction of the effects of mutations on protein function". In: *Adv. Neural Inf. Process. Syst.* 34 (2021).

[30] Zeming Lin et al. "Evolutionary-scale prediction of atomic level protein structure with a language model". In: *bioRxiv* (Oct. 2022), p. 2022.07.20.500902.

[31] A Madani et al. "ProGen: Language Modeling for Protein Generation". In: *arXiv* (2020).

[32] Jesse Vig et al. "{BERT}ology Meets Biology: Interpreting Attention in Protein Language Models". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YWtLZvLmud7.

[33] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[34] Roshan Rao et al. "Evaluating Protein Transfer Learning with TAPE". In: *Adv. Neural Inf. Process. Syst.* 32 (Dec. 2019), pp. 9689–9701.

[35] Matthew Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://www.aclweb.org/anthology/N18-1202.

[36] Timothy A Whitehead et al. "Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing". In: *Nature Biotechnology* 30.6 (2012), p. 543.

[37] Nelson Perdigão et al. "Unexpected features of the dark proteome". In: *Proceedings of the National Academy of Sciences* 112.52 (2015), pp. 15898–15903. ISSN: 0027-8424. DOI: 10.1073/pnas.1508380112. URL: https://www.pnas.org/content/112/52/15898.

[38] Tristan Bepler and Bonnie Berger. "Learning protein sequence embeddings using information from structure". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=SygLehCqtm.

[39] Ethan C Alley et al. "Unified rational protein engineering with sequence-based deep representation learning". In: *Nat. Methods* 16.12 (Dec. 2019), pp. 1315–1322. URL: http://dx.doi.org/10.1038/s41592-019-0598-1.

[40] Stephen F Altschul et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". In: *Nucleic Acids Research* 25.17 (1997), pp. 3389–3402.

[41] Michael Remmert et al. "HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment". In: *Nature Methods* 9.2 (2012), pp. 173–175. ISSN: 1548-7091. DOI: 10.1038/nmeth.1818. URL: http://www.nature.com/articles/nmeth.1818.

[42] Sean R. Eddy. "Profile hidden Markov models." In: *Bioinformatics (Oxford, England)* 14.9 (1998), pp. 755–763.

[43] "IUPAC-IUB Joint Commission on Biochemical Nomenclature (JCBN). Nomenclature and symbolism for amino acids and peptides. Recommendations 1983". In: *Eur. J. Biochem.* 138.1 (Jan. 1984), pp. 9–37. URL: http://dx.doi.org/10.1111/j.1432-1033.1984.tb07877.x.

[44] C B Anfinsen et al. "The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain." In: *Proceedings of the National Academy of Sciences of the United States of America* 47.9 (1961), pp. 1309–14. ISSN: 0027-8424. DOI: 10.1073/pnas.47.9.1309. URL: http://www.ncbi.nlm.nih.gov/pubmed/13683522%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC223141.

[45] Burkhard Rost. "Twilight zone of protein sequence alignments". In: *Protein Engineering, Design and Selection* 12.2 (1999), pp. 85–94. ISSN: 1741-0134. DOI: 10.1093/protein/12.2.85. URL: https://academic.oup.com/peds/article-lookup/doi/10.1093/protein/12.2.85.

[46] Steven E Brenner, Cyrus Chothia, and Tim JP Hubbard. "Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships". In: *Proceedings of the National Academy of Sciences* 95.11 (1998), pp. 6073–6078.

[47] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[48] John Moult et al. "Critical assessment of methods of protein structure prediction (CASP)-Round XII". In: *Proteins: Structure, Function, and Bioinformatics* 86 (2018), pp. 7–15. ISSN: 08873585. DOI: 10.1002/prot.25415. URL: http://doi.wiley.com/10.1002/prot.25415.

[49] Mohammed AlQuraishi. "ProteinNet: a standardized data set for machine learning of protein structure". In: *bioRxiv* (2019). arXiv: 1902.00249. URL: http://arxiv.org/abs/1902.00249.

[50] Jason Weston et al. "Semi-supervised protein classification using cluster kernels". In: *Advances in Advances in Neural Information Processing Systems*. 2004, pp. 595–602.

[51] Hyunjung Shin and Koji Tsuda. "Prediction of Protein Function from Networks". In: *Semi-Supervised Learning*. 2006.

[52] Ehsaneddin Asgari and Mohammad R K Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics". In: *PLoS One* 10.11 (Nov. 2015), e0141287.

[53] Michael Heinzinger et al. "Modeling the Language of Life - Deep Learning Protein Sequences". In: *bioRxiv* (2019). DOI: 10.1101/614313. URL: https://www.biorxiv.org/content/biorxiv/early/2019/04/19/614313.full.pdf.

[54] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: (Aug. 2020), p. 622803.

[55] Adam J. Riesselman, John B. Ingraham, and Debora S. Marks. "Deep generative models of genetic variation capture the effects of mutations". In: *Nature Methods* 15.10 (2018), pp. 816–822. ISSN: 1548-7091. DOI: 10.1038/s41592-018-0138-4. URL: http://www.nature.com/articles/s41592-018-0138-4.

[56] Kevin K Yang et al. "Learned protein embeddings for machine learning". In: *Bioinformatics* 34.15 (2018), pp. 2642–2648.

[57] Alex Wang et al. "Superglue: A stickier benchmark for general-purpose language understanding systems". In: *arXiv* (2019).

[58] Sara El-Gebali et al. "The Pfam protein families database in 2019". In: *Nucleic Acids Research* 47.D1 (2019), pp. D427–D432. ISSN: 0305-1048. DOI: 10.1093/nar/gky995. URL: https://academic.oup.com/nar/article/47/D1/D427/5144153.

[59] Michael Schantz Klausen et al. "NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning". In: *Proteins: Structure, Function, and Bioinformatics* (2019).

[60] Alexey Drozdetskiy et al. "JPred4: a protein secondary structure prediction server". In: *Nucleic Acids Research* 43.W1 (2015), W389–W394. ISSN: 0305-1048. DOI: 10.1093/nar/gkv332. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv332.

[61] James A Cuff and Geoffrey J Barton. "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction". In: *Proteins: Structure, Function, and Bioinformatics* 34.4 (1999), pp. 508–519.

[62] David E. Kim et al. "One contact for every twelve residues allows robust and accurate topology-level protein structure modeling". In: *Proteins: Structure, Function, and Bioinformatics* 82 (2014), pp. 208–218. ISSN: 08873585. DOI: 10.1002/prot.24374. URL: http://www.ncbi.nlm.nih.gov/pubmed/23900763%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4128384%20http://doi.wiley.com/10.1002/prot.24374.

[63] Jie Hou, Badri Adhikari, and Jianlin Cheng. "DeepSF: deep convolutional neural network for mapping protein sequences to folds". In: *Bioinformatics* 34.8 (2017), pp. 1295–1303.

[64] Leticia Stephan Tavares et al. "Strategies and molecular tools to fight antimicrobial resistance: resistome, transcriptome, and antimicrobial peptides". In: *Frontiers in microbiology* 4 (2013), p. 412.

[65] Karen S Sarkisyan et al. "Local fitness landscape of the green fluorescent protein". In: *Nature* 533.7603 (2016), p. 397.

[66] Kevin K Yang, Zachary Wu, and Frances H Arnold. "Machine learning in protein engineering". In: *arXiv* (2018).

[67] Gabriel J Rocklin et al. "Global analysis of protein folding using massively parallel design, synthesis, and testing". In: *Science* 357.6347 (2017), pp. 168–175.

[68] Jeff Donahue et al. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *Journal of Machine Learning Research* (2013). arXiv: 1310.1531. URL: http://arxiv.org/abs/1310.1531.

[69] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[70] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. "Dilated Residual Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[71] Alex Wang et al. "Glue: A multi-task benchmark and analysis platform for natural language understanding". In: *arXiv* (2018).

[72] Ben Krause et al. "Multiplicative LSTM for sequence modelling". In: *arXiv* (2016).

[73] Jianzhu Ma et al. "Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning". In: *Bioinformatics* 31.21 (2015), pp. 3506–3513.

[74] Nicholas Bhattacharya et al. "Interpreting Potts and Transformer Protein Models Through the Lens of Simplified Attention". In: *Pac. Symp. Biocomput.* 27 (2022), pp. 34–45.

[75] Sivaraman Balakrishnan et al. "Learning generative models for protein fold families". In: *Proteins: Structure, Function, and Bioinformatics* 79.4 (2011), pp. 1061–1078. ISSN: 08873585. DOI: 10.1002/prot.22934. URL: http://doi.wiley.com/10.1002/prot.22934.

[76] Magnus Ekeberg et al. "Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models". In: *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 87.1 (Jan. 2013), p. 012707.

[77] Stefan Seemayer, Markus Gruber, and Johannes Söding. "CCMpred - Fast and precise prediction of protein residue-residue contacts from correlated mutations". In: *Bioinformatics* 30.21 (2014), pp. 3128–3130. ISSN: 14602059. DOI: 10.1093/bioinformatics/btu500. URL: https://pubmed.ncbi.nlm.nih.gov/25064567/.

[78] Bohdan Monastyrskyy et al. "New encouraging developments in contact prediction: Assessment of the CASP 11 results". In: *Proteins: Structure, Function, and Bioinformatics* 84 (2016), pp. 131–144.

[79] Sheng Wang et al. "Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model". In: *PLOS Computational Biology* 13.1 (Jan. 2017), pp. 1–34. DOI: 10.1371/journal.pcbi.1005324. URL: https://doi.org/10.1371/journal.pcbi.1005324.

[80] Jianyi Yang et al. "Improved protein structure prediction using predicted inter-residue orientations". In: *bioRxiv* (2019), p. 846279. DOI: 10.1101/846279. URL: https://www.biorxiv.org/content/10.1101/846279v1.

[81] Ahmed Elnaggar et al. "ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.10 (Oct. 2022), pp. 7112–7127. URL: http://dx.doi.org/10.1109/TPAMI.2021.3095381.

[82] Ananthan Nambiar et al. "Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks". In: *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. BCB '20. Virtual Event, USA: Association for Computing Machinery, 2020. ISBN: 9781450379649. DOI: 10.1145/3388440.3412467. URL: https://doi.org/10.1145/3388440.3412467.

[83] Amy X Lu et al. "Self-Supervised Contrastive Learning of Protein Representations By Mutual Information Maximization". In: *bioRxiv* (Sept. 2020), p. 2020.09.04.283929. DOI: 10.1101/2020.09.04.283929. URL: https://doi.org/10.1101/2020.09.04.283929.

[84] Amir Shanehsazzadeh, David Belanger, and David Dohan. "Is Transfer Learning Necessary for Protein Landscape Prediction?" In: (2020). arXiv: 2011.03443 [q-bio.BM].

[85] Christian Dallago et al. "FLIP: Benchmark tasks in fitness landscape inference for proteins". In: *NeurIPS 2021 Track Datasets and Benchmarks*. Aug. 2021.

[86] Minkyung Baek et al. "Accurate prediction of protein structures and interactions using a three-track neural network". In: *Science* 373.6557 (2021), pp. 871–876.

[87] L. Steven Johnson, Sean R. Eddy, and Elon Portugaly. "Hidden Markov model speed heuristic and iterative HMM search procedure". In: *BMC Bioinformatics* 11.1 (2010), p. 431. ISSN: 14712105. DOI: 10.1186/1471-2105-11-431. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-431.

[88] Steve W. Lockless and Rama Ranganathan. "Evolutionarily conserved pathways of energetic connectivity in protein families". In: *Science* 286.5438 (1999), pp. 295–299. ISSN: 00368075. DOI: 10.1126/science.286.5438.295.

[89] Anthony A. Fodor and Richard W. Aldrich. "On Evolutionary Conservation of Thermodynamic Coupling in Proteins". In: *Journal of Biological Chemistry* 279.18 (2004), pp. 19046–19050. ISSN: 00219258. DOI: 10.1074/jbc.M402560200. URL: https://pubmed.ncbi.nlm.nih.gov/15023994/.

[90] John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg. "Graphical models of residue coupling in protein families". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 5.2 (2008), pp. 183–197.

[91] Martin Weigt et al. "Identification of direct residue contacts in protein–protein interaction by message passing". In: *Proc. Natl. Acad. Sci. U. S. A.* 106.1 (Jan. 2009), pp. 67–72.

[92] SK Burley and GA Petsko. "Weakly polar interactions in proteins". In: *Advances in Protein Chemistry* 39 (1988), pp. 125–189.

[93] Rainer Jaenicke. "Stability and stabilization of globular proteins in solution". In: *Journal of Biotechnology* 79.3 (2000), pp. 193–203.

[94] William R Taylor and Michael I Sadowski. "Structural constraints on the covariance matrix derived from multiple aligned protein sequences". In: *PLoS One* 6.12 (2011), e28265.

[95] Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. "Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era". In: *Proc. Natl. Acad. Sci. U. S. A.* 110.39 (Sept. 2013), pp. 15674–15679.

[96] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv* (2016).

[97] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[98] Baris E Suzek et al. "UniRef: comprehensive and non-redundant UniProt reference clusters". In: *Bioinformatics* 23.10 (May 2007), pp. 1282–1288.

[99] Martin Steinegger et al. "HH-suite3 for fast remote homology detection and deep protein annotation". In: *BMC Bioinformatics* 20.1 (2019), p. 473. ISSN: 14712105. DOI: 10.1186/s12859-019-3019-7. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3019-7.

[100] S D Dunn, L M Wahl, and G B Gloor. "Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction". In: *Bioinformatics* 24.3 (Feb. 2008), pp. 333–340.

[101] William Falcon. *PyTorch Lightning*. 2019. URL: https://www.pytorchlightning.ai.

[102] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[103] Joerg Schaarschmidt et al. "Assessment of contact predictions in CASP12: co-evolution and deep learning coming of age". In: *Proteins: Structure, Function, and Bioinformatics* 86 (2018), pp. 51–66.

[104] Rojan Shrestha et al. "Assessing the accuracy of contact predictions in CASP13". In: *Proteins: Structure, Function, and Bioinformatics* 87.12 (2019), pp. 1058–1068.

[105] Roshan M Rao et al. "MSA Transformer". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8844–8856. URL: https://proceedings.mlr.press/v139/rao21a.html.

[106] John-Marc Chandonia, Naomi K Fox, and Steven E Brenner. "SCOPe: classification of large macromolecular structures in the structural classification of proteins—extended database". In: *Nucleic Acids Research* 47.D1 (2019), pp. D475–D481.

[107] Ian Sillitoe et al. "CATH: increased structural coverage of functional space". In: *Nucleic Acids Research* 49.D1 (2021), pp. D266–D273.

[108] Neil Thomas et al. "Tuned Fitness Landscapes for Benchmarking Model-Guided Protein Design". In: *bioRxiv* (Oct. 2022), p. 2022.10.28.514293. URL: https://www.biorxiv.org/content/10.1101/2022.10.28.514293v1?rss=1.

[109] Frances H Arnold. "Design by directed evolution". In: *Acc. Chem. Res.* 31.3 (Mar. 1998), pp. 125–131.

[110] Philip A Romero and Frances H Arnold. "Exploring protein fitness landscapes by directed evolution". In: *Nat. Rev. Mol. Cell Biol.* 10.12 (Dec. 2009), pp. 866–876.

[111] S B Jennifer Kan et al. "Directed evolution of cytochrome c for carbon-silicon bond formation: Bringing silicon to life". In: *Science* 354.6315 (Nov. 2016), pp. 1048–1051.

[112] John A McIntosh et al. "Engineered Ribosyl-1-Kinase Enables Concise Synthesis of Molnupiravir, an Antiviral for COVID-19". In: *ACS Cent. Sci.* (Oct. 2021).

[113] Cara A Tracewell and Frances H Arnold. "Directed enzyme evolution: climbing fitness peaks one amino acid at a time". In: *Curr. Opin. Chem. Biol.* 13.1 (Feb. 2009), pp. 3–9.

[114] Jonathan C Greenhalgh et al. "Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production". In: *Nat. Commun.* 12.1 (Oct. 2021), p. 5825.

[115] Daniel M. Weinreich et al. "Darwinian Evolution Can Follow Only Very Few Mutational Paths to Fitter Proteins". In: *Science (New York, N.Y.)* 312.5770 (Apr. 2006), pp. 111–114. ISSN: 1095-9203. DOI: 10.1126/science.1123539.

[116] Johannes Neidhart, Ivan G. Szendro, and Joachim Krug. "Adaptation in Tunably Rugged Fitness Landscapes: The Rough Mount Fuji Model". In: *Genetics* 198.2 (Oct. 2014), pp. 699–721. ISSN: 0016-6731, 1943-2631. DOI: 10.1534/genetics.114.167668.

[117] Atish Agarwala and Daniel S Fisher. "Adaptive walks on high-dimensional fitness landscapes and seascapes with distance-dependent statistics". In: *Theor. Popul. Biol.* 130 (Dec. 2019), p. 435669. URL: http://dx.doi.org/10.1016/j.tpb.2019.09.011.

[118] Bruce J Wittmann, Yisong Yue, and Frances H Arnold. "Informed training set design enables efficient machine learning-assisted directed protein evolution". In: *Cell Syst* (Aug. 2021).

[119] Kevin K Yang, Zachary Wu, and Frances H Arnold. "Machine-learning-guided directed evolution for protein engineering". en. In: *Nat. Methods* 16.8 (Aug. 2019), pp. 687–694. URL: http://dx.doi.org/10.1038/s41592-019-0496-6.

[120] Philip A Romero, Andreas Krause, and Frances H Arnold. "Navigating the protein fitness landscape with Gaussian processes". In: *Proc. Natl. Acad. Sci. U. S. A.* 110.3 (Jan. 2013), E193–201.

[121] Chase R Freschlin, Sarah A Fahlberg, and Philip A Romero. "Machine learning to navigate fitness landscapes for protein engineering". In: *Curr. Opin. Biotechnol.* 75 (Apr. 2022), p. 102713.

[122] Danqing Zhu et al. "Machine learning-based library design improves packaging and diversity of adeno-associated virus (AAV) libraries". In: *bioRxiv* (Nov. 2021), p. 2021.11.02.467003.

[123] Drew H Bryant et al. "Deep diversification of an AAV capsid protein by machine learning". In: *Nat. Biotechnol.* (Feb. 2021).

[124] Surojit Biswas et al. "Low-N protein engineering with data-efficient deep learning". In: *Nat. Methods* 18.4 (Apr. 2021), pp. 389–396.

[125] Sam Sinai et al. "AdaLead: A simple and robust adaptive greedy search algorithm for sequence design". In: (Oct. 2020). arXiv: 2010.02141 [cs.LG].

[126] Sam Sinai and Eric D Kelsic. "A primer on model-guided exploration of fitness landscapes for biological sequence design". In: (Oct. 2020). arXiv: 2010.10614 [q-bio.QM].

[127] Yuxiang Jiang et al. "An expanded evaluation of protein function prediction methods shows an improvement in accuracy". In: *Genome Biol.* 17.1 (Sept. 2016), p. 184.

[128] Vanessa E Gray et al. "Quantitative Missense Variant Effect Prediction Using Large-Scale Mutagenesis Data". In: *Cell Syst* 6.1 (Jan. 2018), 116–124.e3.

[129] Christian Dallago et al. "FLIP: Benchmark tasks in fitness landscape inference for proteins". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021. URL: https://openreview.net/forum?id=p2dMLEwL8tF.

[130] Douglas M Fowler and Stanley Fields. "Deep mutational scanning: a new style of protein science". In: *Nat. Methods* 11.8 (Aug. 2014), pp. 801–807.

[131] Nicholas C Wu et al. "Adaptation in protein fitness landscapes is facilitated by indirect paths". In: *Elife* 5 (July 2016).

[132] Oksana M Subach et al. "Structural characterization of acylimine-containing blue and red chromophores in mTagBFP and TagRFP fluorescent proteins". In: *Chem. Biol.* 17.4 (Apr. 2010), pp. 333–341.

[133] Christof Angermueller et al. "Model-based reinforcement learning for biological sequence design". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HklxbgBKvr.

[134] E D Weinberger. "Local properties of Kauffman's N-k model: A tunably rugged energy landscape". In: *Phys. Rev. A* 44.10 (Nov. 1991), pp. 6399–6413.

[135] Sungmin Hwang et al. "Universality Classes of Interaction Structures for NK Fitness Landscapes". In: *J. Stat. Phys.* 172.1 (July 2018), pp. 226–278.

[136] Christof Angermueller et al. "Population-Based Black-Box Optimization for Biological Sequence Design". In: (June 2020). arXiv: 2006.03227 [cs.LG].

[137] Atish Agarwala and Daniel S Fisher. "Adaptive walks on high-dimensional fitness landscapes and seascapes with distance-dependent statistics". In: *Theor. Popul. Biol.* 130 (Dec. 2019), pp. 13–49.

[138] R B Potts. "Some generalized order-disorder transformations". In: *Math. Proc. Cambridge Philos. Soc.* 48.1 (Jan. 1952), pp. 106–109.

[139] A S Lapedes et al. *Correlated mutations in protein sequences: Phylogenetic and structural effects*. Tech. rep. Dec. 1998.

[140] H Kacser and J A Burns. "The molecular basis of dominance". In: *Genetics* 97.3-4 (Mar. 1981), pp. 639–666.

[141] Lizhi Ian Gong, Marc A Suchard, and Jesse D Bloom. "Stability-mediated epistasis constrains the evolution of an influenza protein". In: *Elife* 2 (May 2013), e00631.

[142] Chuan Li et al. "The fitness landscape of a tRNA gene". In: *Science* 352.6287 (May 2016), pp. 837–840.

[143] Jakub Otwinowski, David M McCandlish, and Joshua B Plotkin. "Inferring the shape of global epistasis". In: *Proc. Natl. Acad. Sci. U. S. A.* 115.32 (Aug. 2018), E7550–E7558.

[144] Richard A. Neher and Boris I. Shraiman. "Competition between Recombination and Epistasis Can Cause a Transition from Allele to Genotype Selection". In: *Proceedings of the National Academy of Sciences* 106.16 (Apr. 2009), pp. 6866–6871. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0812560106.

[145] Stuart A. Kauffman and Edward D. Weinberger. "The NK Model of Rugged Fitness Landscapes and Its Application to Maturation of the Immune Response". In: *Journal of Theoretical Biology* 141.2 (Nov. 1989), pp. 211–245. ISSN: 0022-5193. DOI: 10.1016/S0022-5193(89)80019-0.

[146] Ronny Lorenz et al. "ViennaRNA Package 2.0". In: *Algorithms Mol. Biol.* 6 (Nov. 2011), p. 26.

[147] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* (July 2021), pp. 1–11.

[148] Christoffer Norn et al. "Protein sequence design by conformational landscape optimization". In: *Proc. Natl. Acad. Sci. U. S. A.* 118.11 (Mar. 2021).

[149] Ziyue Yang, Katarina A Milas, and Andrew D White. "Now What Sequence? Pre-trained Ensembles for Bayesian Optimization of Protein Sequences". In: *bioRxiv* (Aug. 2022), p. 2022.08.05.502972.

[150] Nathan Killoran et al. "Generating and designing DNA with deep generative models". In: (Dec. 2017). arXiv: `1712.06148 [cs.LG]`.

[151] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. *Inceptionism: Going Deeper into Neural Networks*. en. `https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`. Accessed: 2022-10-8. June 2015.

[152] Brandon Carter et al. "Critiquing Protein Family Classification Models Using Sufficient Input Subsets". In: *J. Comput. Biol.* (Dec. 2019).

[153] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: (June 2017). arXiv: `1706.06083 [stat.ML]`.

[154] Rebecca F Alford et al. "The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design". In: *J. Chem. Theory Comput.* 13.6 (June 2017), pp. 3031–3048.

[155] David H Brookes, Amirali Aghazadeh, and Jennifer Listgarten. "On the sparsity of fitness functions and implications for learning". In: *Proc. Natl. Acad. Sci. U. S. A.* 119.1 (Jan. 2022).

[156] Roshan Rao et al. "Transformer protein language models are unsupervised structure learners". In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=fylclEqgvgd`.

[157] Matteo Bisardi et al. "Modeling sequence-space exploration and emergence of epistatic signals in protein evolution". In: (June 2021). arXiv: `2106.02441 [q-bio.BM]`.

[158] Jianzhu Ma et al. "Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning". In: *Bioinformatics* 31.21 (Nov. 2015), pp. 3506–3513.

[159] Andrew W Senior et al. "Improved protein structure prediction using potentials from deep learning". In: *Nature* 577.7792 (Jan. 2020), pp. 706–710.

[160] Maxwell L Bileschi et al. "Using deep learning to annotate the protein universe". In: *Nat. Biotechnol.* 40.6 (June 2022), pp. 932–937.

[161] Kevin K Yang, Alex X Lu, and Nicolo Fusi. "Convolutions are competitive with transformers for protein sequence pretraining". In: *bioRxiv* (May 2022), p. 2022.05.19.492714.

[162] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (Dec. 2014). arXiv: `1412.6980 [cs.LG]`.

[163] Danqing Zhu et al. "Optimal trade-off control in machine learning-based library design, with application to adeno-associated virus (AAV) for gene therapy". In: *bioRxiv* (Sept. 2022), p. 2021.11.02.467003.

[164] Yuedong Yang et al. "Sixty-five years of the long march in protein secondary structure prediction: the final stretch?" In: *Briefings in bioinformatics* 19.3 (2016), pp. 482–494.

[165] Raymond C Stevens. "The cost and value of three-dimensional protein structure". In: *Drug Discovery World* 4.3 (2003), pp. 35–48.

[166] Cindy J Castelle and Jillian F Banfield. "Major new microbial groups expand diversity and alter our understanding of the tree of life". In: *Cell* 172.6 (2018), pp. 1181–1197.

[167] Stefan Seemayer, Markus Gruber, and Johannes Söding. "CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations". In: *Bioinformatics* 30.21 (2014), pp. 3128–3130. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btu500. URL: http://www.ncbi.nlm.nih.gov/pubmed/25064567%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4201158%20https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu500.

[168] Tianqi Chen et al. "Training Deep Nets with Sublinear Memory Cost". In: *arXiv* (2016). arXiv: 1604.06174. URL: http://arxiv.org/abs/1604.06174.

[169] Robbie P Joosten et al. "A series of PDB related databases for everyday needs". In: *Nucleic Acids Research* 39.suppl_1 (2010), pp. D411–D419.

[170] Duncan P Brown, Nandini Krishnamurthy, and Kimmen Sjölander. "Automated protein subfamily identification and classification". In: *PLoS Comput Biol* 3.8 (2007), e160.

[171] Duccio Malinverni and Alessandro Barducci. "Coevolutionary Analysis of Protein Subfamilies by Sequence Reweighting". In: *Entropy* 21.11 (2019), p. 1127.

[172] Sergey Ovchinnikov, Hetunandan Kamisetty, and David Baker. "Robust and accurate prediction of residue-residue interactions across protein interfaces using evolutionary information". In: *Elife* 3 (May 2014), e02030.

[173] Justas Dauparas et al. "Unified framework for modeling multivariate distributions in biological sequences". In: (June 2019). arXiv: 1906.02598 [q-bio.QM].

[174] Baris E Suzek et al. "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches". In: *Bioinformatics* 31.6 (2015), pp. 926–932.

[175] Joerg Schaarschmidt et al. "Assessment of contact predictions in CASP12: Co-evolution and deep learning coming of age". In: *Proteins* 86 Suppl 1 (Mar. 2018), pp. 51–66.

[176] Rojan Shrestha et al. "Assessing the accuracy of contact predictions in CASP13". In: *Proteins* 87.12 (Dec. 2019), pp. 1058–1068.

[177] S D Dunn, L M Wahl, and G B Gloor. "Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction". In: *Bioinformatics* 24.3 (Feb. 2008), pp. 333–340.

Appendix A

Evaluating Protein Transfer Learning with TAPE

A.1 Dataset Details

In Table A.1 we show the size of all train, validation, and test sets.

We provide further details about dataset sources, preprocessing decisions, data splitting, and experimental challenges in obtaining labels for each of our supervised tasks below. For ease of reading, each section starts with the following items:
(Dataset) The source of the dataset and creation of train/test splits.
(Labeling) The current approach to acquiring supervised labels for this task.

Secondary Structure Details

(Dataset) We use a training and validation set from Klausen et al. [59], which is filtered such that no two proteins have greater than 25% sequence identity. We use three test sets, CB513 [61], CASP12 [48], and TS115 [164]. The training set is also filtered at the 25%

Table A.1: Dataset sizes

Task	Train	Valid	Test
Language Modeling	32,207,059	N/A	2,147,130 (Random) / 44,313 (Heldout)
Secondary Structure	8,678	2,170	513 (CB513) / 115 (TS115) / 21 (CASP12)
Contact Prediction	25,299	224	40 (CASP12)
Remote Homology	12,312	736	718 (Fold) / 1254 (Super) / 1272 (Family)
Fluorescence	21,446	5,362	27,217
Stability	53,679	2,447	12,839

identity threshold with these test sets. This filtering tests the model's ability to generalize in the interesting case where test proteins are not closely related to train proteins.

(Labeling) Determining the secondary structure of a protein experimentally requires high-resolution imaging of the structure, a particularly labor intensive task for structural biologists. Imaging often uses Cryo Electron-Microscopy or X-Ray Crystallography, which can take between weeks and years and can cost over $\$200,000$ [165].

Contact Prediction Details

(Dataset) We use training, validation, and test sets from ProteinNet [49], which uses a test set based on the CASP12 [48] competition, with training and validation sets filtered at the 30% sequence identity threshold. This tests the ability of the model to generalize to proteins that are not closely related to any train proteins.

(Labeling) Determining the contacts of a protein requires knowing its full 3D structure. As with secondary structure, determining the 3D structure requires imaging a protein.

Remote Homology Details

(Dataset) We use a training, validation, and test set from [63], derived from the SCOP 1.75 database [7] of hierarchically classified protein domains. All proteins of a given fold are further categorized into related *superfamilies*. Entire superfamilies are held out from the training set, allowing us to evaluate how the model generalizes across evolutionary distance when structure is preserved.

(Labeling) Each fold is annotated from the structure of the sequence, which SCOP pulls from the Protein DataBank [11, 7]. Finding new superfamilies with the same fold is a challenging task, requiring sequencing in extreme environments as is often done in metagenomics [166].

Fluorescence Details

(Dataset) We use data generated by an experimental technique called Deep Mutational Scanning (DMS) [65]. This technique allows for extensive characterizations of small neighborhoods of a parent protein through mutagenesis. We create training, validation, and test splits ourselves, partitioning the data so that train and validation are in a Hamming distance 3 neighborhood of the original protein, while test data is a sample from the Hamming distance 4-15 neighborhood.

(Labeling) DMS is efficient for local characterization near a single protein, but its samples become vanishingly small once neighborhoods start to expand outside of Hamming distance 2.

Stability Details

(Dataset) We use data generated by a novel combination of parallel DNA synthesis and protein stability measurements [67]. We create training, validation, and test splits ourselves, partitioning the data so that training and validation sets come from four rounds of experimental data measuring stability for many candidate proteins, while our test set consists of seventeen 1-Hamming distance neighborhoods around promising proteins observed in the four rounds of experimentation.

(Labeling) This approach for observing stability is powerful because of its throughput, allowing the authors to find the most stable proteins ever observed for certain classes [67]. The authors observe that the computational methods used to guide their selection at each stage could be improved, meaning that in this case better models could actually lead to better labeled data in a virtuous cycle.

A.2 Featurization of Pretrained Models

We followed standard practice for feeding large pretrained models into downstream supervised architectures. For the Transformer, ResNet, and UniRep we extracted a vector of dimension 512, 256 and 1900, respectively, at each position. For the LSTM and Bepler, we obtained a forward and backward vector at each position, which we concatenated. This resulted in vectors of dimension 2048 and 1024, respectively. For details on how these vectors were used for downstream tasks, see the next section.

A.3 Supervised Architectures

For each task, we fixed one supervised architecture and tried one-hot, alignment-based, and neural net based features. We did not perform hyperparameter tuning or significant architecture optimization, as the main goal was to compare feature extraction techniques.

For each task we define the supervised architecture below. If this is a state of the art architecture from other work, we highlight any novel training procedure or inputs they take.

Secondary Structure Architecture

We used the NetSurfP2.0 model from Klausen et al [59]. The model consists of two convolutional layers followed by two bidirection LSTM layers and a linear output layer. The convolutional layers have filter size 32 and kernel size 129 and 257, respectively. The bidirectional LSTM layers have 1024 hidden units each.

In the original model, the authors take in thfe outputs of an HMM-HMM alignment method called HHblits [41] in addition to a one-hot encoding of the sequence, giving 50-dimensional inputs at each position. They train the model on multiple tasks including secondary structure prediction (3 and 8 class), bond-angle prediction, and solvent accessibility

prediction. For clarity, we only compared to the model trained without the multitask training, which in our experiments contributed an extra one to two percent in test accuracy. In addition to multitask training, they balance the losses between different tasks to achieve maximum accuracy on secondary structure prediction. All features and code to do the full multitask training is available in our repository.

Contact Prediction Architecture

We used a supervised network inspired by the RaptorX-Contact model from Ma et al [73]. Since a contact map is a 2D pairwise prediction, we form a 2D input from our 1D features by concatenating the features at position i and j for all i, j. This 2D input is then passed through a convolutional residual network with. The 2D network contains 30 residual blocks with two convolutional layers each. Each convolution in the residual block has filter size 64 and a kernel size of 3.

The original RaptorX method inputs uses alignment-based methods to find similar proteins, then passes these through CCMpred [167] - a Markov Random Field based contact prediction method. This outputs a 2D featurization including mutual information and pairwise potential. This, along with 1D HMM alignment features and the one-hot encoding of each amino acid are fed their network. Unfortunately the code and features are not publically available, so we used the 1D alignment-based features available in ProteinNet [49] instead. While this improved performance significantly, the numbers reported by RaptorX are higher than those we obtained with our implementation.

Remote Homology Architecture

Remote homology requires a single prediction for each protein. To obtain a sequence-length invariant protein embedding we compute an attention-weighted mean of the amino acid embeddings. More precisely, we predict an attention value for each position in the sequence using a trainable dense layer, then use those attention values to compute an attention-weighted mean protein embedding. This protein embedding is then passed through a 512 hidden unit dense layer, a relu nonlinearity, and a final linear output layer to predict logits for all 1195 classes. We note that Hou et al. [63] propose a deep architecture for this task and report state of the art performance. When we compared the performance of this supervised architecture to that of the attention-weighted mean above, the attention-based embedding performed better for all featurizations. As such, we choose to report results using the simpler attention-based downstream architecture.

The current state of the art method in this problem, DeepSF [63], takes in a one-hot encoding of the amino acids, predicted secondary structure labels, predicted solvent accessibility labels, and a 1D alignment-based features. In an ablation study, the authors show that the secondary structure labels are most useful for performance of their model. We report only one-hot and alignment-based results in the main paper to maintain consistency

with alignment-based featurizations for other tasks. All input features used by DeepSF are available in our repository.

Protein Engineering Architectures

Protein engineering also requires a single prediction for each protein. Therefore we use the same architecture as we do for remote homology, computing an attention-weighted mean protein embedding, a dense layer with 512 hidden units, a relu nonlinearity and a final linear output layer to predict the quantity of interest (either stability or fluorescence).

Since we create these training, validation, and test splits ourselves, no clear previous state of the art exists. Related work on protein engineering has used a similar architecture by computing a single protein embedding followed by some form of projection (linear or with a small feed forward network) [39, 54]. These methods also do not take in alignment-based features and only use one-hot amino acids as inputs.

A.4 Training Details

Self-supervised models were all trained on four NVIDIA V100 GPUs on AWS for 1 week. Training used a learning rate of 10^{-3} with a linear warm up schedule, the Adam optimizer, and a 10% dropout rate. Since proteins vary in length significantly, we use variable batch sizes depending on the length of the protein. These sizes also differ based on model architecture, as some models (e.g. the Transformer) have significantly higher memory requirements. Specific batch sizes for each model at each protein length are available in our repository.

Supervised models were trained on two GPUs until convergence (no increase in validation accuracy for 10 epochs) with the exception of the memory-intensive Contact Prediction task, which was trained on four GPUs until convergence. Training used a learning rate of 10^{-4} with a linear warm up schedule, the Adam optimizer, and a 10% dropout rate. We backpropagated fully through all models during supervised fine-tuning.

In addition, due to high memory requirements of some downstream tasks (especially contact prediction) we use memory saving gradients [168] to fit more examples per batch on the GPU.

A.5 Pfam Heldout Families

The following Pfam clans were held out during self-supervised training: CL0635, CL0624, CL0355, CL0100, CL0417, CL0630. The following Pfam families were held out during self-supervised training: PF18346, PF14604, PF18697, PF03577, PF01112, PF03417. First, a "clan" is a cluster of families grouped by the maintainers of Pfam based on shared function or evolutionary origin (see [58] for details). We chose holdout clans and families in pairs, where a clan of novel function is held out together with a family that is similar in sequence but different evolutionarily or functionally. This serves to simultaneously test generalization

Table A.2: Results for small pretrained models on downstream supervised tasks

Method	Structure		Evolutionary	Engineering	
	SS	Contact	Homology	Fluorescence	Stability
Transformer (small)	0.70	0.31	0.13	0.68	0.68
LSTM (small)	0.73	0.26	0.18	0.66	0.67
ResNet (small)	0.73	0.37	0.11	0.43	0.68

across large distances (entirely held out families) and between similar looking unseen groups (e.g. the paired holdout clan and holdout family).

A.6 Bepler Supervised Training

We perform supervised pretraining using the same architecture described in Bepler et al. [38]. We train on the same tasks, a paired remote homology task and contact map prediction task. However, in order to accurately report results on downstream secondary structure, contact map, and remote homology datasets, which were filtered by sequence identity, we perform this same sequence identity filtering on the supervised pretraining set. This reduced the supervised pretraining dataset size by 75% which likely reduced the effectiveness of the supervised pretraining. Both filtered and unfiltered supervised pretraining datasets are made available in our repository.

A.7 Model Size Ablation

In this benchmark we made the choice to train relatively large, 40 million parameter models as larger models have been found to improve performance in other applications of deep learning. To determine whether this trend holds for our benchmark, as well as to quantify the performance difference, we evaluate smaller versions of our three models (the Transformer, LSTM, and ResNet).

Our Transformer model has 6 layers, a hidden dimension of 256, a filter dimension of 512, and 8 attention heads, for a total of 3,315,200 parameters. Our LSTM model has 3 layers with 128 hidden units each, for a total of 796288 parameters. Our ResNet has 8 layers, a filter size of 256, a kernel size of 3, and a dilation rate of 2, for a total of 3,268,992 parameters. Each model was trained for 1,000,000 gradient updates on Pfam, in the same manner that the corresponding large models were trained. Results are reported in Table A.2.

We note several interesting phenomena from this table. First, we see a drop in performance across all models and tasks, with the exception of the ResNet on the Contact Prediction task and the Transformer on the Fluorescence task. Second, with the exception of the

Table A.3: Detailed secondary structure results

		Three-Way Accuracy (Q3)			Eight-Way Accuracy (Q8)		
		CB513	CASP12	TS115	CB513	CASP12	TS115
No Pretrain	Transformer	0.70	0.68	0.73	0.51	0.52	0.58
	LSTM	0.71	0.69	0.74	0.47	0.48	0.52
	ResNet	0.70	0.68	0.73	0.55	0.56	0.61
Pretrain	Transformer	0.73	0.71	0.77	0.59	0.59	0.64
	LSTM	0.75	0.70	0.78	0.59	0.57	0.66
	ResNet	0.75	0.72	0.78	0.58	0.58	0.64
	Bepler [38]	0.73	0.70	0.76	0.58	0.57	0.65
	Alley [39]	0.73	0.72	0.77	0.57	0.59	0.63
Baseline	One-hot	0.69	0.68	0.72	0.52	0.53	0.58
	Alignment	**0.8**	**0.76**	**0.81**	**0.63**	**0.61**	**0.68**

Contact Prediction task, the relative ordering of the models is preserved, even while overall performance decreases. As the Contact Prediction task has the most complicated downstream architecture, it suggests that the downstream architecture has a large effect on performance.

A.8 Detailed Results on Supervised Tasks

Here we provide detailed results on each task, examining multiple metrics and test-conditions to further determine what the models are learning.

Secondary Structure Results

We perform both three-class and eight-class secondary structure classification following the DSSP labeling system [169]. Three way classification tags each position as either Helix, Strand or Other. Eight-way classification breaks these three labels into more specialized classes, for example Helix is broken into 3-turn, 4-turn or 5-turn helix. Table A.3 shows results on these tasks. We note that test-set performance is comparable for all three test sets, in particular alignment does better at both eight-way and three-way classification by a large margin.

We follow the standard notation, where Q3 refers to three-way classification accuracy and Q8 refers to eight-way classification accuracy.

Table A.4: Detailed short-range contact prediction results. Short range contacts are contacts between positions separated by 6 to 11 positions, inclusive.

		AUPRC	P@L	P@L/2	P@L/5
No Pretrain	Transformer	0.29	0.25	0.32	0.4
	LSTM	0.23	0.22	0.26	0.33
	ResNet	0.2	0.18	0.24	0.31
Pretrain	Transformer	0.35	0.28	0.35	0.46
	LSTM	0.35	0.26	0.36	0.49
	ResNet	0.32	0.25	0.34	0.46
	Bepler [38]	0.33	0.27	0.35	0.44
	Alley [39]	0.27	0.23	0.3	0.39
Baseline	One-hot	0.3	0.26	0.34	0.42
	Alignment	**0.51**	**0.35**	**0.5**	**0.66**

Table A.5: Detailed medium-range contact prediction results. Medium range contacts are contacts between positions separated by 12 to 23 positions, inclusive.

		AUPRC	P@L	P@L/2	P@L/5
No Pretrain	Transformer	0.2	0.18	0.24	0.31
	LSTM	0.13	0.13	0.15	0.19
	ResNet	0.15	0.14	0.18	0.23
Pretrain	Transformer	0.23	0.19	0.25	0.33
	LSTM	0.23	0.2	0.26	0.34
	ResNet	0.23	0.18	0.25	0.35
	Bepler [38]	0.26	0.22	0.29	0.37
	Alley [39]	0.2	0.17	0.23	0.32
Baseline	One-hot	0.2	0.17	0.23	0.3
	Alignment	**0.45**	**0.32**	**0.45**	**0.59**

Table A.6: Detailed long-range contact prediction results. Long range contacts are contacts between positions separated by 24 or more positions, inclusive.

		AUPRC	P@L	P@L/2	P@L/5
No Pretrain	Transformer	0.09	0.15	0.17	0.19
	LSTM	0.05	0.1	0.12	0.15
	ResNet	0.06	0.11	0.13	0.15
Pretrain	Transformer	0.1	0.17	0.2	0.24
	LSTM	0.11	0.2	0.23	0.27
	ResNet	0.06	0.1	0.13	0.17
	Bepler [38]	0.11	0.18	0.22	0.26
	Alley [39]	0.09	0.17	0.2	0.22
Baseline	One-hot	0.07	0.13	0.16	0.22
	Alignment	**0.2**	**0.33**	**0.42**	**0.51**

Contact Prediction Results

We report all metrics commonly used to capture contact prediction results [73] in tables A.5 and A.6. The metrics "P@K" are precision for the top K contacts, where all contacts are sorted from highest confidence to lowest confidence. Note that L is the length of the protein, so "P@L/2", for example, denotes the precision for the $L/2$ most likely predicted contacts in a protein of length L. In Table A.5 we report all metrics for medium range contacts, which are contacts for positions between five and twelve amino acids apart. In Table A.6 we report all metrics for long range contacts, which are contacts for positions greater than 12 amino acids apart.

All results decay as we transition from short range to long range contacts, which we note is *not* the case for many state of the art methods from recent CASP competitions [73, 23].

Remote Homology Results

In Table A.7, we report results on three remote homology test datasets constructed in Hou et al [63]. Recall that "Fold" has the most distantly related proteins from train, while "Superfamily" and "Family" are increasingly related (see Section A.1 for more details). This is reflected in the accuracy in Table A.7, which increases drastically as the test sets get easier.

Fluorescence Results

Fluorescence distribution in the train, validation, and test sets is bimodal, with one mode corresponding to bright proteins and one mode corresponding to dark proteins. The dark

Table A.7: Detailed remote homology prediction results

		Fold		Superfamily		Family	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
No Pretrain	Transformer	0.09	0.21	0.07	0.2	0.31	0.58
	LSTM	0.12	0.28	0.13	0.29	0.68	0.85
	ResNet	0.1	0.24	0.07	0.19	0.39	0.6
Pretrain	Transformer	0.21	0.37	0.34	0.51	0.88	0.94
	LSTM	**0.26**	**0.43**	**0.43**	**0.59**	**0.92**	**0.97**
	ResNet	0.17	0.29	0.31	0.44	0.77	0.87
	Bepler [38]	0.17	0.30	0.20	0.36	0.79	0.91
	Alley [39]	0.23	0.39	0.38	0.54	0.87	0.94
Baseline	One-hot	0.09	0.21	0.08	0.21	0.39	0.66
	Alignment	0.09	0.21	0.09	0.24	0.53	0.77

Table A.8: Detailed fluorescence prediction results. ρ denotes Spearman's ρ.

		Full Test Set		Bright Mode Only		Dark Mode Only	
		MSE	ρ	MSE	ρ	MSE	ρ
No Pretrain	Transformer	2.59	0.22	0.08	0.08	3.79	0
	LSTM	2.35	0.21	0.11	0.05	3.43	-0.01
	ResNet	2.79	-0.28	**0.07**	-0.07	4.1	-0.01
Pretrain	Transformer	0.22	**0.68**	0.09	0.60	0.29	**0.05**
	LSTM	**0.19**	0.67	0.12	0.62	**0.22**	0.04
	ResNet	3.04	0.21	0.12	0.05	4.45	0.02
	Bepler [38]	2.17	0.33	0.08	0.06	3.17	0.02
	Alley [39]	0.20	0.67	0.13	**0.63**	0.24	0.04
Baseline	One-hot	2.69	0.14	0.08	0.03	3.95	0.0

Table A.9: Overall stability prediction results

		Spearman's ρ	Accuracy
No Pretrain	Transformer	-0.06	0.5
	LSTM	0.28	0.6
	ResNet	0.61	0.68
Pretrain	Transformer	**0.73**	**0.70**
	LSTM	0.69	0.69
	ResNet	**0.73**	0.66
	Bepler [38]	0.64	0.67
	Alley [39]	**0.73**	0.69
Baseline	One-hot	0.19	0.58

mode is significantly more diverse in the test set than the train and validation sets, which makes sense as most random mutations will destroy the refined structure necessary for fluorescence. With this in mind, we report Spearman's ρ and mean-squared-error (MSE) on the whole test-set, on only dark mode, and on only the bright mode in Table A.8. The drop in MSE for both modes shows that pretraining helps our best models distinguish between dark and bright proteins. However low Spearman's ρ for the dark mode suggests that models are not able to rank proteins within this mode.

Stability Results

The goal of the Rocklin et al. [67] experiment was to find highly stable proteins. In the last stage of this experiment they examine variants of the the most promising candidate proteins. Therefore we wish to measure both whether our model was able to learn the landscape around these candidate proteins, as well as whether it successfully identified those variants with greater stability than the original parent proteins. In Table A.9 we report Spearman's ρ to measure the degree to which the landscape was learned. In addition, we report classification accuracy of whether a mutation is beneficial or harmful using the predicted stability of the parent protein as a decision boundary.

In Table A.10 report all metrics separately for each of the four protein topologies tested in Rocklin et al [67], where α denotes a helix and β denotes a strand (or β-sheet). We do this because success rates varied significantly by topology in their experiments, so some topologies (such as $\alpha\alpha\alpha$ were much easier to optimize than others (such as $\alpha\beta\beta\alpha$). We find that our prediction success also varies significantly by topology.

Table A.10: Stability prediction results broken down by protein topology

		$\alpha\alpha\alpha$		$\alpha\beta\beta\alpha$		$\beta\alpha\beta\beta$		$\beta\beta\alpha\beta\beta$	
		ρ	Acc	ρ	Acc	ρ	Acc	ρ	Acc
No Pretrain	Transformer	-0.39	0.49	-0.41	0.47	0.52	0.5	0.25	0.52
	LSTM	-0.07	0.57	0.39	0.7	-0.43	0.56	-0.34	0.56
	ResNet	0.64	0.69	0.16	0.69	0.63	0.67	0.65	0.67
Pretrain	Transformer	0.66	0.68	**0.48**	0.73	0.65	**0.71**	0.65	0.67
	LSTM	0.71	**0.7**	0.17	0.73	**0.68**	0.67	**0.67**	**0.7**
	ResNet	0.68	0.68	0.15	0.63	0.61	0.68	0.6	0.68
	Bepler [38]	0.33	0.66	0.24	**0.79**	0.54	0.7	0.58	0.53
	Alley [39]	**0.72**	0.66	0.11	0.76	**0.68**	0.66	0.65	0.67
Baseline	One-hot	0.58	0.59	0.04	0.58	-0.05	0.58	0.54	0.58

Appendix B

Interpreting Potts and Transformer Protein Models Through the Lens of Simplified Attention

B.1 Recovering Factored Attention from Standard Attention

Potts and Factored Attention estimate a single undirected graphical model from the training data. While a single graph can be a good approximation for the structure associated with a protein family, many families have *subfamilies* with different functional specializations and even different underlying contacts [170, 171]. Since subfamily identity is rarely known, allowing edge weights to be a function of sequence could enable the estimation of a family of graphs.

In the language of the Transformer, factored attention estimates a single graph because it computes queries and keys using only the positional encoding. We show more precisely that factored attention can be recovered from standard attention by computing queries and keys from one-hot positional encodings and values from one-hot sequence embeddings.

Single attention layer. Given a sequence of dense vectors $x = (x_1, \ldots, x_L)$ with $x_i \in \mathbb{R}^p$, the attention mechanism of the Transformer encoder (multihead scaled dot product self-attention) produces a continuous representation $y \in \mathbb{R}^{L \times p}$. If head size is d, this representation is computed using H heads (W_Q, W_K, W_V), where $W_Q, W_K, W_V \in \mathbb{R}^{p \times d}$. Queries, keys, and values are defined as $Q = xW_Q, K = xW_K, V = xW_V$. For a single head (W_Q, W_K, W_V), the output is given by

$$y = \text{softmax}\Big(\frac{QK^T}{\sqrt{d}}\Big)V.$$

The full output in $\mathbb{R}^{dH}$ is produced by concatenating all head outputs. A single Transformer encoder layer passes the output through a dense layer, applying layer-norms and residual

connection to aid optimization.

For the first layer, the input x is a sequence of discrete tokens. To produce a dense vector combining sequence and position information, positional encodings and sequence embeddings are combined. The positional encoding $E_{pos} \in \mathbb{R}^{L\times e}$ produces a dense vector of dimension e for each position i. The sequence embedding $E_{seq} \in \mathbb{R}^{A\times e}$ maps each element of the vocabulary to a dense vector of dimension e. Typically these are combined through summation to produce a dense vector $\tilde{x}_i = E_{seq}(x_i) + E_{pos}(i)$, which is input to the Transformer as described above.

For this paper, we use only multi-head self-attention without the dense layer, layer norm, or residual connections, as these drastically hurt performance when employed for one layer.

Factored attention from standard attention. Written explicitly, the input Transformer layer computes queries for a single head with $Q = (E_{pos} + E_{seq}(x))\, W_Q$. Keys and values are computed similarly. To recover factored attention, we instead compute queries and keys via $Q = E_{pos}W_Q$ and $K = E_{pos}W_K$, while values are given by $V = E_{seq}(x)W_V$. For simplicity, we one-hot encode both position and sequence, which corresponds using identity matrices $E_{pos} = I_L \in \mathbb{R}^{L\times L}$ and $E_{seq} = I_A \in \mathbb{R}^{A\times A}$.

Implicit single-site term in single-layer attention. For a single layer of attention, the product $E_{pos}W_V$ is a matrix in $\mathbb{R}^{L\times A}$. This matrix does not depend on sequence inputs, thus allowing it to act as a single-site term. This suggests why inclusion of an explicit single-site term in Figure B.10 had no effect for single-layer attention.

B.2 Losses

The loss for all three models is of the form

$$\ell(\theta; x) = \mathcal{L}(\theta; x) + cR(\theta), \tag{B.1}$$

where $\mathcal{L}$ is either pseudolikelihood or masked language modeling and R is a regularizer.

Potts regularization. Consider the order-4 interaction tensor W, where $W^{ij} \in \mathbb{R}^{A\times A}$ gives the parameters associated to edge (i, j). We regularize W by setting $R(\theta) = \sum_{i<j} \|W^{ij}\|_F^2$. This term is multiplied by $\lambda \cdot L \cdot A$, following [172].

Factored attention regularization. Since factored attention is also a fully connected pairwise MRF, we use identical regularization to that of Potts. The order-4 tensor W is given by

$$W_{ab}^{ij} = \sum_{h=1}^{H} \operatorname{symm}\Big(\operatorname{softmax}\left(W_Q^h W_K^{h\,T}\right)\Big)_{ij} W_V^h(a, b). \tag{B.2}$$

Single-layer attention regularization. Due the lack of an MRF interpretation for single-layer attention, we chose to use weight decay as is typically done for attention models. This corresponds to setting $R(\theta)$ to be the sum of Frobenius norm squared for all weights W_Q, W_K and W_V.

Single-site term. When any model has a single-site term, we follow standard practice and regularize its Frobenius norm squared.

B.3 ProtBERT-BFD head selection

layer	head	P@L
29	7	0.517
29	8	0.396
29	4	0.394
29	2	0.353
29	11	0.333
29	0	0.299
28	3	0.275
29	15	0.177
29	6	0.167
29	12	0.158
28	4	0.141
29	9	0.139
28	6	0.125
28	5	0.125
3	4	0.115
28	11	0.106

Table B.1: The top 16 heads in ProtBERT-BFD whose attention maps gave the most precise contact maps across 500 validation families. Most of the top performing heads are found in the last layer. The top six heads were selected for our contact extraction in all results.

B.4 Data and Metrics

B.5 Selection of Protein Families

We used the following sets of families for model development:

1. A set of 748 families was chosen for performance evaluation. All metrics reported in the paper are on this set, with a single choice of hyperparameters for Potts models, factored attention, and standard attention. The 748 families were chosen randomly from the Yang *et al.* [80] dataset, which consists of 15,051 MSAs generated from the databases UniClust30 and UniRef100 [98], as well as metagenomic datasets. Our random sample is representative of the range of MSA depths and protein lengths, see Figure B.1.

2. A set of six families from the 748 was chosen to choose hyperparameters for single-layer attention. They were chosen to span a range of MSA depth (large and small), as well

as three different regimes of Potts performance (Good, Ok, Poor). These families were used to tune hyperparameters as described in Section B.9. See Table B.2.

3. A set of ten families from the 748 was chosen where factored attention performed very poorly in our initial experiments. Half were chosen to be long proteins and the other half to be short. This set was used to optimize learning rate and regularization for factored attention to ensure reasonable model performance. See Table B.3.

4. 500 entirely disjoint families were further selected randomly from [80] and used to compute average precision for each head in ProtBERT-BFD [81]. Performance on these families was used for selecting the top 6 heads, see Table B.1.

PDB	Sequences	Length	Potts Performance
3er7_1_A	33673	118	Good
5fo5_1_B	17560	88	Ok
2w18_1_A	33619	308	Poor
4gnr_1_A	2073	351	Good
5mkc_1_A	515	207	Ok
3e9l_1_A	146	292	Poor

Table B.2: 6 families chosen for hyperparameter optimization for single-layer attention.

PDB	Sequences	Length
4k61_1_A	2145	140
4l3r_1_A	5535	143
3cy4_1_B	1064	154
6fdg_1_A	2325	155
3p6b_1_B	4353	186
1jm1_1_A	17130	202
4yt2_1_A	15481	343
3vmm_1_A	4383	471
4egc_1_A	9929	539
3gq7_1_A	6568	605

Table B.3: 10 families chosen for hyperparameter optimization for factored attention

B.6 Producing Contact Maps

A PDB structure gives 3D coordinates for every atom in a structure. We use Euclidean distance between the beta carbons to define distance between any pair of positions. In the

Figure B.1: The length and MSA size distribution for our 748 family subset (red) compared to the full 15,051 families in the trRosetta dataset selected for training

case of glycine, the alpha carbon is used. A pair of positions where this distance is less than 8Å is declared to be a contact.

B.7 Scoring Contact Predictions

Given a predicted contact map $\widehat{C} \in \mathbb{R}^{L \times L}$ and a true contact map $C \in \{0,1\}^{L \times L}$, we describe metrics for scoring $\widehat{C}$.

A sequence $x = (x_1, \ldots, x_L)$ of length L has $\binom{L}{2}$ potential contacts. Since we see $\mathcal{O}(L)$ contacts, contact prediction a sparse prediction task. Accordingly, we focus on precision-recall based quantitative analyses of $\hat{C}$. Common practice in the field is to sort all $\binom{L}{2}$ entries of $\hat{C}$ in decreasing order and evaluate precision at various length thresholds, such as the top L or $L/10$ predictions [104]. Note that this analysis is similar to choosing recall cutoffs

along a precision-recall curve, where sorted length index plays the role of recall on the x axis. Unlike recall, length-based cutoffs do not rely on knowledge of the actual number of contacts. In addition to the precision at various length (recall) cutoffs, we also computed Area Under the Precision-Recall Curve (AUC), which we define as the average of Precision @ L for $L = 1, ..., 10$. AUC is a widely used metric for comparing classifiers when the positive class is rare.

B.8 Hyperparameters

Potts. We used $\lambda = 0.5$, learning rate of 0.5, and batch size 4096. Pad, gap, and mask were all encoded with the same token. The Potts model is trained using a modified version of Adam presented in Ref. [173]. This modification was made to improve performance of Adam to match that of L-BFGS.

Factored attention. We AdamW with a learning rate of 5×10^{-3} and set $\lambda = 0.01$. The default head size was set to 32 unless stated otherwise.

Single-layer attention. We set embedding size of 256, head size of 64, and number of heads 128. The model is trained with AdamW using a learning rate of 5×10^{-3} and weight decay of 2×10^{-3}. Attention dropout of 0.05 is also applied. The batch size is 32 and mask prob for masked language modeling is 0.15. We use a separate mask token and pad,gap token.

ProtBERT-BFD. ProtBERT-BFD has 30 layers each with 16 heads and a hidden size of 1024. The training dataset is a mixture of UniRef50 [174] and BFD. It has $2,122$ million protein sequences. See [81] more information.

B.9 Hyperparameter Sweep Details

Potts. The Potts model implementation using psuedolikelihood has been optimized by others, so we did not tune performance. Since performance with MLM was comparable to pseudolikelihood, we did not sweep for MLM either.

Single-layer attention. Standard attention is by far the most sensitive model to hyperparameters. To find a reasonable set of hyperparameters, we first swept over the six families in Table B.2, performing a grid search over

- $H \in \{32, 64, 128, 256, 512\}$
- $d \in \{32, 64, 128, 256, 512\}$
- $e \in \{128, 256, 512\}$
- attention dropout in $\{0, 0.05, 0.1\}$
- learning rate in $\{5 \times 10^{-3}, 1 \times 10^{-2}\}$
- weight decay in $\{0, 1 \times 10^{-3}, 2 \times 10^{-3}\}$

We found that the choice $H = 256$, $d = 64$, $e = 256$, attention dropout of 0.05, learning rate of 5×10^{-3} and weight decay of $\times 10^{-3}$ performed well across all six families. Due to GPU memory constraints, we had to set $H = 128$ for further runs.

Factored attention. We swept factored attention over the families in Table B.3, performing a grid search over

- learning rate in $\{1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 5 \times 10^{-2}\}$
- regularization coefficient in $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}\}$

We found that learning rate of 5×10^{-3} and regularization of 0.01 were effective, but that other configurations such as regularization of 5×10^{-3} also performed well. Both H and d are evaluated extensively in our results.

B.10 Additional Figures

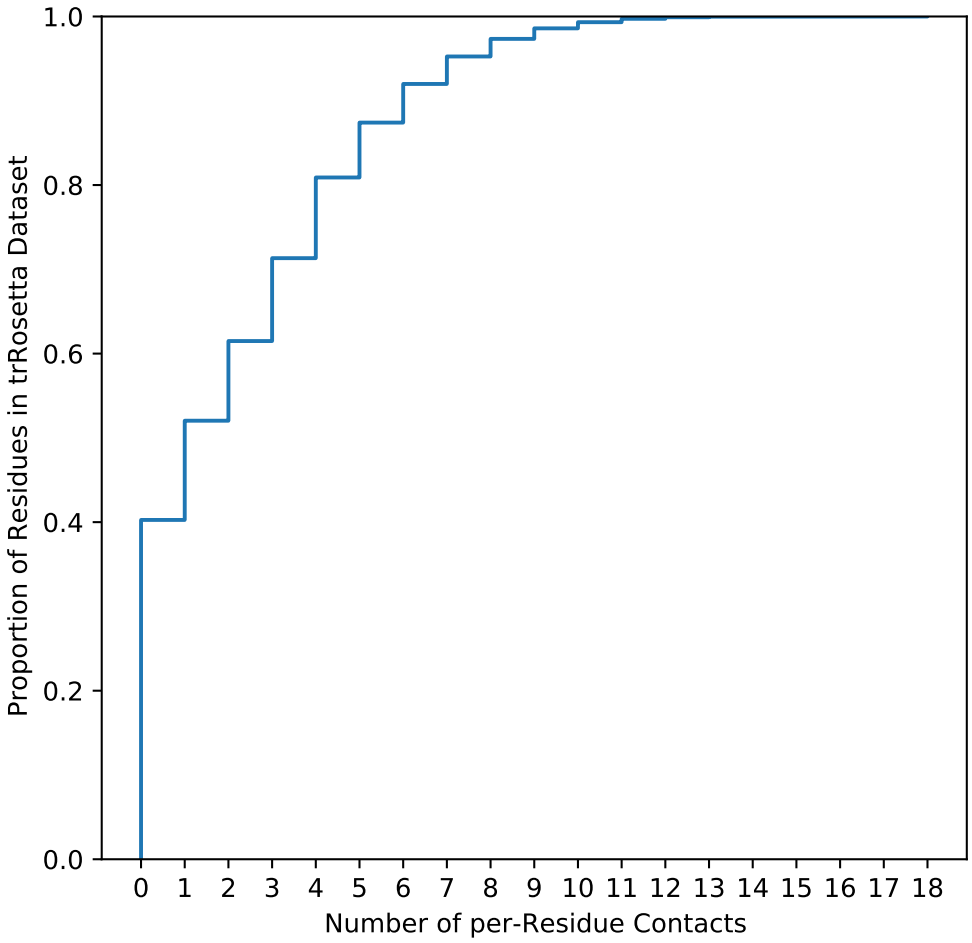


Figure B.2: The empirical CDF of number of per-residue contacts for 3,747,101 residues in 15,051 structures in the trRosetta dataset.

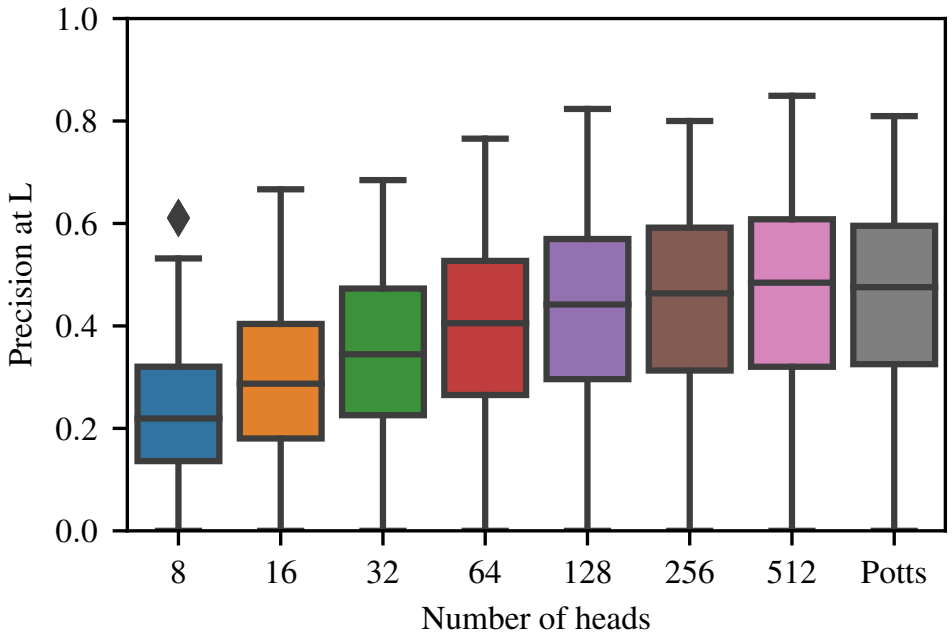


Figure B.3: Reducing the number of heads causes a much steeper decrease in precision at L.

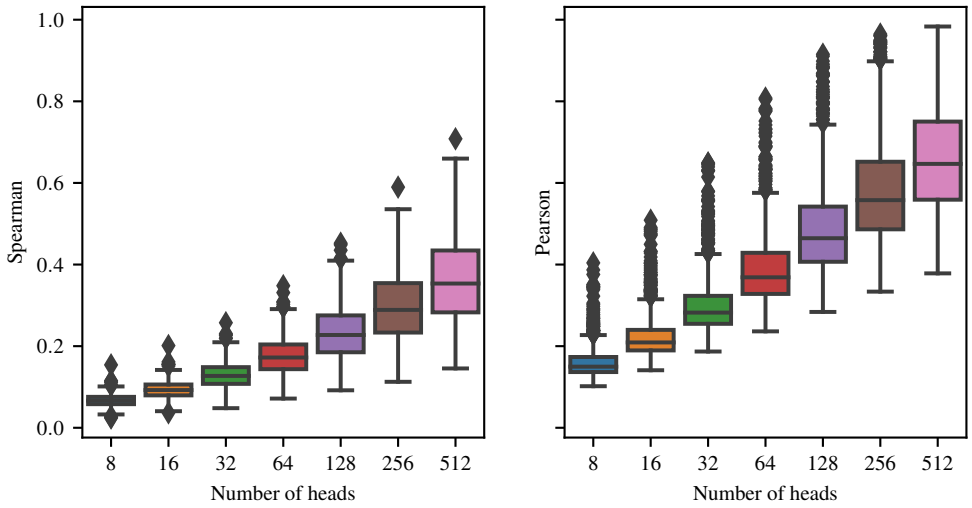


Figure B.4: Effect of number of heads on correlation between the order-4 weight tensors for factored attention (see Equation B.2) and Potts (see Section 3.3).

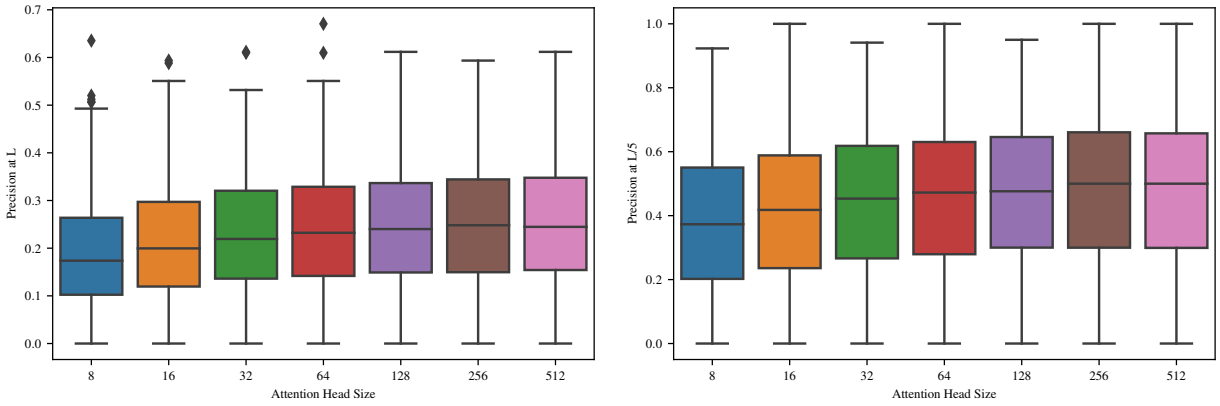


Figure B.5: Effect of head size on factored attention precision at L and $L/5$ over 748 families. Increasing head size has a small effect on precision, though not nearly as pronounced as the number of heads.

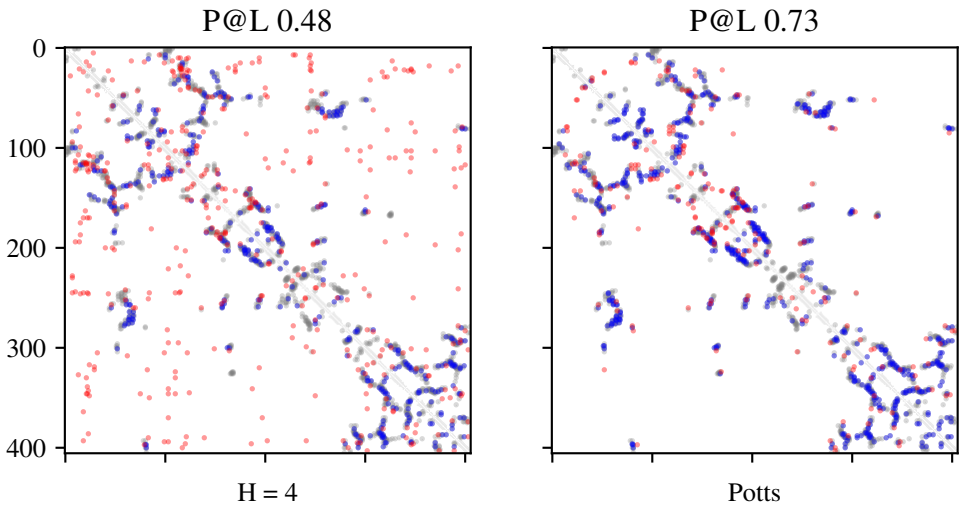


Figure B.6: Factored attention with 4 heads has degraded performance for precision at L for family *3n2a*.

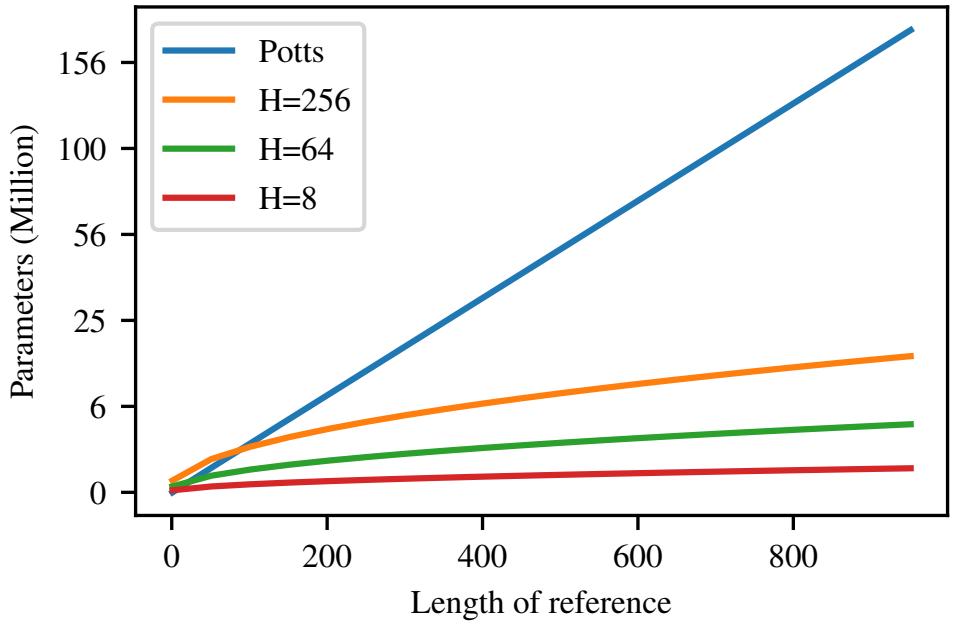


Figure B.7: Number of parameters versus length for MRF models.

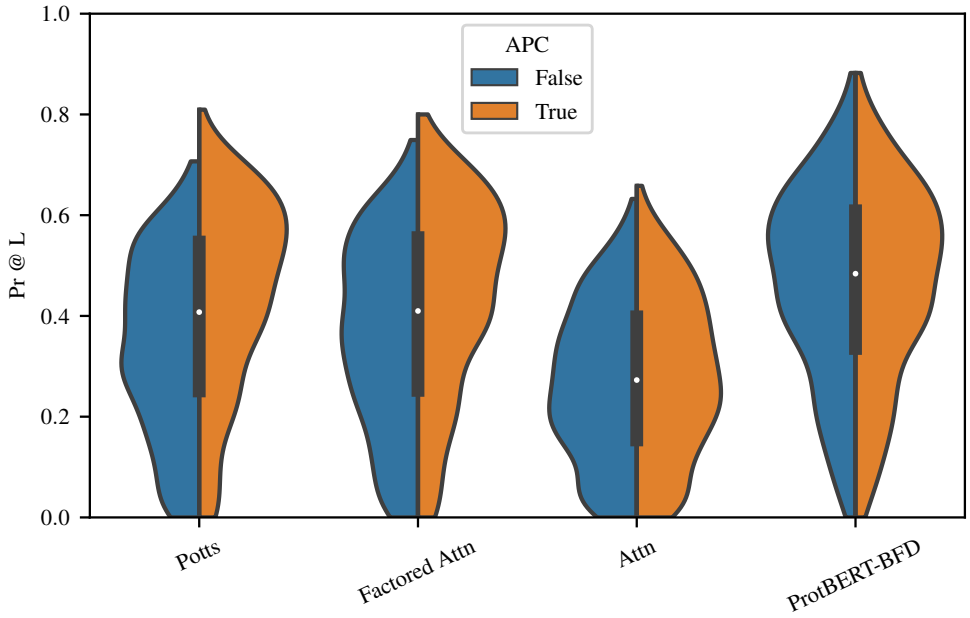


Figure B.8: APC has a significant positive effect on the performance of Potts and factored attention. It makes only a slight difference on the performance of the other two models.

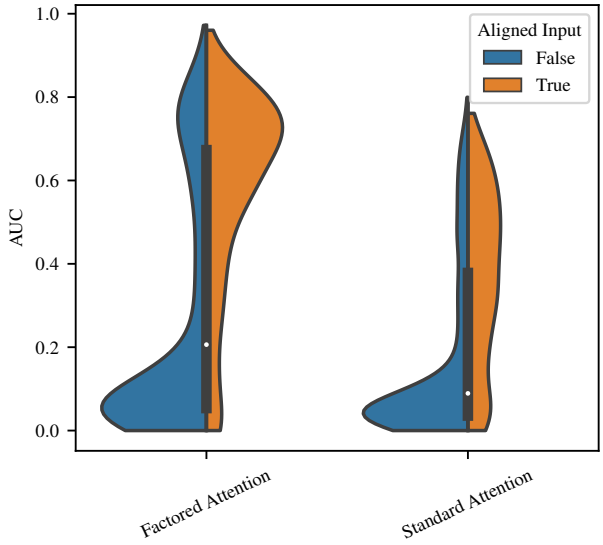


Figure B.9: Training on unaligned families degrades performance on almost all families.

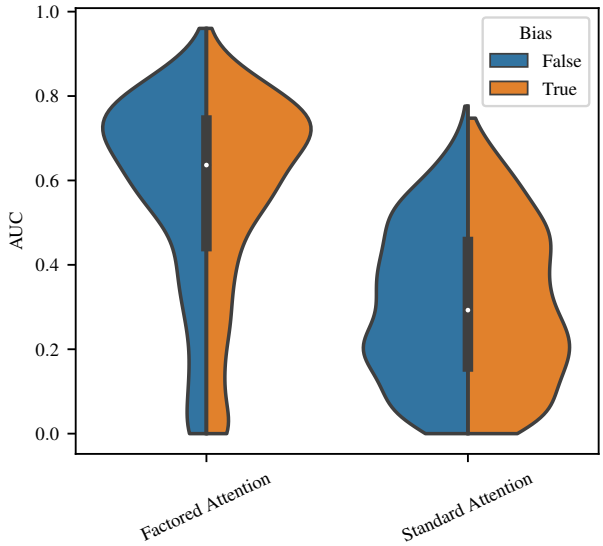


Figure B.10: The addition of a single-site term to either factored or standard attention produces little additional benefit.

Appendix C

Tuned Fitness Landscapes for Benchmarking Model Guided Protein Design

C.1 Potts Models and Evaluation Sets

Landscape Details

PDB	Length	# Seqs	Prec. @ L	Epistatic Horizon	α_{s_+}	p_ϵ	$\bar{\epsilon}_{+,+}$	$\sigma(s)$
3bfo	85	12153	0.78	-72.3	0.0099	0.42	0.013	2.22
3er7	118	33672	0.71	166.6	0.036	0.51	-0.0087	2.15
3my2	126	5497	0.82	-358.4	0.011	0.48	0.0049	3.01
5hu4	145	6440	0.81	-389.3	0.016	0.36	0.0052	2.87
3gfb	347	13554	0.76	-689.7	0.041	0.46	0.0027	2.91

Table C.1: Untuned landscape details. Contact precision is computed in the standard way: predicting the top L entries (>6 apart in the primary sequence) in **H** to be contacts, and computing precision [175, 176]. α_{s_+} refers to the fraction of adaptive singles with effect $s_{i\beta} > 0$. p_ϵ refers to the fraction of reciprocal sign epistasis for pairs of adaptive singles. $p_\epsilon = \frac{\#\{\epsilon_{+,+} < 0\}}{\#\{\epsilon_{+,+}\}}$

PDB	Functional Keywords
3bfo	Immunoglobulin-like beta sandwich
3er7	Nuclear transport factor
3my2	Transmembrane protein
5hu4	Prokaryotic Sortase
3gfb	L-threonine Dehydrogenase

Table C.2: Functional keywords associated with the selected PDBs.

Fitting Potts Models

The initial training of the Potts model involves sampling batches from an alignment X. We train the model to maximize the regularized pseudolikelihood objective

$$Loss(\mathbf{h}, \mathbf{H}; X) = \mathcal{L}(\mathbf{h}, \mathbf{H}; X) + R(\mathbf{h}, \mathbf{H}),$$

where the regularization term is given by

$$R(\mathbf{h}, \mathbf{H}) = \frac{1}{2}\lambda AL||\mathbf{H}||_F^2 + \lambda||\mathbf{h}||_F^2,$$

following the scaling procedure in [172]. The Potts model is trained using a modified version of Adam [162] presented in Dauparas, et al. [173], modified to improve performance of Adam to match that of L-BFGS, using batches X_b from the overall alignment X. Before computing any forward passes, we symmetrize $\mathbf{H}$ and mask the diagonal. All models were trained using $\lambda = 0.01$, Adam learning rate 0.5, and batch size 128. Training was done for 200 steps on a NVIDIA GeForceRTX 2080 Ti GPU. The training script can be found at `https://github.com/songlab-cal/factored-attention`. Models are trained using the "use-bias" flag to explicitly include $\mathbf{h}$. From the 748 potts models trained in Bhattacharya et al. [74], the 5 PDB ids listed in Table C.1 were selected manually from the top performing models with respect to contact precision @ L. Note that the deeper the alignment, the more robust the Potts model training is to hyperparameter choices.

Ridge Hyperparameter	Grid
L2 penalty (α)	10^x for $x \in \{-3, -2.5, -2, ..., 1.0, 1.5, 2.0\}$

Table C.3: Tuned hyperparameters for the Ridge model. (Grid size: 11)

CNN Hyperparameter	Grid
Learning Rate (Adam)	10^x for $x \in \{-3, -2.9, -2.8, ..., 2.0\}$
Batch Size	$\{64, 128\}$
Num Training Epochs	$\{100, 500, 1000\}$
Num filters	$\{16, 32, 64\}$

Table C.4: Tuned hyperparameters for the CNN model. (Grid size: 198)

CNN Hyperparameter	Fixed Value
Kernel Size	5
Hidden Size	64
Activation Function	ReLU
Dropout probability	25%

Table C.5: Fixed hyperparameters for the CNN model.

Epistasis-enriched evaluation sets

In Section 4.4 we describe an evaluation set based on combining multiple adaptive singles into multi-mutants. In this section we describe two additional evaluation sets for each landscape based on enrichment for epistasis. The motivation for selecting variants with large magnitude epistatic effects is to confound the linear model, and test the nonlinear model's ability to learn epistasis from a training set of multi-mutants. We build two evaluation sets: Adaptive Epistasis and Deleterious Epistasis. For both sets the procedure for constructing them is the same, but with the sign of the epistatic terms inverted.

We construct the full $L \times L \times A \times A$ tensor of epistatic terms $\epsilon_{i\beta,j\gamma}$. Ranking these terms by their value, we pick out the highest 1000 terms (for Deleterious Epistasis, we pick out the lowest 1000). From this set of strong epistatic interactions, we construct a pool of site-allele pairs: $M = \{(i\beta_1, j\gamma_1), \ldots, (i\beta_{1000}, j\gamma_{1000})\}$. From this pool of pairs M we construct variants at the desired distance from the wildtype. For an evaluation set at distance 6, we choose 3 pairs at random and combine them. We continue to sample combinations of epistatic pairs until we have an evaluation set of the desired size $n = 200$. Note that some site-allele pairs conflict with one another by mutating the same position. We discard combinations that do not reach the desired distance.

C.2 Quadratic Landscape Theory

In the following, we develop the basic theory for quadratic landscapes in detail. We will derive a tuning scheme which allows us to separately control the distribution of single mutant fitness effects double mutant fitness effects. This will allow us to *tune* landscapes in order to explore different regimes of the overall optimization space.

We will be interested in understanding fitness landscapes defined on sequence space. We will consider sequences of length L on A characters, encoded by vectors $\mathbf{x}$ of length LA, which are one-hot every A elements. (In computational settings, the sequence is often represented as an $L \times A$ matrix, one-hot in the second index.)

Quadratic fitness function

Consider a fitness function $\mathcal{F}$ given by

$$\mathcal{F}(\mathbf{x}) = \mathbf{h}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x}. \tag{C.1}$$

Here $\mathbf{H}$ is an $LA \times LA$ symmetric coupling matrix, and $\mathbf{h}$ is a length LA vector. Note that in computational settings, $\mathbf{H}$ is often implemented as a tensor of dimension $L \times L \times A \times A$, and $\mathbf{h}$ as a tensor of dimension $L \times A$.

We note that in biological applications, we generally care about fitness differences; fitness functions which differ by a constant value are considered to be the same.

Given the L-hot structure of $\mathbf{x}$, we see that the L distinct $A \times A$-dimensional subblocks of $\mathbf{H}$ corresponding to within-site interactions are special. In particular, only the diagonal terms contribute to $\mathcal{F}$, since $\mathbf{x}$ is one-hot within a block. Due to this one-hot structure, without loss of generality we can absorb the within-site interactions into the linear term by setting $\hat{\mathbf{h}}_{i\alpha} = \mathbf{h}_{i\alpha} + \frac{1}{2}\mathbf{H}_{i\alpha,i\alpha}$, and $\hat{\mathbf{H}}_{i\alpha,i\beta} = 0$ for each site i and characters α and β, and $\hat{\mathbf{H}} = \mathbf{H}$ otherwise. The functional form of our fitness is unchanged:

$$\mathcal{F}(\mathbf{x}) = \hat{\mathbf{h}}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \hat{\mathbf{H}}\mathbf{x}. \tag{C.2}$$

Thus, for the remainder of the notes we guarantee, without loss of generality, that in addition to being symmetric, $\mathbf{H}$ has 0 diagonal, i.e. $\mathbf{H}_{i\alpha,i\beta} = 0$.

Local statistics

We are interested in the statistics near a particular sequence $\mathbf{x}_0$, which we call the "wildtype" sequence. For example, in enzyme design, we often start with a wildtype sequence which can be used in a reaction of interest, and the goal of the optimization is to arrive at a designed sequence which carries out the reaction more efficiently.

Without loss of generality, for the remainder of the notes we will refer to the wildtype sequence with a generic character $\mathbf{x}_0(i) = a$ at all positions i. We often consider the relative

fitness $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0)$ rather than the absolute fitness $\mathcal{F}(\mathbf{x})$. The quadratic model is defined by two quantities: the single mutant fitness effects s and the *pairwise epistasis* ϵ. The single mutant fitness effects are defined by $s(\mathbf{x}_1) = \mathcal{F}(\mathbf{x}_1) - \mathcal{F}(\mathbf{x}_0)$, where $\mathbf{x}_1$ is a single mutant which differs from $\mathbf{x}_0$ in exactly one of the L positions. We can write out the effect explicitly. Let $\mathbf{x}_0(i)$ denote the character at position i in the wildtype sequence $\mathbf{x}_0$. Consider a mutation $s_{i\beta}$ at site i, which takes character $\mathbf{x}_0(i) = a$ to character β. We have:

$$s_{i\beta} = \mathbf{h}_{i\beta} - \mathbf{h}_{ia} + \sum_{j=1}^{L} \left(\mathbf{H}_{i\beta,ja} - \mathbf{H}_{ia,ja}\right). \tag{C.3}$$

In many cases, these linear effects are enough to begin to design sequences and optimize over the fitness landscape. Note that this linear structure depends on both $\mathbf{h}$ and $\mathbf{H}$.

The higher order interactions can be quantified using *pairwise epistasis*. In general, the term epistasis is used by geneticists to refer to interaction between the effects of multiple mutations. There are many ways to quantify these interactions. We focus on a definition of pairwise epistasis which measures the deviation from linearity of a landscape.

Given a double mutant $\mathbf{x}_{12}$, with single mutant sequences given by $\mathbf{x}_1$ and $\mathbf{x}_2$, we define the *pairwise epistasis* $\epsilon(\mathbf{x}_{12})$ by

$$\epsilon(\mathbf{x}_{12}) = \mathcal{F}(\mathbf{x}_{12}) - \mathcal{F}(\mathbf{x}_1) - \mathcal{F}(\mathbf{x}_2) + \mathcal{F}(\mathbf{x}_0). \tag{C.4}$$

This definition can be motivated by re-writing as

$$\begin{aligned}\epsilon(\mathbf{x}_{12}) &= \mathcal{F}(\mathbf{x}_{12}) - \mathcal{F}(\mathbf{x}_0) - (\mathcal{F}(\mathbf{x}_1) - \mathcal{F}(\mathbf{x}_0) + \mathcal{F}(\mathbf{x}_2) - \mathcal{F}(\mathbf{x}_0)) \\ &= \mathcal{F}(\mathbf{x}_{12}) - \mathcal{F}(\mathbf{x}_0) - (s(\mathbf{x}_1) + s(\mathbf{x}_2)).\end{aligned}$$

In other words, it's the part of the fitness difference between $\mathbf{x}_{12}$ and $\mathbf{x}_0$ which can't be explained by the single mutants $\mathbf{x}_1$ and $\mathbf{x}_2$.

If the two mutations are $a \to \beta$ at site i and $a \to \gamma$ at site j, then the epistasis $\epsilon_{i\beta,j\gamma}$ is given by

$$\epsilon_{i\beta,j\gamma} = \mathbf{H}_{i\beta,j\gamma} - \mathbf{H}_{i\beta,ja} - \mathbf{H}_{ia,j\gamma} + \mathbf{H}_{ia,ja}. \tag{C.5}$$

Difference expansion

It is useful to explicitly write the fitness difference $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0)$ for some general $\mathbf{x}$, in order to understand and manipulate local statistics. Let $M = \{m_1, m_2, ...m_k\}$ be the sequence of k mutations from wildtype in $\mathbf{x}$, where $m_l = (i_l, \beta_l)$. No two mutations affect the same position, so $i_l \neq i_{l'}$ for $l \neq l'$. We have

$$\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0) = \mathbf{h}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} - \mathbf{h}^\top \mathbf{x}_0 - \frac{1}{2}\mathbf{x}_0^\top \mathbf{H}\mathbf{x}_0 \tag{C.6}$$

We can rewrite this in terms of the sequence difference $\boldsymbol{\delta} \equiv \mathbf{x} - \mathbf{x}_0$. We have

$$\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0) = (\mathbf{h}^\top + \mathbf{x}_0^\top \mathbf{H})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{H}\boldsymbol{\delta}. \tag{C.7}$$

The first term captures linear effects with respect to $\boldsymbol{\delta}$, and the second term captures quadratic effects. By comparing the first term to equation C.3, we can see that it in fact corresponds to the sum of the single mutant fitness effects:

$$(\mathbf{h}^\top + \mathbf{x}_0^\top \mathbf{H})\boldsymbol{\delta} = \sum_{(i,\beta)\in M} s_{i\beta}. \tag{C.8}$$

The second term is related to the epistatic effects, which we show explicitly. We have

$$\boldsymbol{\delta}^\top \mathbf{H} \boldsymbol{\delta} = \sum_{(i,\beta),(j,\gamma)\in M^2} \mathbf{H}_{i\beta,j\gamma} - \mathbf{H}_{i\beta,ja} - \mathbf{H}_{ia,j\gamma} + \mathbf{H}_{ia,ja}, \tag{C.9}$$

where M^2 refers to ordered pairs of mutations drawn from M. For $i \neq j$,

$$\mathbf{H}_{i\beta,j\gamma} - \mathbf{H}_{i\beta,ja} - \mathbf{H}_{ia,j\gamma} + \mathbf{H}_{ia,ja} = \epsilon_{i\beta,j\gamma} \tag{C.10}$$

$$\boldsymbol{\delta}^\top \mathbf{H} \boldsymbol{\delta} = \sum_{(i,\beta)\neq(j,\gamma)\in M^2} \epsilon_{i\beta,j\gamma} + \sum_{(i,\beta)\in M} \left(\mathbf{H}_{i\beta,i\beta} - 2\mathbf{H}_{i\beta,ia} + \mathbf{H}_{ia,ia}\right). \tag{C.11}$$

We showed that without loss of generality, we can reparameterize $\mathbf{H}$ and $\mathbf{h}$ so that $\mathbf{H}$ has no diagonal terms. Assume this is the case. Then, we can write

$$\frac{1}{2}\boldsymbol{\delta}^\top \mathbf{H} \boldsymbol{\delta} = \frac{1}{2} \sum_{(i,\beta)\neq(j,\gamma)\in M^2} \epsilon_{i\beta,j\gamma}. \tag{C.12}$$

So we have shown that the second term in Equation C.7 is the sum of the pairwise epistatis effects for all pairs of mutations in $\mathbf{x}$ relative to the wildtype $\mathbf{x}_0$.

The expansion in Equation C.7 is useful for two reasons. Theoretically, it shows that the single mutant effects and epistatic effects control fitness differences completely, and gives us an easy way to compute them:

$$\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0) = \sum_{(i,\beta)\in M} s_{i\beta} + \frac{1}{2} \sum_{(i,\beta)\neq(j,\gamma)\in M^2} \epsilon_{i\beta,j\gamma}. \tag{C.13}$$

The practical consequence is that we can use the decomposition to separately manipulate the single mutant and epistasis properties of the landscape, as we will discuss in Section C.2.

Tuning landscapes

In order to benchmark and understand methods for exploring fitness landscapes, we want to test those methods on fitness landscapes with variable properties. In particular, given some fitness function $\mathcal{F}$ of interest, we are interested in modifications of $\mathcal{F}$ which make the problem "easier" or "harder" by some metric.

For a quadratic $\mathcal{F}$, a set of simple modifications is given by shifting and scaling the distribution of fitnesses of single and double mutants relative to the wildtype. In particular, we can independently shift (add a constant to) and scale (multipy by a constant) the single mutant statistics s and the epistasis statistics ϵ uniformly for all sequences.

As we will see, this is different from simply modifying $\mathbf{h}$ and $\mathbf{H}$. Modifying s and ϵ corresponds to modifying the landscape in terms of first and second order expansions around the wildtype $\mathbf{x}_0$. In many biological problems, we care about understanding behavior near the wildtype; in addition, inferred landscape (e.g. using DCA [15]) are likely correlated with the "true" fitness landscape in limited neighborhood of the wildtype.

The shifting and scaling approach we outline maintains the relative ordering of fitnesses within the single mutants and within the double mutants. If we start with a fitness landscape whose properties are relevant for optimization, the modified landscape is one which has some similar qualitative features (e.g. important interactions in the original landscape are important in the tuned landscape). The modified landscapes can also probe different questions, such as "What happens when epistasis is more important than single mutant effects?"

We note that in most applications, we only care about the *relative* values of $\mathcal{F}$ (e.g. $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0)$), rather than the absolute values. We will take that approach here. If the absolute value also matters, for example if $\mathcal{F}(\mathbf{x}_0)$ needs to be set to 0, then this can be accomplished by adding the appropriate constant to $\mathcal{F}$.

Shifting the single-mutant distribution.

Suppose we wish to shift the distribution of single mutant fitness effects, relative to wildtype, by some constant μ_s, without modifying ϵ. This can be accomplished by modifying $\mathbf{h}$ such that $\tilde{\mathbf{h}} = \mathbf{h} + \mathbf{v}$. Given $\mathcal{F}$ with parameters $\mathbf{h}$ and $\mathbf{H}$, we define $\tilde{\mathcal{F}}$ as

$$\tilde{\mathcal{F}}(\mathbf{x}) = (\mathbf{h} + \mathbf{v})^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} \tag{C.14}$$

Using the expansion in Equation C.7, we have

$$\tilde{\mathcal{F}}(\mathbf{x}) - \tilde{\mathcal{F}}(\mathbf{x}_0) = (\mathbf{h} + \mathbf{v} + \mathbf{x}_0\mathbf{H})^\top \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{H}\boldsymbol{\delta} \tag{C.15}$$

We know that the first term controls s and the second controls ϵ. Therefore, with the appropriate choice of $\mathbf{v}$, we can modify s without modifying ϵ.

Let $\mathbf{v} = -\mu_s \mathbf{x}_0$. We note that $(\mathbf{x}_0^\top \boldsymbol{\delta})_{i\beta} = -1$ when $\mathbf{x}$ has a mutation $a \to \beta$ at position i, and 0 otherwise. Then we have:

$$\begin{aligned}\tilde{s}_{i\beta} &= \mathbf{h}_{i\beta} - \mathbf{h}_{ia} + \mu_s + \sum_{j=1}^{L} (\mathbf{H}_{i\beta,ja} - \mathbf{H}_{ia,ja}) \\ &= s_{i\beta} + \mu_s\end{aligned}$$

which corresponds exactly to the desired shift.

We note that the choice of $\mathbf{v}$ is not unique, since the quadratic form of $\mathcal{F}$, coupled with the gauge symmetry induced by the structured L-hot nature of of $\mathbf{x}$ means that the constant function can be written in many different ways. For example, the shift $\tilde{\mathbf{h}} = \mathbf{h} + \frac{c}{L}\mathbf{1}$ is equivalent to adding a constant c to the fitness function. This lack of uniqueness is not a problem computationally because our chosen form for $\mathbf{v}$ achieves the desired s and ϵ distributions - which are all that's needed to define $\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0)$.

Shifting the epistatic distribution.

Shifting the epistatic distribution is more complicated. From Equation C.7, we see that modifying $\mathbf{H}$ affects both the epistasis distribution as well as the single-mutant distribution. Therefore, we will modify both $\mathbf{H}$ and $\mathbf{h}$ in order to modify ϵ without changing any of the s.

We shift $\tilde{\mathbf{H}} = \mathbf{H} + \mathbf{C}$ and $\tilde{\mathbf{h}} = \mathbf{h} + \mathbf{w}$. Since $\tilde{\mathbf{H}}$ and $\mathbf{H}$ are symmetric and have 0 diagonal, $\mathbf{C}$ must be symmetric and have 0 diagonal. Our desired modified fitness function $\tilde{\mathcal{F}}$ therefore, is:

$$\tilde{\mathcal{F}}(\mathbf{x}) - \tilde{\mathcal{F}}(\mathbf{x}_0) = \left(\mathbf{h} + \mathbf{w} + \mathbf{x}_0(\mathbf{H} + \mathbf{C})\right)^\top \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top(\mathbf{H} + \mathbf{C})\boldsymbol{\delta} \tag{C.16}$$

which has the same s as $\mathcal{F}$, but all ϵ shifted by μ_ϵ. We proceed by deriving $\mathbf{C}$ to modify ϵ, and then compute $\mathbf{w}$ to ensure there is no change in s.

From Equation C.5, we see that one way to change the epistasis is to modify the (ia, ja) terms in $\mathbf{H}$, and no others. This suggests that $\mathbf{C}$ should be proportional to $\mathbf{x}_0\mathbf{x}_0^\top$, which is equal to 1 at (ia, ja) and 0 otherwise. We define

$$\mathbf{C}_{i\beta,j\gamma} = \begin{cases} \mu_\epsilon & \text{for } \; i \neq j, \; \beta = \gamma = a \\ 0 & \text{otherwise} \end{cases} \tag{C.17}$$

In other words,

$$\mathbf{C} = \mu_\epsilon \left(\mathbf{x}_0\mathbf{x}_0^\top - \text{diag}\left(\mathbf{x}_0\mathbf{x}_0^\top\right)\right)$$

Computing the epistasis for the modified $\tilde{\mathbf{H}} = \mathbf{H} + \mathbf{C}$ using Equation C.5, we have:

$$\begin{aligned} \tilde{\epsilon}_{i\beta,j\gamma} &= \mathbf{H}_{i\beta,j\gamma} - \mathbf{H}_{i\beta,ja} - \mathbf{H}_{ia,j\gamma} + \mathbf{H}_{ia,ja} + \mu_\epsilon \\ &= \epsilon_{i\beta,j\gamma} + \mu_\epsilon \end{aligned}$$

which gives us the intended shift. To ensure that s is unchanged, we set

$$\mathbf{h} + \mathbf{w} + (\mathbf{H} + \mathbf{C})\mathbf{x}_0 = \mathbf{h} + \mathbf{H}\mathbf{x}_0$$

Which gives us $\mathbf{w} = -\mathbf{C}\mathbf{x}_0$. Note that $\mathbf{x}_0^\top\mathbf{x}_0 = L$ and $\text{diag}(\mathbf{x}_0\mathbf{x}_0^\top)\mathbf{x}_0 = \mathbf{x}_0$. Then,

$$\begin{aligned} \mathbf{w} &= -\mathbf{C}\mathbf{x}_0 \\ &= -\mu_\epsilon\mathbf{x}_0\mathbf{x}_0^\top\mathbf{x}_0 + \mu_\epsilon\text{diag}\left(\mathbf{x}_0\mathbf{x}_0^\top\right)\mathbf{x}_0 \\ &= -L\mathbf{x}_0 + \mathbf{x}_0 \\ &= -\mu_\epsilon(L-1)\mathbf{x}_0 \end{aligned}$$

Setting $\mathbf{w} = -\mu_\epsilon(L-1)\mathbf{x}_0$, s is left unchanged as desired.

Scaling the distributions.

Now we consider the problem of *scaling* the distributions. That is, we want to modify $\mathbf{h}$ and $\mathbf{H}$ such that s and ϵ are multiplied uniformly by constants λ_s and λ_ϵ respectively. Using the difference expansion in equation C.7, we can see that to accomplish this we need to choose constants A, B and vector $\mathbf{u}$ such that:

$$B\mathbf{H} = \lambda_\epsilon \mathbf{H} \tag{C.18}$$

and

$$A\mathbf{h} + \mathbf{u} + B\mathbf{H}\mathbf{x}_0 = \lambda_s(\mathbf{h} + \mathbf{H}\mathbf{x}_0). \tag{C.19}$$

We immediately see that $B = \lambda_\epsilon$. In order to obtain the correct scaling of s, we have:

$$A\mathbf{h} + \mathbf{u} + B\mathbf{H}\mathbf{x}_0 = \lambda_s(\mathbf{h} + \mathbf{H}\mathbf{x}_0). \tag{C.20}$$

With our extra degree of freedom, we choose to set $A = \lambda_s$, so we have:

$$\mathbf{u} = (\lambda_s - \lambda_\epsilon)\mathbf{H}\mathbf{x}_0. \tag{C.21}$$

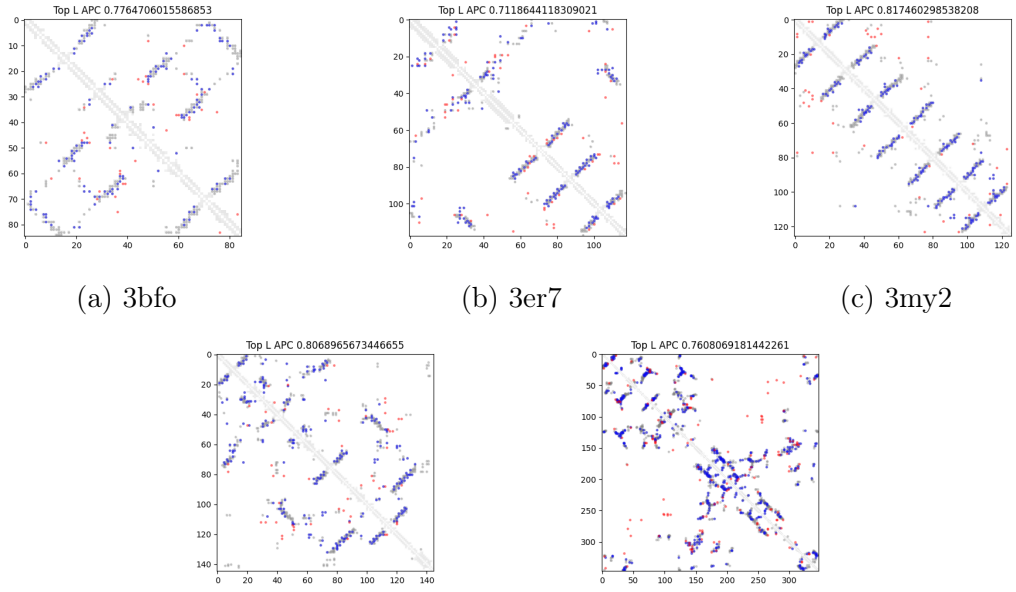
Our final fitness function is therefore

$$\tilde{\mathcal{F}}(\mathbf{x}) - \tilde{\mathcal{F}}(\mathbf{x}_0) = (\lambda_s \mathbf{h} + (\lambda_s - \lambda_\epsilon)\mathbf{H}\mathbf{x}_0)^\top \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top(\lambda_\epsilon \mathbf{H})\boldsymbol{\delta}. \tag{C.22}$$

Each of the modifications outlined in Sections C.2, C.2 and C.2 can be composed:

$$\tilde{\mathcal{F}}(\mathbf{x}) - \tilde{\mathcal{F}}(\mathbf{x}_0) = \lambda_s \sum_{(i,\beta)\in M} [s_{i\beta} + \mu_s] + \frac{\lambda_\epsilon}{2} \sum_{(i,\beta)\neq(j,\gamma)\in M^2} [\epsilon_{i\beta,j\gamma} + \mu_\epsilon]\,. \tag{C.23}$$

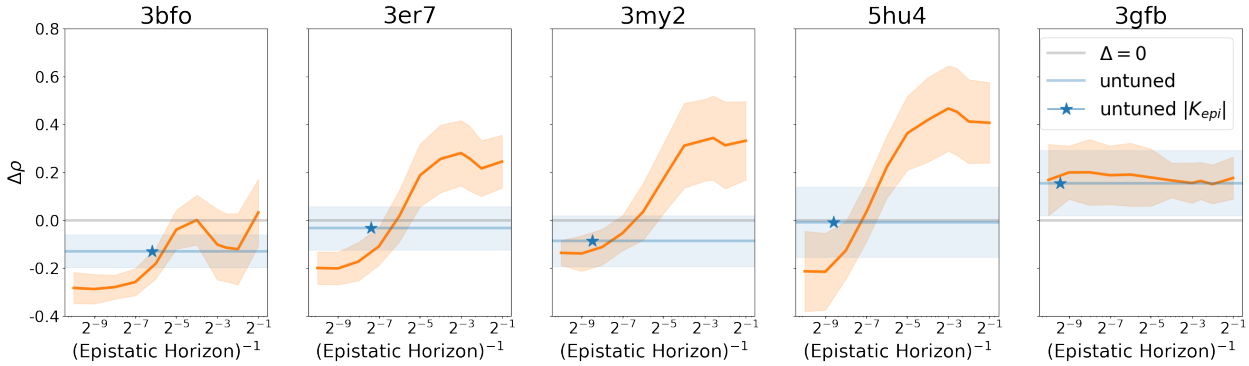
C.3 Supplemental Figures



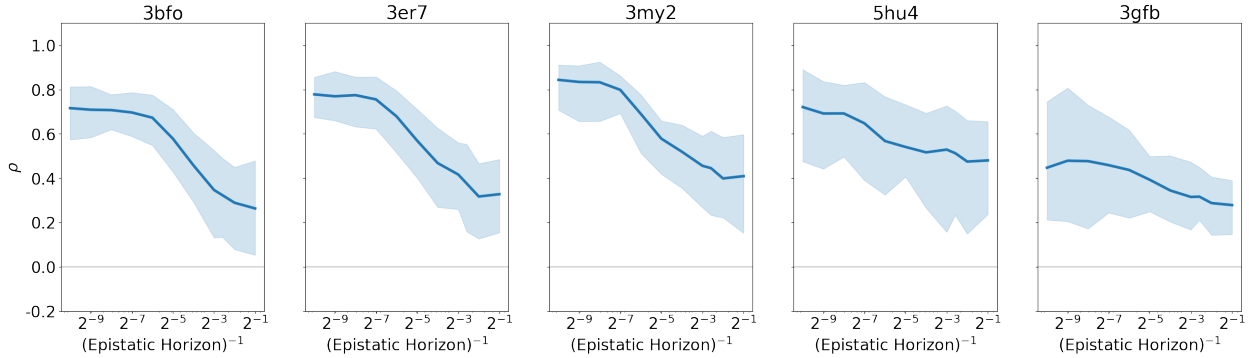
(a) 3bfo (b) 3er7 (c) 3my2

(d) 5hu4 (e) 3gfb

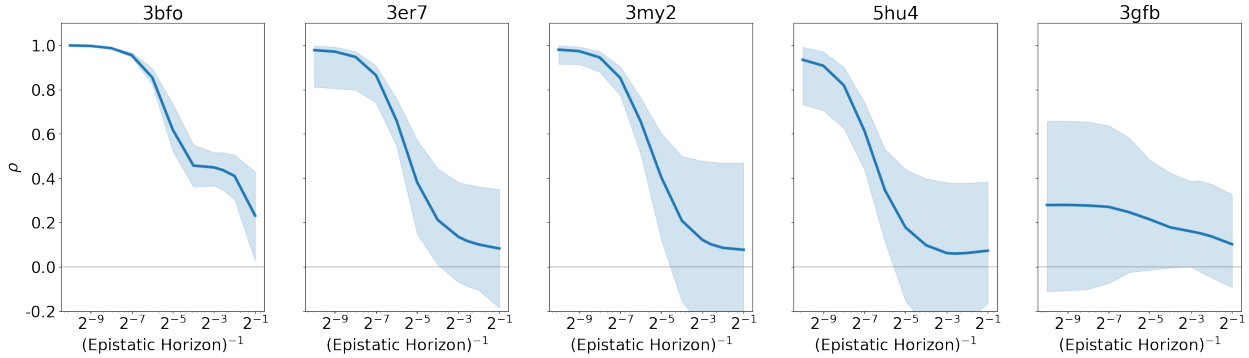
Figure C.1: Predicted contact maps derived from Potts models fit to the corresponding alignment. Shown are Top L predicted contacts after APC correction [177], with the precision shown in the title of each plot. Blue are true predictions, red are false predictions. In grey are all contacts.



(a) Copy of Figure 4.3. Differential performance ($\Delta\rho$) of the CNN versus the Ridge model on ranking combinations of adaptive singles.

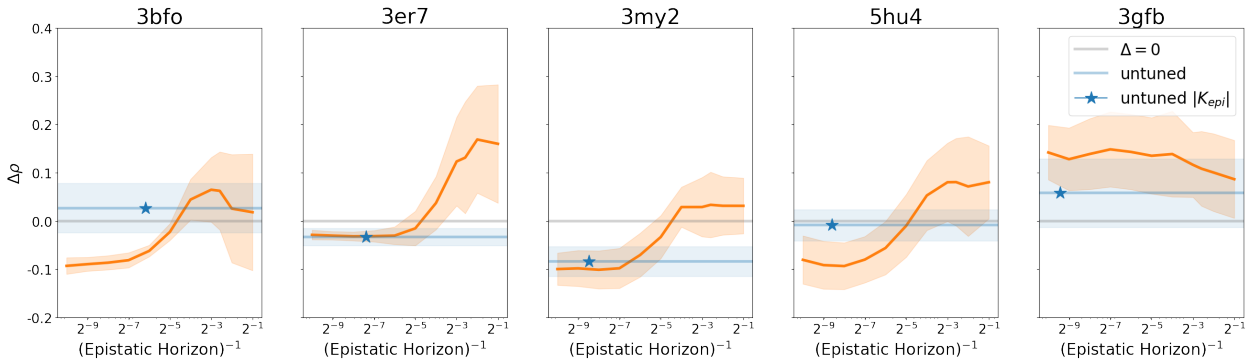


(b) Performance (ρ) of the CNN model on ranking combinations of adaptive singles.

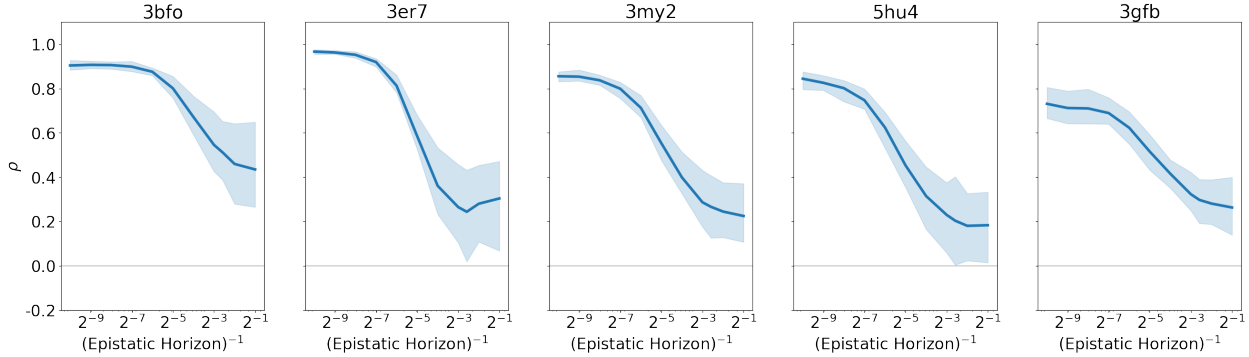


(c) Performance (ρ) of the Ridge model on ranking combinations of adaptive singles. $\rho \approx 1$ for linear landscapes with large horizons.

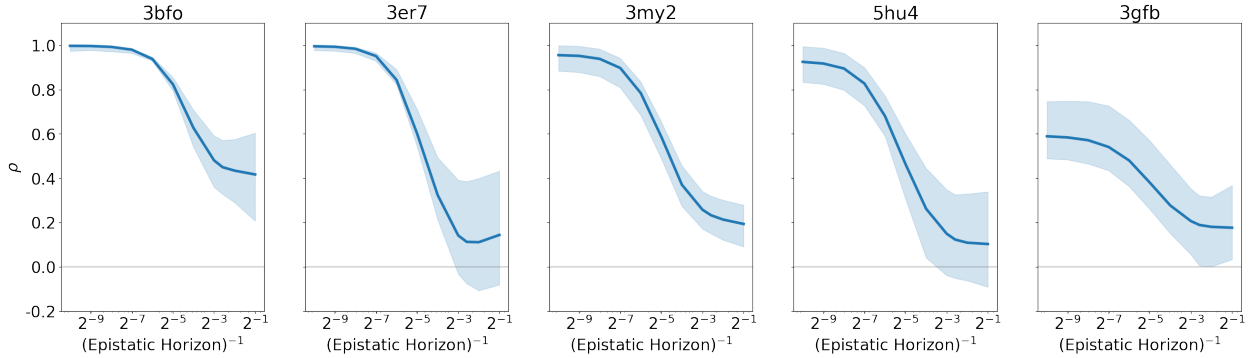
Figure C.2: Performance (ρ) on ranking sequences constructed from combinations of adaptive singles.



(a) Differential performance ($\Delta\rho$) of the CNN versus the Ridge model on ranking sequences enriched for deleterious epistasis

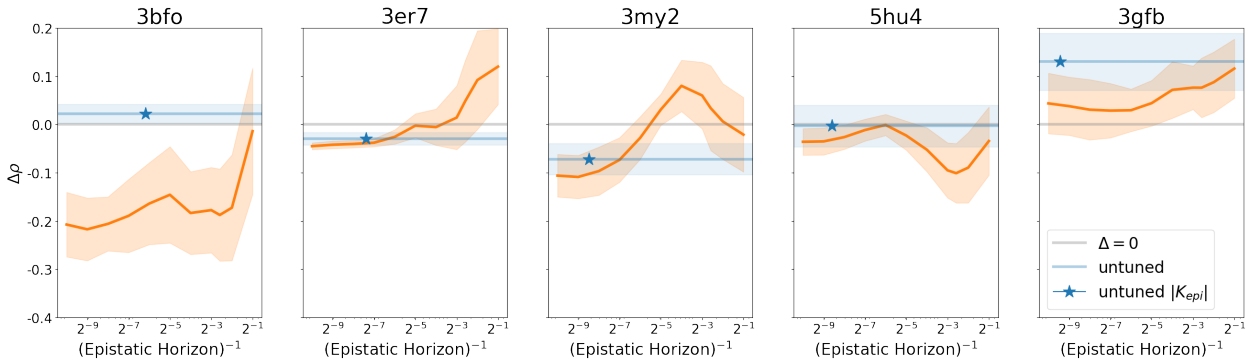


(b) Performance (ρ) of the CNN model on ranking sequences enriched for deleterious epistasis.

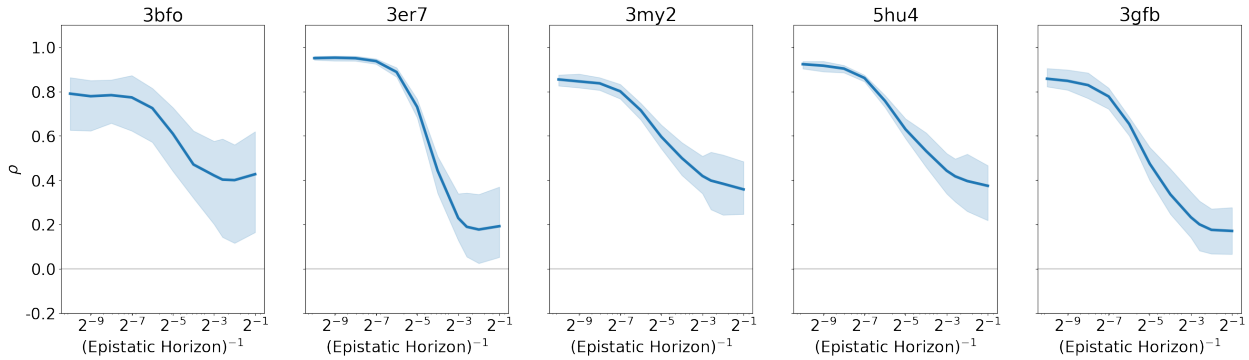


(c) Performance (ρ) of the Ridge model on ranking sequences enriched for deleterious epistasis.

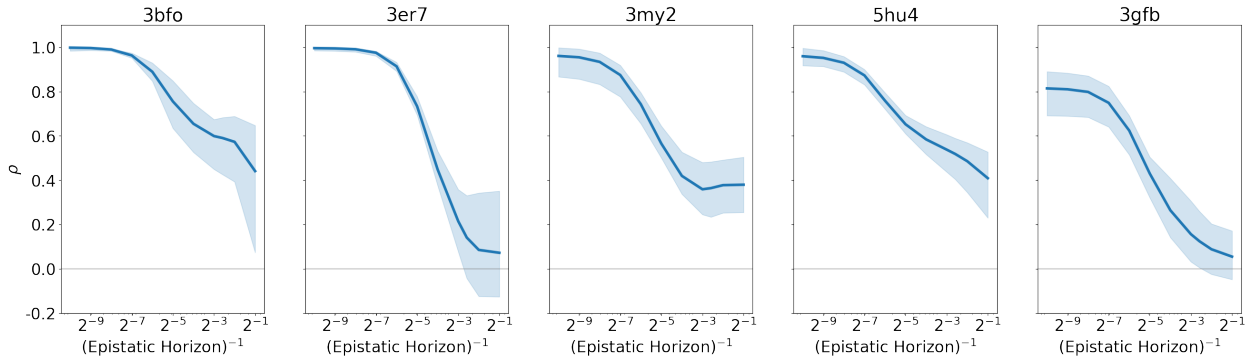
Figure C.3: Performance (ρ) on ranking sequences enriched for deleterious epistasis.



(a) Differential performance ($\Delta\rho$) of the CNN versus the Ridge model on ranking sequences enriched for adaptive epistasis.



(b) Performance (ρ) of the CNN model on ranking sequences enriched for adaptive epistasis.



(c) Performance (ρ) of the Ridge model on ranking sequences enriched for adaptive epistasis.

Figure C.4: Performance (ρ) on ranking sequences enriched for adaptive epistasis.

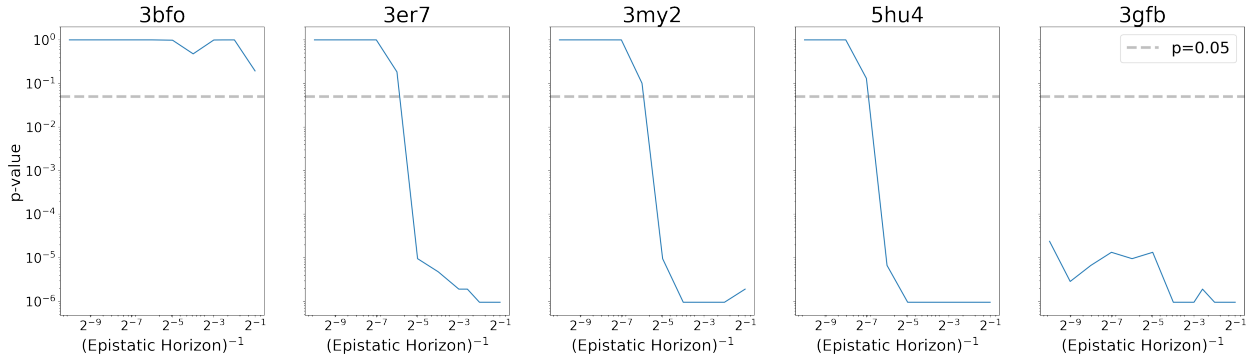


Figure C.5: Evaluation set: Adaptive Singles. One-sided Wilcoxon signed-rank test p-value on $\Delta\rho$ where samples are paired by training set replicates. This tests the null hypothesis that the two models have similar evaluation performance (i.e., the distribution of $\Delta\rho$ is symmetric). The grey line indicates the significance threshold p=0.05. 3er7, 3my2, 5hu4 all exhibit a clear transition point where the distribution of CNN model performance is significantly better than the distribution of Ridge model performance. The CNN significantly outperforms the Ridge model across all horizons for 3gfb.

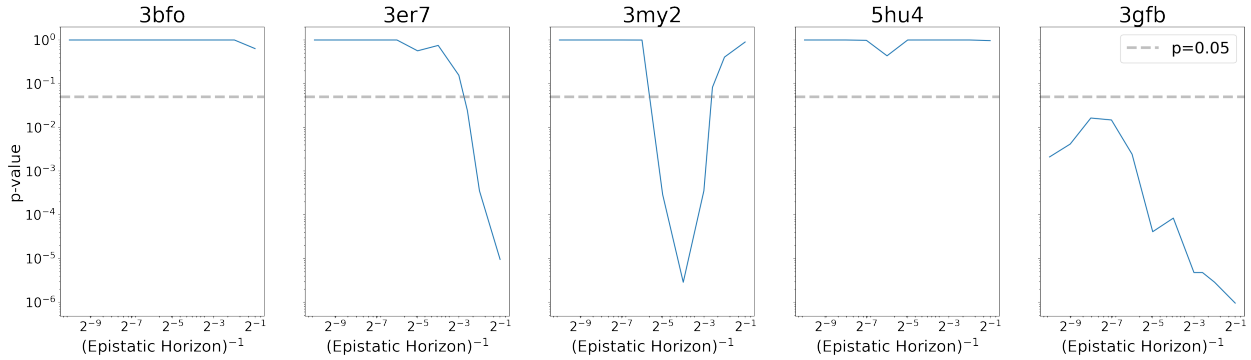


Figure C.6: Evaluation set: Adaptive Epistasis. One-sided Wilcoxon signed-rank test p-value on $\Delta\rho$ where samples are paired by training set replicates. This tests the null hypothesis that the two models have similar evaluation performance (i.e., the distribution of $\Delta\rho$ is symmetric). The grey line indicates the significance threshold p=0.05.

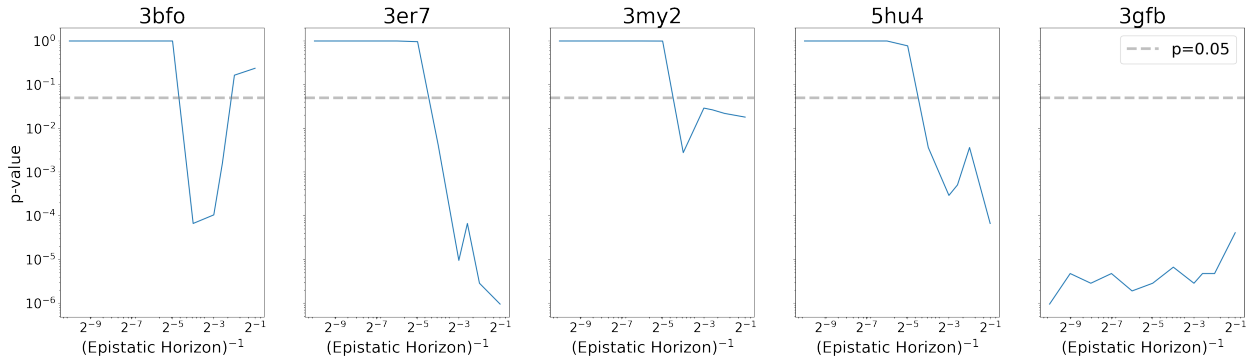


Figure C.7: Evaluation set: Deleterious Epistasis. One-sided Wilcoxon signed-rank test p-value on $\Delta\rho$ where samples are paired by training set replicates. This tests the null hypothesis that the two models have similar evaluation performance (i.e., the distribution of $\Delta\rho$ is symmetric). The grey line indicates the significance threshold p=0.05.

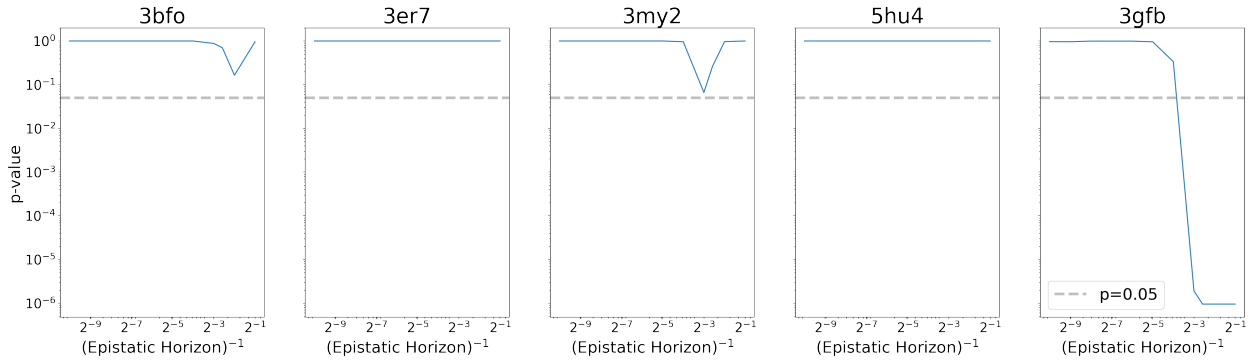


Figure C.8: Evaluation set: Adaptive Singles. One-sided Wilcoxon signed-rank test p-value on ΔMSE where samples are paired by training set replicates. This tests the null hypothesis that the two models have similar evaluation performance (i.e., the distribution of ΔMSE is symmetric). The grey line indicates the significance threshold p=0.05.