Learning Protein Sequence-Function Relationships for Protein Engineering

by

Sam Gelman

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2023

Date of final oral examination: 10/11/23

The dissertation is approved by the following members of the Final Oral Committee:
Anthony Gitter, Associate Professor, Biostatistics and Medical Informatics
Philip Romero, Assistant Professor, Biochemistry
Yingyu Liang, Assistant Professor, Computer Sciences
Yixuan Sharon Li, Assistant Professor, Computer Sciences

To those whose unwavering love and support
helped turn my dream into a reality.

ACKNOWLEDGMENTS

This dissertation is a testament to the incredible support and guidance I have received from countless individuals throughout this journey, and I am deeply grateful to each one of them for helping bring this work to fruition.

I want to thank my wife Hannah, who was with me through every long night, every setback, and every breakthrough. Her love and unwavering support illuminated my path. This achievement is not just mine; it is ours.

I am grateful to my parents Jack and Larisa for their steadfast belief in me and the sacrifices they made so that I could pursue my dream.

For their continuous support and encouragement, I want to thank my siblings Ben, Sasha, and Roman; my grandparents Boris and Galina; Lena, who has been like a family member; and my in-laws Grant, Stephanie, Marty, and Mary Ann. A special thanks to Ben for not only his support and friendship but also for his invaluable help in brainstorming and refining scientific ideas.

I want to thank all my friends, many of whom were fellow graduate students, for supporting me, making life fun, and distracting me from my studies. These include, among many others, Michael Bowen, Brad Boccuzzi, David Merrel, Ben Kaufman, Kylie Moynihan, Aaron Baker, Chris Magnano, Varun Sah, Tuan Dinh, Danny McNeela, Richelle Wilson, Nate Carlin, members of the Gitter and Romero Labs, and members of Kanopy Dance Company.

I want to thank the incredible staff at Sencha Tea Bar, including Holly, Lyla, Ally, Ben, and Jess, for fostering an environment brimming with support and camaraderie. Their smiling faces and warm conversations helped me push through the toughest days.

This work owes its success to collaborations with a remarkable group of scientists and students, including Bryce Johnson, Sarah Fahlberg, Sameer D'Costa, Chase Freschlin, and the dedicated members of both the Gitter and Romero Labs.

I want to express my gratitude to my advisor, Tony Gitter, who inspired me with his sincerity and commitment to the scientific process. Tony's mentorship shaped not only my academic pursuits but also my values and work ethic.

I want to thank my co-advisor, Phil Romero, for his valuable scientific insights and guidance, which helped shape my research over the years.

Finally, I want to thank my committee members Tony Gitter, Phil Romero, Sharon Li, and Yingyu Liang for their time and expertise in reviewing and providing feedback on my research and dissertation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Understanding the relationship between protein sequence and function is necessary to engineer novel proteins with applications in bioenergy, medicine, and agriculture. However, the complexity of the protein sequence-to-function mapping, compounded by the vast number of potential protein variants, poses a significant challenge in the field of protein engineering. This dissertation presents two original research studies describing advanced machine learning methods to address these challenges. The core of the research lies in developing and refining machine learning techniques to learn the protein sequence-function relationship from experimental data, thereby facilitating the engineering of protein variants with desired traits.

The first study describes a supervised deep learning framework to learn the sequence–function mapping from deep mutational scanning data. We tested several neural network architectures, including a graph convolutional network that incorporates protein structure. Our supervised learning approach displays superior performance over physics-based and unsupervised prediction methods. We find that networks that capture nonlinear interactions and share parameters across sequence positions are important for learning the relationship between sequence and function. This framework enables the design of proteins with enhanced properties, as exemplified by the engineering of a protein G B1 domain (GB1) variant that binds to immunoglobulin G with substantially higher affinity than wild-type GB1.

The second study introduces Mutational Effect Transfer Learning (METL), a method for predicting protein function that bridges the gap between traditional biophysics-based and machine learning approaches. We pretrain a transformer encoder on millions of molecular simulations to capture the relationship between protein sequence, structure, energetics, and stability. We then fine-tune the neural network to harness these fundamental biophysical signals and apply them when predicting protein functional scores from experimental assays. METL excels in protein engineering tasks like generalizing from small training sets and position extrapolation, although existing methods that train on evolutionary signals remain

powerful for many types of experimental assays. We demonstrate METL's ability to design functional green fluorescent protein variants when trained on only 64 examples.

The original research presented in this dissertation contributes to the rapidly evolving and dynamic field of computational methods for protein engineering. It encompasses not only novel findings but also provides a comprehensive background and commentary on the broader scope of the field. The conclusion reflects on my journey through graduate studies at the University of Wisconsin-Madison, covering topics such as performing research in a rapidly advancing field and the essence of conducting high-quality scientific work.

1 INTRODUCTION

1.1 Motivation

Proteins are diverse biomolecules that are prevalent throughout nature and serve many important functions in biology. Transport proteins like hemoglobin carry oxygen within our bodies, enzyme proteins like amylase digest nutrients, and hormone proteins like insulin regulate crucial physiological processes. Indeed, proteins are vital to nearly every cellular process. Their importance and ubiquity have earned them the informal title *the workhorses of the cell*.

While proteins have evolved over millions of years to perform their biological roles, modern science has unlocked the ability to harness their potential for new purposes. The field of protein engineering focuses on creating or modifying proteins to fulfill new functions. Some of the most impactful achievements in protein engineering have been in medicine. Protein-based therapeutics such as antibodies (Carter and Rajpal, 2022) and cytokines (Deckers et al., 2023) have been engineered to enhance developability and overall therapeutic efficacy, leading to better patient health outcomes.

Outside of medicine, engineered enzymes, which are a specific type of protein, are used in diverse applications. In laundry detergents, engineered enzymes can efficiently break down stains; in food processing, they can enhance flavor; and in organic synthesis, they can drive chemical reaction pipelines (Jemli et al., 2016). These biocatalyst enzymes are engineered to enhance catalytic activity and stability under adverse conditions, thereby improving efficiency and sustainability of

industrial processes (Alcántara et al., 2022).

Protein engineering enables us to harness the potential of proteins for human benefit, but it is an inherently challenging task due to the vastness of protein sequence space. Every protein is composed of an amino acid sequence that determines its three-dimensional structure and function. Assuming a sequence length of 300 amino acids and considering only the 20 standard amino acids commonly found in nature, the number of potential sequences is a staggering 20^{300}. To put that in perspective, this number is larger than the number of atoms in the known universe, which is estimated to be only 10^{82}. Compounding this challenge is that the mapping from protein sequence to function can be highly complex, shaped by thousands of intricate molecular interactions, dynamic conformational ensembles, and nonlinear relationships between biophysical properties.

The vastness and complexity of protein space presents a challenge for both experimental and computational protein engineering methods. Experimentally, it is not feasible to test the functional properties of all theoretical protein variants, and it is hard to know how a particular variant will function without testing it. Computational strategies can leverage multiple types of information, including experimental data, to extract important functional insights that can aid in protein engineering. They can make predictions about the effects of mutations or even generate entirely new sequences to prioritize for experimental testing. However, complex features and the nature of proteins, where minor sequence changes can have large functional impacts, make it difficult to predict how changes in amino acid sequence affect function.

Despite these challenges, recent breakthroughs in both experimental and computational methodologies promise to bring a new era of advancement in protein engineering (Bordin et al., 2023; Kouba et al., 2023). Novel high-throughput mutagenesis and screening techniques have enabled scientists to evaluate the function of up to a million protein variants in a single experiment (Fowler and Fields, 2014). These high-throughput methods provide rich information about how protein sequences relate to function. Concurrently, the field of machine learning has made exponential strides over the last decade. Neural networks have demonstrated remarkable success when applied to tasks in computer vision and natural language processing. New types of data, ever increasing computational power, mature software libraries, and advancements in neural network architecture and training techniques position us to see the same success in protein engineering that we have seen in other fields.

In this dissertation, I detail my contributions toward the challenging goal of understanding and computationally modeling protein sequence-function relationships for protein engineering. My work coincides with a period of great interest in protein engineering. The field has exploded in popularity and the landscape is rapidly evolving with new methodologies. In the remainder of this introduction, I provide the necessary biological and computational background to understand my contributions. Further, I define the scope of my research, contextualizing it within the broader field of computational methods for protein engineering, as the field stands today.

1.2 Biological Background

Proteins

Proteins are diverse and complex biomolecules composed of amino acids. They perform a range of fundamental molecular functions. These functions are often highly specialized and can range from binding other molecules to providing structural support to cells. The mechanism by which a protein carries out its function is determined by its unique amino acid sequence and resulting three-dimensional structure.

Every protein is defined by a unique amino acid sequence. There are twenty different amino acids commonly found in nature, often represented using 3-letter or 1-letter character codes (Table 1.1). These twenty amino acids have their own chemical and physical properties. For instance, an amino acid can be polar or nonpolar, hydrophobic or hydrophilic, or small or large. Proteins vary in length but can consist of hundreds to thousands of amino acids. Physical and chemical interactions between amino acids cause proteins to fold spontaneously into their three-dimensional structures.

The three-dimensional conformation of a protein is absolutely vital for its function. Even a minor sequence change, such as substituting one amino acid for a different one, can cause a change in the protein's structure and functional properties. This change could be positive, negative, or neutral, depending on the specific mutation and functional property of interest. For instance, an amino acid substitution could improve a protein's binding affinity to a particular binding partner, but it

Name	3-Char	1-Char	Name	3-Char	1-Char
Alanine	Ala	A	Leucine	Leu	L
Arginine	Arg	R	Lysine	Lys	K
Asparagine	Asn	N	Methionine	Met	M
Aspartic acid	Asp	D	Phenylalanine	Phe	F
Cysteine	Cys	C	Proline	Pro	P
Glutamic acid	Glu	E	Serine	Ser	S
Glutamine	Gln	Q	Threonine	Thr	T
Glycine	Gly	G	Tryptophan	Trp	W
Histidine	His	H	Tyrosine	Tyr	Y
Isoleucine	Ile	I	Valine	Val	V

Table 1.1: **Amino acids.** The twenty common amino acids and their abbreviations.

could also make the protein less stable. Understanding the relationship between protein sequence, structure, and function is fundamental to protein engineering.

The Sequence-Function Mapping

The sequence-function mapping (also known as the fitness landscape or sequence-function space) refers to the mapping from a protein amino acid sequence to a particular phenotype or functional property (Romero and Arnold, 2009). The mapping can be viewed as a function $f(x) = y_p$, where the input x is a protein amino acid sequence and the output y_p is a value describing the protein's phenotype or functional activity for a specific property p. The functional property could be any protein property of interest, simple or complex, from evolutionary fitness to enzymatic activity to binding affinity or beyond. The sequence-function mapping forms a surface in high dimensional space, where peaks might represent high-functioning sequences and valleys might represent low-functioning sequences

(Figure 1.1). The goal of protein engineering is to optimize over this sequence-function surface to identify high functioning sequences.

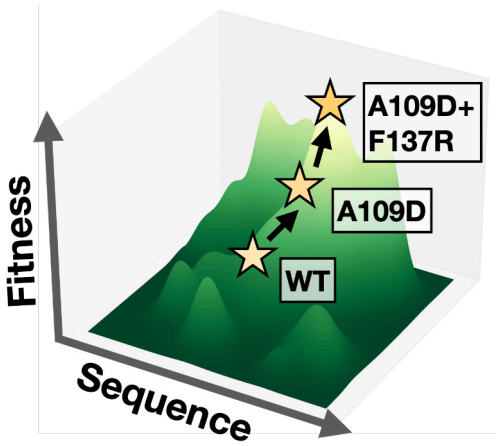

Figure 1.1: **A theoretical fitness landscape.** The stars represent individual protein sequences and are labeled according to their mutations from the wild-type (WT) sequence, where "wild-type" refers to the original, non-mutated form of the protein found in nature. The label consists of the WT amino acid, the sequence position where the amino acid substitution occurs, and the replacement amino acid. The goal of protein engineering is to find peaks in this landscape that represent high fitness sequences.

The mapping from sequence to function can be extremely complex. First, the number of possible protein sequences is massive, making it difficult to map the entire sequence-function landscape. The vast majority of protein sequences are non-functional, although the space around a functioning sequence may contain a higher concentration of functioning sequences. Single amino acid substitutions can have a profound effect on function, meaning the fitness landscape is not always smooth, even in regions containing functional sequences. Another complicating factor is epistasis, a phenomenon where the effects of multiple mutations can combine to have a nonlinear effect on function. In other words, a mutation that is beneficial on its own might actually be harmful when combined with another. To further em-

phasize the context-dependence of mutations, the environment in which a protein operates can also affect function. A mutation might have different effects based on environmental factors like temperature and the presence of other molecules. Moreover, proteins have multiple functions and molecular properties that may be affected differently by the same mutation. These factors make it challenging to model and predict how changes in amino acid sequence affect function.

Experimental Sequence-Function Data

Recent advances in DNA sequencing and high-throughput screening have transformed the ability to experimentally characterize the functional attributes of protein variants. Historically, assessing the function of even a single protein variant was a labor-intensive process and could only occur on small scales. Modern experimental methods referred to as deep mutational scanning (Fowler and Fields, 2014) have enabled scientists to measure the functional properties of hundreds of thousands of protein variants in a single experiment.

In a typical deep mutational scan, a base sequence undergoes mutagenesis, which is often random or systematically designed to capture all single substitutions, to generate a library of protein sequence variants. These variants usually differ from the base sequence by a few amino acid substitutions. Following mutagenesis, a sample of the generated library is sequenced using next-generation sequencing techniques, forming the *pre-screening* set. Subsequently, a high-throughput functional assay is used to select the variants with a specific functional property, creating a set of variants that is enriched for function. This set is sequenced and

forms the *post-screening* set. Finally, a ratio-based functional score is calculated for each variant based on the number of sequencing reads in the *pre-screening* and *post-screening* sets (Rubin et al., 2017).

In broader terms, deep mutational scanning data consists of large numbers of protein sequence variants that each have an associated score that quantifies their activity or fitness in a high-throughput function assay. The sequence variants are typically derived from a single base sequence and differ from each other by a few amino acid substitutions. The significance of the functional score depends on the protein, the experimental parameters, and the biological assay. For example, the high-throughput assay could measure specific properties such as the brightness of a fluorescent protein (Sarkisyan et al., 2016), the binding affinity of an IgG-binding protein (Olson et al., 2014), or the catalytic activity of an enzyme (Romero et al., 2015).

Deep mutational scanning data can be extremely valuable, but there are notable limitations. First and foremost, it is not feasible to test all possible protein variants. Furthermore, the random mutagenesis used for many deep mutational scans means it is hard to control exactly what variants are generated and tested, which can lead to gaps in the experimental data. Many deep mutational scans focus on single substitution variants only. This limitation could be significant in protein engineering, where there is interest in how multiple mutations can combine to achieve desired functional properties, especially in the context of epistatic effects, or nonlinear interactions between mutations. Additionally, deep mutational scans can be costly and difficult to perform.

A significant drawback in some deep mutational scans is dataset quality. Dataset quality can impact our understanding of and attempts to model the sequence-function mapping. Multiple factors can affect dataset quality, including the number of sequencing reads for each variant, the resolution of the assay, and the nature of the measured functional property (Gelman et al., 2021). Attempts are made to ensure data quality by filtering out variants based on the number of sequencing reads, performing multiple experimental and technical replicates, and computing scores across multiple replicates. Ultimately, understanding experimental methodologies and recognizing the potential for dataset noise is important for drawing meaningful conclusions and modeling the sequence-function mapping.

Even in the era of deep mutational scanning, low-throughput experiments are still commonly performed and provide valuable insights into protein function. They enable scientists to test specific variants of interest whereas deep mutational scans usually sample mutations randomly. Furthermore, it is not always possible to set up a high-throughput experimental assay, and in these cases a low-throughput experiment may be the only option for acquiring data about a particular phenotype. Finally, a low-throughput assay could have higher resolution and reduced noise.

Whether sequence-function data comes from low throughput or high throughput experiments, it represents a sample of the vast sequence-function space. The data, although limited, can provide rich and valuable information about protein variant function. Statistics and machine learning can leverage this information to model and extract insights about the nature of the protein sequence-function mapping.

1.3 Computational Background

Machine Learning

Broadly speaking, machine learning is a sub-field of computer science and statistics focused on modeling and extracting meaningful insights from data. Computational approaches for protein engineering span a wide range of machine learning techniques. This section touches on several important aspects of machine learning relevant for understanding this dissertation.

Supervised and Unsupervised Learning

Machine learning models can be categorized by how they learn from data, and the learning approaches are largely determined by the type of data available. Labeled data consists of input-output pairs that explicitly inform the model about what the output should be for a given input. For instance, experimental sequence-function datasets are considered labeled datasets because every protein variant is labeled with a functional score. Conversely, unlabeled data consists solely of inputs without associated labels. Evolutionary data, described in more detail in Section 1.4, often comes in the form of a collection of sequences known to be functional in nature. These sequences do not have explicit labels, but the fact that they are all functional in nature indicates there is an underlying signal that can be leveraged by machine learning methods.

Typically, machine learning methods that learn input-output mappings from labeled data are considered *supervised* machine learning methods. On the other

hand, machine learning methods that extract underlying patterns or representations from unlabeled data are considered *unsupervised*. There is a spectrum of machine learning techniques ranging from supervised to unsupervised, including approaches that employ semi-supervised and self-supervised learning. For example, natural language models often learn from large collections of unlabeled text. These methods generate labels from the data itself by performing tasks like predicting the next word of a sentence. In this case, the input becomes the sentence leading up to the next word, and the label becomes the next word. This type of learning is referred to as self-supervised, and this approach is also used by protein language models, described in Section 1.4.

Neural Networks

Neural networks are a type of machine learning model capable of learning complex, nonlinear input-output mappings, extracting meaningful, higher-level features from raw inputs, and generalizing from training data to new, unseen inputs (Ching et al., 2018). Neural networks accept raw inputs in the form of vectors or matrices representing text, images, or sequences. They transform the raw input through a series of successive layers. Each layer performs some type of mathematical operation on its input, such as multiplying the input by a set of learned weights, to produce an output. The final layer of the neural network outputs a meaningful prediction, perhaps representing a classification label for an image, or in our case, a functional score prediction for a protein variant.

Neural networks learn iteratively from data by updating their weights in re-

sponse to an objective function. The objective functions defines what is considered a desired or undesired output, and an optimizer calculates how to change the existing weights to push the network output toward the desired output. Neural networks are optimized using gradient-descent based algorithms such as stochastic gradient descent and Adam (Kingma and Ba, 2017).

There are numerous neural network architectures, including fully connected networks, convolutional networks (Alzubaidi et al., 2021), recurrent networks (Lipton et al., 2015), and transformers (Vaswani et al., 2017). Different neural network architectures confer different inductive biases that affect learned patterns. In practice, this means that some network architectures may be better suited for some types of data. For instance, convolutional networks work well with images, and transformers have been shown to work well with text data. Research into architectures is ongoing, and transformers have also been applied to computer vision tasks (Han et al., 2023).

What kind of neural network architecture is best for protein sequences is still an open question (Yang et al., 2023). Proteins, being three-dimensional molecules, have different considerations than both images and text. Convolutional networks, recurrent networks, and transformers are just some of the neural network architectures that have been applied to modeling protein sequences. The original work in Chapter 2 explores incorporating protein structure into graph convolutional neural networks, and the original work in Chapter 3 implements a protein structure-based relative position embedding for transformer models. Both of these are attempts to optimize the neural network architecture in response to the biological context of

proteins.

Transfer learning

Transfer learning is broadly important in modern machine learning, and it is a large part of the protein fitness prediction framework described in Chapter 3. The concept behind transfer learning is to train a machine learning model on a set of data from one domain and transfer the learned knowledge to a different but related domain (Weiss et al., 2016; Zhuang et al., 2020). Transfer learning can be used to improve performance of machine learning models when there is limited training data for the target domain. The training data and inductive biases transferred from the source domain enable models to generalize better in the target domain. In the context of computational methods for protein engineering, experimental data can be limited in size or have biases that make it difficult for machine learning models to learn and generalize to new sequences. Transfer learning is one way to improve the predictive generalization performance of protein fitness predictors.

1.4 Survey of the Current Landscape

The volume of protein data has exploded over the last decade with advances in DNA sequencing, three-dimensional structure determination, and high-throughput protein function screening. With these increasing data, statistics and machine learning approaches have emerged as powerful methods to understand the complex mapping from protein sequence to function.

Experimental Sequence-Function Data

Experimental sequence-function data, such as deep mutational scanning data (Section 1.2), provides direct insights into how amino acid sequence changes affect specific protein functional properties. This type of data can be extremely valuable for protein engineering, especially when the biological assay aligns with the function that is being engineered. Experimental data is commonly used as training data for supervised machine learning methods. Indeed, my work, described in this dissertation, focuses specifically on creating machine learning models capable of accurately predicting experimental functional scores, using this type of experimental data as training data. Even protein fitness prediction methods that do not train on experimental data, such as unsupervised or zero-shot protein function predictors, often use experimental data to perform benchmarking and evaluation.

Deep mutational scanning has enabled scientists to generate more experimental sequence-function data than ever before. As a result, at least two sequence-function databases have come online during the course of my research: ProtaBank (Wang et al., 2018) and MaveDB (Esposito et al., 2019). These databases contain thousands of individual entries representing both high-throughput and low-throughput sequence-function datasets. Additionally, for the task of predicting fitness of mutated proteins, a curated collection of deep mutational scanning datasets was made available under the name ProteinGym (Notin et al., 2022).

In the early stages of my research, I explored modeling deep mutational scanning data as a classification problem using positive-unlabeled learning (Bekker and Davis, 2020). Deep mutational scanning typically produces two sets of protein

sequence variants: the *pre-selection* and *post-selection* sets (Section 1.2). Depending on the experimental parameters, the *pre-selection* set can be seen as containing unlabeled sequences, while the *post-selection* set can be seen as containing positive (functional) sequences. Indeed, others pursued this positive-unlabeled strategy with success (Song et al., 2021).

However, the release of Enrich2 (Rubin et al., 2017), a software tool for computing functional scores from raw sequencing read counts, provided a stronger theoretical and practical basis for formulating the modeling problem as a regression on functional scores. For the most part, the field has adopted the regression framework as the primary modeling approach for deep mutational scanning data. Since Enrich2, several other tools for computing functional scores from raw sequencing read counts have become available, including DiMSum (Faure et al., 2020) and mutscan (Soneson et al., 2023). It is common for researchers who publish deep mutational scanning datasets to provide pre-computed functional scores.

Natural and Evolutionary Data

Natural protein sequences and protein structures provide a rich source of evolutionary information, which is often used for computational protein analysis. Databases such as UniProt (The UniProt Consortium, 2023) and InterPro (Paysan-Lafosse et al., 2023) catalog and organize known protein sequences. The number of protein sequences contained in UniProt has risen steadily over the last decade, and the database currently contains greater than 550,000 curated entries. InterPro has cataloged many of these sequences into over 20,000 protein families. Addition-

ally, the Protein Data Bank (Berman et al., 2000) provides a repository for protein three-dimensional structures.

Unlike experimental sequence-function data, the types of information contained in these sequence databases do not directly measure function using biological assays. However, this sequence information still contains a strong signal about what makes a protein functional, originating from natural selection and evolution. Sequences that are conserved throughout evolution are likely to be important for maintaining function. Statistical and machine learning methods can leverage this underlying signal to make predictions about protein variant fitness.

Computational Methods for Protein Engineering

With the expansion of available protein data and the standardization of data repositories, there has been a surge in the number of protein-related computational methods. While some methods are designed specifically for protein engineering, others are capable of performing a wider range of protein-related tasks. These computational methods can be broadly categorized by the types of data they use (evolutionary, experimental, biophysics, sequence, structure), the computational approaches and models they employ (predictive, generative, machine learning, statistics), and the way they prioritize variants for experimental characterization (direct generation of functional proteins or a search over sequence space through an optimization framework). It is important to note that many methods exhibit significant overlap, combining multiple data types and modeling approaches. Thus, the aforementioned categories mainly serve as a framework for understanding

related methods, rather than a way to definitively categorize each method.

Within protein engineering pipelines, computational methods play an essential role in prioritizing variants for experimental characterization. Some computational methods, such as generative methods, can directly output protein sequences or structures that may have enhanced function. These methods can go as far as outputting proteins that are not based on existing natural proteins, in a process known as de novo protein design (Huang et al., 2016; Pan and Kortemme, 2021; Ding et al., 2022; Watson et al., 2023).

However, many of the methods discussed in this section are predictive rather than generative. Predictive methods can estimate the fitness of variants, but they do not inherently prioritize them. In order to prioritize variants for experimental characterization, these methods need to be paired with an optimization framework or algorithm, such as simulated annealing. The combination of the prediction method and optimization framework serves as a way to computationally explore the sequence-function landscape and find high-functioning sequences. Optimization for variant prioritization is an active research area (Brookes et al., 2019; Angermueller et al., 2020; Fannjiang and Listgarten, 2020; Linder and Seelig, 2021).

Evolutionary Data Methods

An important category of computational methods for protein engineering is those that utilize evolutionary information. The evolutionary record, in the form of known sequences that occur in nature, contains a strong signal about what makes a protein functional. These computational methods are designed to capture that

underlying signal. Such methods typically employ unsupervised or self-supervised machine learning techniques, and they use evolutionary data in the form of either multiple sequence alignments (MSAs) or raw, unaligned sequences.

Multiple sequence alignments consist of homologous sequences that are aligned to account for insertions, deletions, and amino acid substitutions that can occur over the course of evolution. They usually cover a specific protein family or closely related proteins. Multiple sequence alignments make it evident what sequence positions are conserved throughout evolution and thus important for the function of a given protein family. Methods that utilize MSAs have demonstrated excellent performance when evaluated as fitness predictors on experimental data, ranging from statistical co-evolutionary models like EVMutation (Hopf et al., 2017), to neural network-based models like DeepSequence (Riesselman et al., 2018) and EVE (Frazer et al., 2021), to unique methodologies such as GEMME (Laine et al., 2019), which bases its predictions on evolutionary conservation estimated from Joint Evolutionary Trees (Engelen et al., 2009).

One drawback of MSA-based methods is that it is not possible to align sequences from diverse protein families because the sequences are too divergent. Thus, an MSA must be created for every target protein family. However, some protein families do not have informative alignments due to a small number of existing natural sequences in the family. Raw, unaligned sequences still contain the same underlying evolutionary signal as MSAs, but they can include diverse sequences and encapsulate the entirety of known sequences.

Protein language models, inspired by natural language models, have emerged as

powerful methods to capture the diversity of known protein sequences (Alley et al., 2019; Rives et al., 2021; Bepler and Berger, 2021; Elnaggar et al., 2022; Yang et al., 2023; Elnaggar et al., 2023; Chandra et al., 2023). Trained on large, diverse protein sequence databases like UniProt, these models typically employ self-supervised training techniques like masked token prediction or next token prediction. Protein language models utilize a range of different neural network architectures including sequence convolutional networks, recurrent neural networks, and transformers. Protein language models learn a rich representation of protein sequences that captures underlying signals in the evolutionary data, including protein stability and fitness.

Protein language models, and unsupervised evolutionary methods in general, can make fitness predictions for protein variants using zero-shot approaches. These scoring approaches often use log-odds ratios or log-likelihood scores with the MSA or wild-type sequence providing necessary background context to score a given variant. Protein language models can also serve as foundation models that can be fine-tuned on experimental data. In addition to predicting protein fitness, protein language models can be useful across a range of other protein analysis tasks, such as protein structure prediction (Lin et al., 2023).

Protein language models are undergoing rapid improvement with changes in data, architecture, and model capacity. Some protein language models can utilize MSAs in additional to raw, unaligned sequences (Rao et al., 2021; Notin et al., 2022). Parallel to the trend in natural language processing, there has been an increase in capacity of protein language models (Hesslow et al., 2022), with larger models

offering some improvement. However, it remains to be seen whether larger models will bring the same kinds of revolutionary performance to the protein space that they brought to the natural language space (Chen et al., 2023).

Physics-Based Methods

Proteins are three-dimensional molecules that perform their functions through physical and biochemical interactions. Thus, protein structure and physical modeling provide an informative signal and foundation for computational methods to understand protein function. Rosetta stands out as a comprehensive, physics-based molecular modeling software suite that is capable of performing a range of tasks, including protein structure prediction, protein docking, and protein stability prediction (Alford et al., 2017). It uses biophysical energy functions and is parameterized in part with experimental protein structures to model how amino acids interact on a molecular scale. In addition to Rosetta, another noteworthy physics-informed tool is FoldX, although FoldX is more narrowly focused on predicting protein stability upon mutation (Schymkowitz et al., 2005). Both Rosetta and FoldX are more correlated with deep mutational scanning scores than other stability predictors (Gerasimavicius et al., 2023).

There has been increased interest in how to effectively use physics information to help prioritize variants for protein engineering. Rosetta has internal optimization tools to explore the energy landscape and design stable protein structures, including de novo protein design. In addition to Rosetta's internal optimization tools, Rosetta's calculated energy terms and stability predictions can be used as

features in supervised learning frameworks (Wang et al., 2022a; Harmalkar et al., 2023). The original research described in Chapter 3 focuses on using Rosetta molecular simulations to pretrain neural networks as part of a larger transfer learning framework with experimental data.

Experimental Data Methods

Experimental data offers direct insights into how amino acid sequence changes affect specific protein functional properties. Thus, this type of data provides immense value for computational protein engineering methods. While some computational methods use experimental data alone, others use it in combination with evolutionary or physics-based data.

Recent research has focused on understanding the nuances and specifics of modeling deep mutational scanning data. Neural networks have been trained as a complementary analytics tool as part of a deep mutational scan to help understand the fitness landscape of a green fluorescent protein (Sarkisyan et al., 2016). Another study explored a wider range of data encodings and machine learning models, including support vector machines, random forests, and convolutional neural networks (Xu et al., 2020). They found that sequence convolutional networks with an amino acid physicochemical property encoding (Kawashima et al., 2008) performed stronger than other tested methods. Other studies have gone deeper into modeling specifics of deep mutational scanning data, considering factors such as epistatic interactions and epistatic regularization (Otwinowski et al., 2018; Aghazadeh et al., 2021; Tareen et al., 2022). The original work in Chapter 2 explores how different

types of neural networks, including graph convolutional neural networks that incorporate protein structure information, can be applied to model deep mutational scanning data.

Experimental data can be limited in size, coming from low-throughput instead of high-throughput experiments. It can also be biased in terms of the sampled sequence space, potentially only containing examples of single amino acid substitutions or missing examples of mutations in certain positions. Thus, many computational approaches combine experimental data with evolutionary or physics-based information to provide additional signal or inductive biases to help improve predictive generalization performance.

A straightforward yet surprisingly effective approach has been to combine a one hot sequence encoding and evolutionary features in a linear regression supervised framework (Hsu et al., 2022). However, other methods have taken more integrated approaches to combining different types of data, including evolutionary data and protein structure (Luo et al., 2021; Li et al., 2023). Furthermore, experimental data can be used to fine-tune protein language models to predict experimental functional scores. The protein language models provide an inductive bias that improves predictive performance over training a model on only experimental data from scratch. One study specifically examined finetuning a protein language model on small amounts of labeled data to perform low-N protein engineering (Biswas et al., 2021). The original work described in Chapter 3 explores integrating biophysical information in a transfer learning framework with experimental data.

Benchmarking

As research into computational methods for protein engineering has become more established, the need for standardized benchmarking suites has increased. Other protein-related fields such as structure prediction have standardized benchmarks and competitions like CASP (Kryshtafovych et al., 2019). There have been several attempts to set up benchmarking suites for protein language models and protein engineering methods, including TAPE (Rao et al., 2019), FLIP (Dallago et al., 2022), FLOP (Groth et al., 2023), and ProteinGym (Notin et al., 2022). However, these only cover specific application scenarios, and so far none have emerged as the definitive standard for evaluating computational protein engineering methods. In my research, I have combined relevant evaluations from these benchmarks and others.

Beyond Protein Engineering

Computational protein research extends beyond protein engineering, encompassing diverse objectives and methodologies with applications in various areas of biology and medicine. Some of these research areas overlap with computational protein engineering methods. This section examines three sub-fields of computational protein research with relevance to protein engineering: predicting protein three-dimensional structures, predicting whether a given sequence variant is pathogenic in humans, and protein function annotation with Gene Ontology prediction.

Accurately predicting protein structures has long been an important task in protein science. The Folding@Home project, which has garnered substantial at-

tention in popular culture, has been ongoing for over 20 years (Voelz et al., 2023). Structure prediction has traditionally relied on homology modeling from previously determined structures and physics-based energy minimization techniques (Kuhlman and Bradley, 2019). More recently, there has been a breakthrough in protein structure prediction with deep learning and the introduction of AlphaFold 2 (Jumper et al., 2021). Protein language models, which can be used for protein fitness prediction, have also been shown to learn the necessary underlying information needed to predict protein structure when incorporated into a structure prediction framework (Lin et al., 2023). Accurate structure prediction is relevant not only for general understanding of proteins, but also protein engineering efforts. For instance, AlphaFold structures were used for some of the molecular simulations described in Chapter 3.

Another relevant area of computational protein research concerns variant effect predictors (VEPs), which predict whether a given amino acid substitution might be pathogenic or harmful to humans (Adzhubei et al., 2010; Hecht et al., 2015; Vaser et al., 2016; Aghazadeh et al., 2021; Livesey and Marsh, 2022). On the surface, VEPs have many similarities to protein fitness predictors used for protein engineering. VEPs employ a range of models including neural networks that take amino acid sequences as inputs, and they often use evolutionary or deep mutational scanning data for training or evaluation. Indeed, some protein fitness prediction methods used for protein engineering have also been evaluated as VEPs (Hopf et al., 2017; Livesey and Marsh, 2023).

However, there are also several key differences between VEPs and protein fitness

predictors used in protein engineering. Unlike fitness predictors, some VEPs train on specialized databases containing examples of harmful variants. VEPs typically output qualitative predictions, classifying whether a given mutation is pathogenic, instead of quantitative estimates of protein molecular function. Additionally, many VEPs focus on predicting single amino acid substitutions instead of modeling multiple mutations with epistatic interactions. For the most part, these differences make VEPs specialized enough as to not be directly comparable to protein fitness prediction methods used for protein engineering.

Another key area of related research, and the final one discussed in this dissertation, pertains to predicting the biological and cellular functions of proteins. The Gene Ontology (GO) knowledgebase catalogues biological, cellular, and molecular functions of proteins (Ashburner et al., 2000; The Gene Ontology Consortium et al., 2023). Predicting GO terms from this database as a way to annotate proteins with their biological function is an active sub-field of computational protein research (Zhao et al., 2020; Vu and Jung, 2021). While predicting GO terms for unknown natural proteins is quite different than predicting the fitness of protein variants, there is some overlap in the underlying methodologies for these two objectives. For instance, both types of methods use protein sequence features and machine learning. Additionally, GO term prediction benefits from informative protein representations like those from protein language models, which are also used for protein fitness prediction.

1.5 Scope and Dissertation Overview

My contributions toward machine learning methods for protein engineering stand as part of the ongoing innovation and expanding potential of the field. While this introduction has touched on a wide range of data types and methods, my research specifically focuses on deep learning methods for predicting protein variant fitness. In particular, I examine neural networks to model the relationship between protein sequence and function, as captured by experimental data like deep mutational scanning data.

Within this problem scope, I explore a number of important open questions in the field, which include, but are not limited to:

- What neural network architectures work best?
- What types of signals and training data are helpful?
- How do we incorporate biological context like protein structure?
- How does the quality of data affect the learning process?
- How can we achieve strong generalization when experimental data is limited?
- How do we incorporate biophysical knowledge into models?

This dissertation presents original research and perspectives on these questions and others. The remainder of this dissertation is organized as follows. Chapters 2 and 3 describe my original research in computational methods for protein engineering. Chapter 2 presents an exploration of neural network architectures

including fully connected networks, sequence convolutional networks, and graph convolutional networks that incorporate protein structure information. This chapter includes comparisons to other types of methods such as unsupervised evolutionary and physics-based methods. Furthermore, I detail experiments that evaluate the effects of data quality on the learning process and investigate the underlying representations learned by the models. Chapter 3 introduces Mutational Effect Transfer Learning (METL), a method for predicting protein variant fitness that leverages transfer learning from molecular simulations. I present a detailed evaluation of METL on challenging protein engineering-related tasks and a comprehensive look at how METL compares to evolutionary-based methods. Finally, Chapter 4 provides a summary of my original work and contribution to the field, and a reflection on the years of research leading to this dissertation.

The work presented in this chapter was performed in collaboration with Sarah A. Fahlberg, Pete Heinzelman, Philip A. Romero, and Anthony Gitter (Gelman et al., 2021):

> Gelman, Sam, Sarah A. Fahlberg, Pete Heinzelman, Philip A. Romero, and Anthony Gitter. 2021. Neural networks to learn protein sequence-function relationships from deep mutational scanning data. *Proceedings of the National Academy of Sciences* 118(48):e2104878118. DOI: 10.1073/pnas.2104878118.

2.1 Introduction

Understanding the mapping from protein sequence to function is important for describing natural evolutionary processes, diagnosing genetic disease, and designing new proteins with useful properties. This mapping is shaped by thousands of intricate molecular interactions, dynamic conformational ensembles, and nonlinear relationships between biophysical properties. These highly complex features make it challenging to model and predict how changes in amino acid sequence affect function.

The volume of protein data has exploded over the last decade with advances in DNA sequencing, three-dimensional structure determination, and high-throughput screening. With these increasing data, statistics and machine learning approaches have emerged as powerful methods to understand the complex mapping from protein sequence to function. Unsupervised learning methods such as EVmutation

(Hopf et al., 2017) and DeepSequence (Riesselman et al., 2018) are trained on large alignments of evolutionarily related protein sequences. These methods can model a protein family's native function, but they are not capable of predicting specific protein properties that were not subject to long-term evolutionary selection. In contrast, supervised methods learn the mapping to a specific protein property directly from sequence-function examples. Many prior supervised learning approaches have limitations, such as the inability to capture nonlinear interactions (Fox et al., 2007; Song et al., 2021), poor scalability to large datasets (Romero et al., 2013), making predictions only for single-mutation variants (Gray et al., 2018), or a lack of available code (Xu et al., 2020). Other learning methods leverage multiple sequence alignments and databases of annotated genetic variants to make qualitative predictions about a mutation's effect on organismal fitness or disease, rather than making quantitative predictions of molecular phenotype (Vaser et al., 2016; Adzhubei et al., 2010; Hecht et al., 2015). There is a current need for general, easy to use supervised learning methods that can leverage large sequence-function datasets to predict specific molecular phenotypes with the high accuracy required for protein design. We address this need with a usable software framework that can be readily adopted by others for new proteins (Wang and Gamazon, 2022).

We present a deep learning framework to learn protein sequence-function relationships from large-scale data generated by deep mutational scanning experiments. We train supervised neural networks to learn the mapping from sequence to function. These trained networks can then generalize to predict the functions of previously unseen sequences. We examine network architectures with different

representational capabilities including linear regression, nonlinear fully connected networks, and convolutional networks that share parameters. Our supervised modeling approach displays strong predictive accuracy on five diverse deep mutational scanning datasets and compares favorably with state-of-the-art physics-based and unsupervised prediction methods. Across the different architectures tested, we find that networks that capture nonlinear interactions and share information across sequence positions display the greatest predictive performance. We explore what our neural network models have learned about proteins and how they comprehend the sequence-function mapping. The convolutional neural networks learn a protein sequence representation that organizes sequences according to their structural and functional differences. In addition, the importance of input sequence features displays a strong correspondence to the protein's three-dimensional structure and known key residues. Finally, we used an ensemble of the supervised learning models to design five protein G B1 domain (GB1) sequences with varying distances from the wild-type. We experimentally characterized these sequences and found the top design binds to immunoglobulin G (IgG) with at least an order of magnitude higher affinity than wild-type GB1.

2.2 Results

A Deep Learning Framework to Model the Sequence–Function Mapping

Neural networks are capable of learning complex, nonlinear input-output mappings; extracting meaningful, higher-level features from raw inputs; and generalizing from training data to new, unseen inputs (Ching et al., 2018). We develop a deep learning framework to learn from large-scale sequence-function data generated by deep mutational scanning. Deep mutational scanning data consist of thousands to millions of protein sequence variants that each have an associated score that quantifies their activity or fitness in a high-throughput function assay (Fowler and Fields, 2014). We encode the protein sequences with a featurization that captures the identity and physicochemical properties of each amino acid at each position. Our approach encodes the entire protein sequence and thus can represent multimutation variants. We train a neural network to map the encoded sequences to their associated functional scores. After it is trained, the network generalizes and can predict functional scores for new, unseen protein variants (Fig. 2.1a).

We test four supervised learning models to explore how different internal representations influence the ability to learn the mapping from protein sequence to function: linear regression and fully connected, sequence convolutional, and graph convolutional neural networks (Fig. 2.1b). Linear regression serves as a simple baseline because it cannot capture dependencies between sites, and thus, all residues make additive contributions to the predicted fitness. Fully connected networks

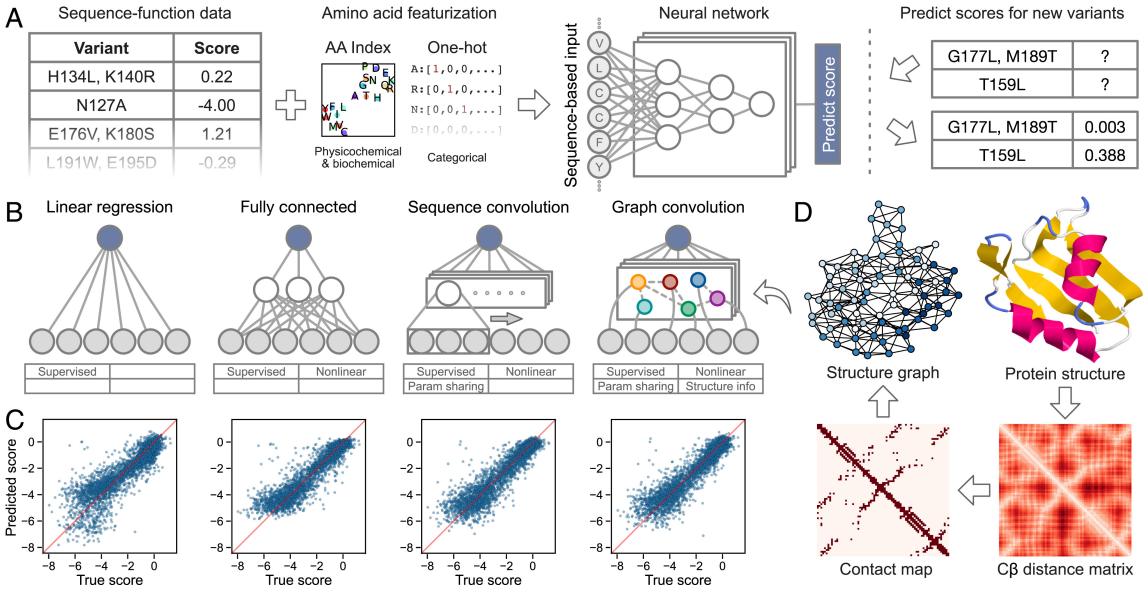

Figure 2.1: **Overview of our supervised learning framework.** (a) We use sequence-function data to train a neural network that can predict the functional score of protein variants. The sequence-based input captures physicochemical and biochemical properties of amino acids and supports multiple mutations per variant. The trained network can predict functional scores for previously uncharacterized variants. (b) We tested linear regression and three types of neural network architectures: fully connected, sequence convolutional, and graph convolutional. (c) Scatterplots showing performance of trained networks on the Pab1 dataset. (d) Process of generating the protein structure graph for Pab1. We create the protein structure graph by computing a residue distance matrix from the protein's three-dimensional structure, thresholding the distances, and converting the resulting contact map to an undirected graph. The structure graph is the core part of the graph convolutional neural network.

incorporate multiple hidden layers and nonlinear activation functions, enabling them to learn complex nonlinearities in the sequence to function mapping. In contrast to linear regression, fully connected networks are capable of modeling how combinations of residues jointly affect function beyond simple additive effects. These nonadditive effects are known as mutational epistasis (Starr and Thornton, 2016; Olson et al., 2014). Neither linear regression nor fully connected networks are

able to learn meaningful weights for amino acid substitutions that are not directly observed in the training set.

Convolutional neural networks have parameter sharing architectures that enable them to learn higher-level features that generalize across different sequence positions. They learn convolutional filters that identify patterns across different parts of the input. For example, a filter may learn to recognize the alternating pattern of polar and nonpolar amino acids commonly observed in β-strands. Applying this filter would enable the network to assess β-strand propensity across the entire input sequence and relate this higher-level information to the observed protein function. Importantly, the filter parameters are shared across all sequence positions, enabling convolutional networks to make meaningful predictions for mutations that were not directly observed during training. We develop a sequence-based convolutional network that integrates local sequence information by applying filters using a sliding window across the amino acid sequence. We also develop a structure-based graph convolutional network that integrates three-dimensional structural information and may allow the network to learn filters that correspond to structural motifs. The graph convolutional network applies filters to neighboring nodes in a graph representation of the protein's structure. The protein structure graph consists of a node for each residue and an edge between nodes if the residues are within a specified distance in three-dimensional space (Fig. 2.1d).

Evaluating Models Learned from Deep Mutational Scanning Data

We evaluated the predictive performance of the different network architectures on five diverse deep mutational scanning datasets representing proteins of varying sizes, folds, and functions: Aequorea victoria green fluorescent protein (avGFP), β-glucosidase (Bgl3), GB1, poly(A)-binding protein (Pab1), and ubiquitination factor E4B (Ube4b) (Table 2.1 and Fig. 2.2a). These datasets range in size from ~25,000 to ~500,000 sequence-score examples. We randomly split each dataset into training, tuning, and testing sets to optimize hyperparameters and evaluate predictive performance on data that were not seen during training. The learned models displayed excellent test set predictions for most datasets, with Pearson's correlation coefficients ranging from 0.55 to 0.98 (Fig. 2.2b). The trends are generally similar using Spearman's correlation coefficient (Fig. A.1), although the differences between linear regression and the neural networks are smaller.

	Description	Organism	Molecular function	Selection	Length	Variants	Ref
avGFP	Green fluorescent protein	A. victoria	Fluorescence	Brightness	237	54,024	(Sarkisyan et al., 2016)
Bgl3	Beta glucosidase	Streptococcus sp.	Hydrolysis of β-glucosidic linkages	Enzymatic activity	501	26,653	(Romero et al., 2015)
GB1	IgG-binding domain of protein G	Streptococcus sp.	IgG-binding	IgG-Fc binding	56	536,084	(Olson et al., 2014)
Pab1	RRM domain of Pab1	S. cerevisiae	poly(A)-binding	mRNA binding	75	40,852	(Melamed et al., 2013)
Ube4b	U-box domain of E4B	M. musculus	Ubiquitin activating enzyme activity	Ubiquitin ligase activity	102	98,297	(Starita et al., 2013)

Table 2.1: **Deep mutational scanning datasets.** We evaluated the models on deep mutational scanning datasets representing proteins of varying sizes, folds, and functions.

For comparison, we also evaluated the predictive performance of established physics-based and unsupervised learning methods Rosetta (Alford et al., 2017), EVmutation (Hopf et al., 2017), and DeepSequence (Riesselman et al., 2018), which are not trained using the deep mutational scanning data. Our supervised learning approach achieves superior performance to these other methods on all five protein datasets, demonstrating the benefit of training directly on sequence-function data

(Fig. 2.2b). This result is unsurprising because supervised models are tailored to the specific protein property and sequence distribution in the dataset. Rosetta predictions consider the energetics of the protein structure and therefore do not capture the more specific aspects of protein function. Unsupervised methods such as EVmutation and DeepSequence are trained on natural sequences and thus only capture aspects of protein function directly related to natural evolution. Despite their lower performance, physics-based and unsupervised methods have the benefit of not requiring large-scale sequence-function data, which are often difficult and expensive to acquire.

The different supervised models displayed notable trends in predictive performance across the datasets. The nonlinear models outperformed linear regression, especially on variants with low scores (Fig. A.2), high epistasis (Fig. A.3), and in the case of avGFP, larger numbers of mutations (Fig. A.4). The three nonlinear models performed similarly when trained and evaluated on the full training and testing sets. However, the convolutional networks achieved a better mean squared error when evaluating single-mutation variants in Pab1 and GB1 (Fig. A.4). For most proteins, the convolutional networks also had superior performance when trained on smaller training sets (Fig. 2.2c).

The quantitative evaluations described thus far involve test set variants that have similar characteristics to the training data. We also tested the ability of the models to extrapolate to more challenging test sets. In mutational extrapolation, the model makes predictions for variants containing mutations that were not seen during training. The model must generalize based on mutations that may occur in the

same or other positions. The convolutional networks achieved strong performance for one dataset ($r > 0.9$), moderate performance for two additional datasets ($r > 0.6$), and outperformed linear regression and fully connected networks across all datasets (Fig. 2.2d). In positional extrapolation, the model makes predictions for variants containing mutations in positions that were never modified in the training data. The performance of all models is drastically reduced (Fig. A.5), highlighting the difficulty of this task (Mater et al., 2020). In theory, the parameter sharing inherent to convolutional networks allows them to generalize the effects of mutations across sequence positions. This capability may explain the convolutional networks' superior performance with reduced training set sizes and mutational extrapolation. However, it is still difficult for the convolutional networks to perform well when there are no training examples of mutations in a particular position, such as in positional extrapolation.

The sequence convolutional and graph convolutional networks displayed similar performance across all evaluation metrics, despite the inclusion of three-dimensional protein structure information in the graph topology. To assess the impact of integrating protein structure in the neural network architecture, we created graph convolutional networks with misspecified baseline graphs that were unrelated to the protein structure. These baseline graphs include shuffled, disconnected, sequential, and complete graph structures (Fig. A.6). We found that networks trained using these misspecified baseline graphs had accuracy similar to networks trained with the actual protein structure graph, indicating that protein structure information is contributing little to the model's performance (Fig. A.7). We also

trained the convolutional networks with and without a final fully connected layer and found that this fully connected layer was more important than a correctly specified graph structure. In almost all cases, this final fully connected layer helps overcome the misspecified graph structure (Fig. A.7). Overall, these results suggest that the specific convolutional window is not as critical as sharing parameters across different sequence positions and integrating information with a fully connected layer.

The goal of protein engineering is to identify optimized proteins, and models can facilitate this process by predicting high-activity sequences from an untested pool of sequences. Pearson's correlation captures a model's performance across all variants, but it does not provide information regarding a model's ability to retrieve and rank high-scoring variants. We evaluated each model's ability to predict the highest-scoring variants within a given experimental testing budget (Fig. 2.2e). We calculated recall by asking each model to identify the top N variants from the test set, where N is the budget, and evaluating what fraction of the true top 100 variants was covered in this predicted set. The supervised models consistently achieve higher recall than Rosetta and the unsupervised methods, although the differences are small for Pab1 and Bgl3. In practice, the budget depends on the experimental costs of synthesizing and evaluating variants of the given protein. For GB1, a feasible budget may be 100 variants, and the supervised models can recall over 60% of the top 100 sequences with that budget.

Another important performance metric for protein engineering is the ability to prioritize variants that have greater activity than the wild-type protein. We

calculated the mean and maximum scores of the top N predicted test set variants ranked by each model (Figs. A.8 and A.9). We find that the variants prioritized by the supervised models have greater functional scores than the wild type on average, even when considering variants ranked beyond the top thousand sequences for some datasets. In contrast, Rosetta and the unsupervised models generally prioritize variants with mean scores worse than the wild type. The maximum score of the prioritized variants is also important because it represents the best variant suggested by the model. We find that nearly all models are able to prioritize variants with a maximum score greater than the wild type. The relative performance of each model is dependent on the dataset. Notably, the unsupervised methods perform very well on Bgl3, with EVmutation identifying the top variant with a budget of 20. Meanwhile, the supervised methods perform very well on Ube4b, prioritizing a variant with the true maximum score with a budget as small as five variants.

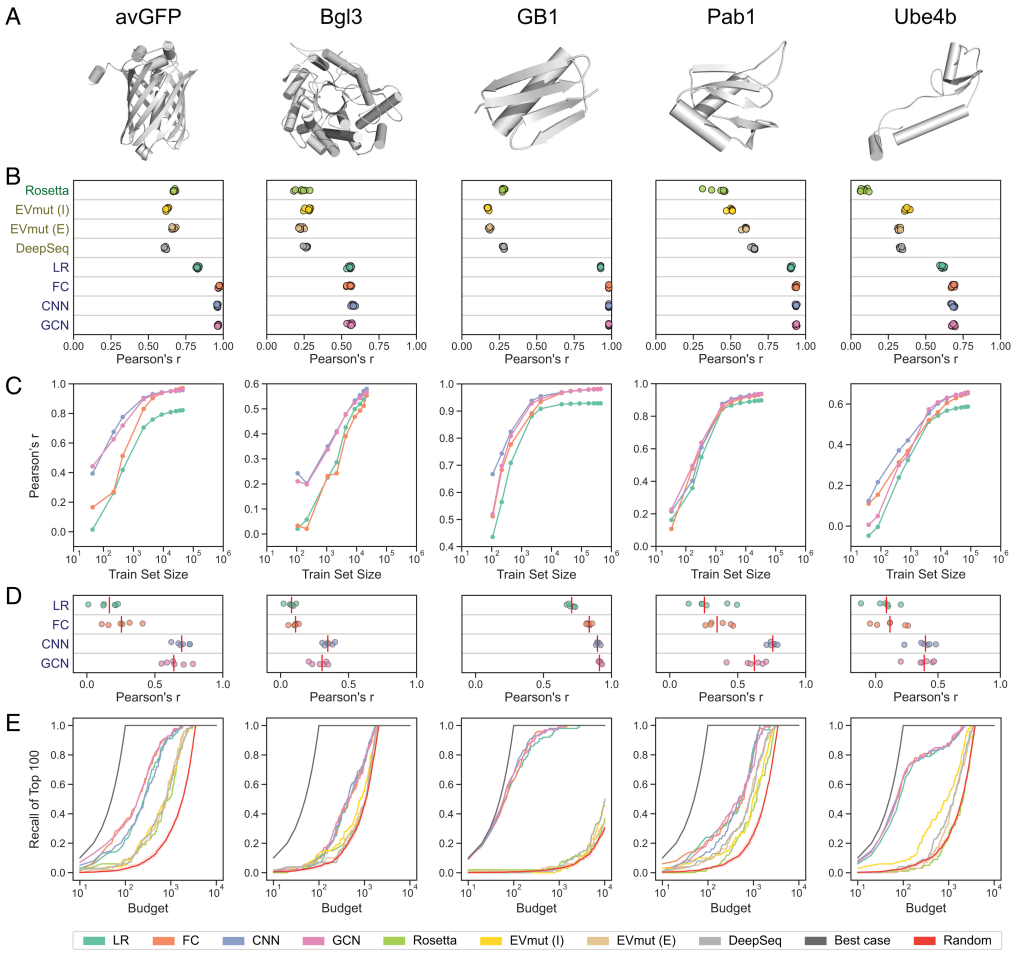

Figure 2.2: **Evaluation of neural networks and comparison with unsupervised methods.** (a) Three-dimensional protein structures. (b) Pearson's correlation coefficient between true and predicted scores for Rosetta, EVmutation, DeepSequence, linear regression (LR), fully connected network (FC), sequence convolutional network (CNN), and graph convolutional network (GCN). EVmutation (I) refers to the independent formulation of the model that does not include pairwise interactions. EVmutation (E) refers to the epistatic formulation of the model that does include pairwise interactions. Each point corresponds to one of seven random train-tune-test splits. (c) Correlation performance of supervised models trained with reduced training set sizes. (d) Model performance when making predictions for variants containing mutations that were not seen during training (mutational extrapolation). Each point corresponds to one of six replicates, and the red vertical lines denote the medians. (e) The fraction of the true 100 best-scoring variants identified by each model's ranking of variants with the given budget. The random baseline is shown with the mean and a 95% CI.

Role of Data Quality in Learning Accurate Sequence–Function Models

The performance of the supervised models varied substantially across the five protein datasets. For example, the Pearson correlation for the Bgl3 models was ~0.4 lower than the GB1 models. Although it is possible some proteins and protein families are intrinsically more difficult to model, practical considerations, such as the size and quality of the deep mutational scanning dataset, could also affect protein-specific performance. Deep mutational scanning experiments use a high-throughput assay to screen an initial gene library and isolate variants with a desired functional property. The initial library and the isolated variants are sequenced, and a fitness score is computed for each variant based on the frequency of reads in both sets. The quality of the calculated fitness scores depends on the sensitivity and specificity of the high-throughput assay, the number of times each variant was characterized in the high-throughput assay, and the number of DNA sequencing reads per variant. If any one of these factors is too low, the resulting fitness scores will not reflect the true fitness values of the characterized proteins, which will make it more difficult for a model to learn the underlying sequence to function mapping.

We assessed how experimental factors influence the success of supervised learning by resampling the full GB1 dataset to generate simulated datasets with varying protein library sizes and numbers of DNA sequencing reads. The library size is the number of unique variants screened in the deep mutational scan. The GB1 dataset is ideal for this analysis because it contains most of the possible single and double mutants and has a large number of sequencing reads per variant. We

trained sequence convolutional models on each simulated dataset and tested each network's predictions on a "true", non-resampled test set (Fig. 2.3). Models trained on simulated datasets with small library sizes performed poorly because there were not sufficient examples to learn the sequence-function mapping. This result is expected and is in line with the performance of models trained on reduced training set sizes on the original GB1 dataset (Fig. 2.2c). Interestingly, we also found that datasets with large library sizes can perform poorly if there are not sufficient DNA sequencing reads to reliably estimate the frequency of each variant. This highlights a trade-off between the number of sequence-function examples in a dataset and the quality of its fitness scores. Given a fixed sequencing budget, there exists an optimal intermediate library size that balances these two competing factors. The Bgl3 dataset's poor performance may be the result of having too many unique variants without sufficient sequencing coverage, resulting in a low number of reads per variant and therefore unreliable fitness scores. Future deep mutational scanning libraries could be designed to maximize their size and diversity while ensuring that each variant will have sufficient reads within sequencing throughput constraints.

Figure 2.3: **Trade-off between library size and number of sequencing reads.** Performance of sequence convolutional models trained on GB1 datasets that have been resampled to simulate different combinations of protein library size and number of sequencing reads in the deep mutational scan. An "X" signifies that the combination of library size and number of reads produced a dataset with fewer than 25 variants and was, therefore, excluded from the experiment. Having a large library size can be detrimental to supervised model performance if there are not enough reads to calculate reliable functional scores.

Learned Models Provide Insight into Protein Structure and Mechanism

Our neural networks transform the original amino acid features through multiple layers to map to an output fitness value. Each successive layer of the network constructs new latent representations of the sequences that capture important aspects of protein function. We can visualize the relationships between sequences in these latent spaces to reveal how the networks learn and comprehend protein function. We used Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2020) to visualize test set sequences in the latent space at the last layer of the GB1 sequence convolutional network (Fig. 2.4a). The latent space organizes the test set sequences based on their functional score, demonstrating that the network's

internal representation, which was learned to predict function of the training set examples, also generalizes to capture the sequence-function relationship of the new sequences. The latent space features three prominent clusters of low-scoring variants that may correspond to different mechanisms of disrupting GB1 function. Two clusters, referred to as "G1" and "G2", contain variants with mutations in core residues near the protein's N and C termini, respectively (Fig. A.10). Mutations at these residues may disrupt the protein's structural stability and thus decrease the activity measured in the deep mutational scanning experiment (Olson et al., 2014). Residue cluster "G3" contains variants with mutations at the IgG binding interface, and these likely decrease activity by disrupting key binding interactions. This clustering of variants based on different molecular mechanisms suggests the network is learning biologically meaningful aspects of protein function.

We can also use the neural network models to understand which sequence positions have the greatest influence on protein function. We computed integrated gradients attributions (Sundararajan et al., 2017) for all training set variants in Pab1 and mapped these values onto the three-dimensional structure (Fig. 2.4b). Pab1's sequence positions display a range of attributions spanning from negative to positive, where a negative attribution indicates that mutations at that position decrease the protein's activity. Residues at the RNA binding interface tend to display negative attributions, with the key interface residue N127 having the largest negative attribution. The original deep mutational scanning study found that residue N127 cannot be replaced with any other amino acid without significantly decreasing Pab1 binding activity (Melamed et al., 2013). Position D151 has one of the largest

positive attributions, which is consistent with the observation that aspartic acid (D) is uncommon at position 151 in naturally occurring Pab1 sequences (Melamed et al., 2013). The sequence convolutional network is able to learn biologically relevant information directly from raw sequence-function data, without the need to specify detailed molecular mechanisms.

Finally, we used the Pab1 sequence convolutional network to make predictions for all possible single-mutation variants (Fig. 2.4c). The resulting heat map highlights regions of the Pab1 sequence that are intolerant to mutations and shows that mutations to proline are deleterious across most sequence positions. It also demonstrates the network's ability to predict scores for amino acids that were not directly observed in the dataset. The original deep mutational scan characterized 1,244 single-mutation variants, yet the model can make predictions for all 1,500 possible single-mutation variants. For example, mutation F170P was not experimentally observed, but the model predicts it will be deleterious because proline substitutions at other positions are often highly deleterious. This generalization to amino acid substitutions not observed in the data is only possible with models that share parameters across sequence positions.

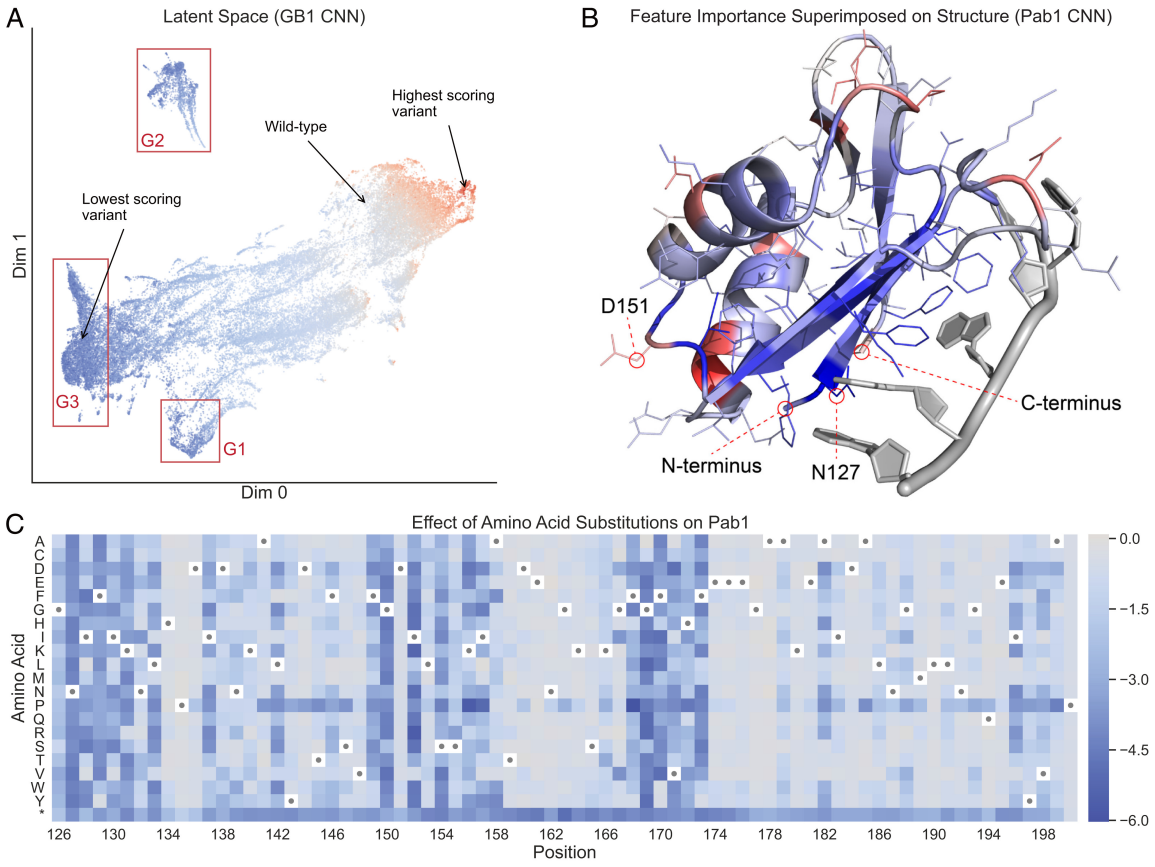

Figure 2.4: **Neural network interpretation.** (a) A UMAP projection of the latent space of the GB1 sequence convolutional network (CNN), as captured at the last internal layer of the network. In this latent space, similar variants are grouped together based on the transformation applied by the network to predict the functional score. Variants are colored by their true functional score, where red represents high-scoring variants and blue represents low-scoring variants. The clusters marked G1 and G2 correspond to variants with mutations at core residues near the start and end of the sequence, respectively. Cluster G3 corresponds to variants with mutations at surface interface residues. (b) Integrated gradients feature importance values for the Pab1 CNN, aggregated at each sequence position and superimposed on the protein's three-dimensional structure. Blue represents positions with negative attributions, meaning mutations in those positions push the network to output lower scores, and red represents positions with positive attributions. (c) A heat map showing predictions for all single mutations from the Pab1 CNN. Wild-type residues are indicated with dots, and the asterisk is the stop codon. Most single mutations are predicted to be neutral or deleterious.

Designing Distant Protein Sequences with Learned Models

Our trained neural networks describe the mapping from sequence to function for a given protein family. These models can be used to design new sequences that were not observed in the original deep mutational scanning dataset and may have improved function. The protein design process involves extrapolating a model's predictions to distant regions of sequence space. Because the models were trained and evaluated only on sequences with local changes with respect to the wild type, it is unclear how these out-of-distribution predictions will perform.

We tested the ability of our supervised models to generalize beyond the training data by designing a panel of new GB1 variants with varying distances from the wild-type sequence (Fig. 2.5a). GB1 is a small 8-kDa domain from streptococcal protein G that binds to the fragment crystallizable (Fc) domain of mammalian IgG. GB1's structure is composed of one α-helix that is packed into a four-stranded β-sheet. GB1's interaction with IgG is largely mediated by residues in the α-helix and third β-strand.

The design process was guided by an ensemble of the four models (linear regression and fully connected, sequence convolutional, and graph convolutional networks) to fully leverage different aspects of the sequence-function mapping captured by each model. We used a random-restart hill-climbing algorithm to search over GB1 sequence space for designs that maximize the minimum predicted fitness over the four models. Maximizing the minimum predicted fitness over the four models ensures that every model predicts the designed sequences to have high fitness. We applied this sequence optimization method to design five GB1

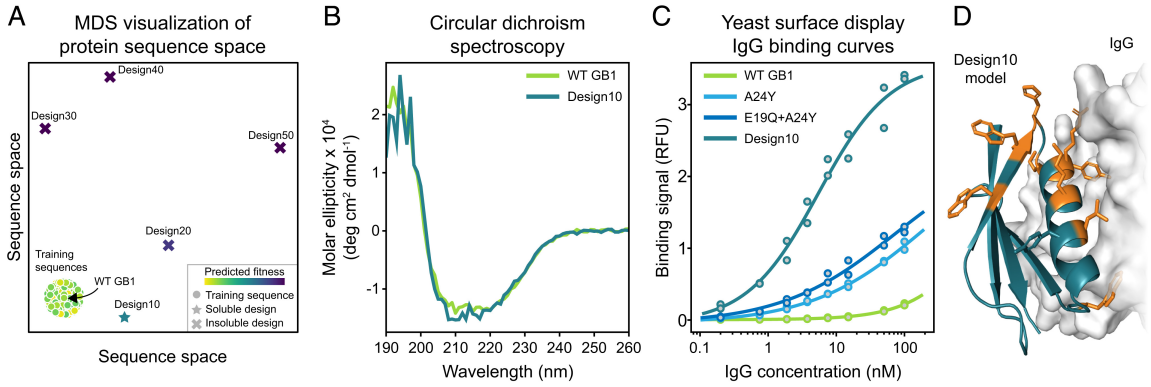

Figure 2.5: **Neural network-based protein design.** (a) Multidimensional scaling (MDS) sequence space visualization of the wild-type (WT) GB1 sequence, the GB1 training sequences, and the five designed proteins. Design10 to Design50 are progressively farther from the training distribution. Design10 is expressed as a soluble protein, while the more distant designs were insoluble. (b) Circular dichroism spectra of purified wild-type GB1 and Design10. Both proteins display highly similar spectra that are indicative of α-helical protein structures. (c) IgG binding curves of wild-type GB1 variants. Design10 displays substantially higher binding affinity than wild-type GB1, A24Y, and E19Q + A24Y. All measurements were performed in duplicate. Binding signal is reported in relative fluorescence units (RFU). (d) The locations of Design10's 10 mutations (shown in orange) relative to the IgG binding interface. The Design10 structure was predicted de novo using Rosetta.

variants with increasing numbers of mutations (10, 20, 30, 40, 50) from the wild type, representing sequence identities spanning from 82 to 11% (Table A.1). We expect designs with fewer mutations to be more likely to fold and function because they are more similar to the training data.

We experimentally tested the five GB1 designs by synthesizing their corresponding genes and expressing them in Escherichia coli. We found that the 10-mutant design, referred to as Design10, was expressed as a soluble protein, but the more distant designs were insoluble (Fig. 2.5a). We were unable to further characterize Design20 to Design50 because their insoluble expression prevented downstream protein purification. We performed circular dichroism spectroscopy on Design10

and found that it had nearly identical spectra to wild-type GB1, suggesting they have similar secondary structure content (Fig. 2.5b).

The original GB1 deep mutational scan measured binding to the Fc region of IgG; therefore, our supervised models should capture a variant's binding affinity. We tested Design10's ability to bind to IgG using a yeast display binding assay. We also tested wild-type GB1 and the top single (A24Y) and double (E19Q + A24Y) mutants from the original deep mutational scanning dataset. We found that Design10 binds to IgG with a K_d of 5 nM, which is substantially higher affinity than wild-type GB1, A24Y, or E19Q + A24Y (Fig. 2.5c). We were unable to precisely determine wild-type GB1, A24Y, or E19Q + A24Y's dissociation constants because our assay could not reliably measure binding at IgG concentrations above 100 nM. The data showed qualitative trends where wild type had the lowest affinity, followed by A24Y and then, E19Q + A24Y. Our measurements indicate that wild-type GB1's K_d is well above 100 nM, which is consistent with measurements from the literature that have found that this interaction is in the 250- to 900-nM range (Jha et al., 2014; Watanabe et al., 2019). Based on our estimates and others' previous measurements, we conservatively estimate that Design10 binds human IgG with at least 20-fold higher affinity than wild-type GB1.

Closer inspection of the Design10 sequence revealed that it was not simply composed of the top 10 single mutations for enrichment and even included 4 mutations whose individual effects on the predicted functional score ranked below the top 300. In addition, Design10's predicted score was more than two times greater than the variant comprising the top 10 single mutations. This highlights the

ability of the design process to capture nonlinear interactions and leverage synergies between sites. We also evaluated the robustness of our findings by rerunning the 10-mutant design process 100 independent times and evaluating the diversity of the designs (Table A.2).

We built a model of Design10's three-dimensional structure using Rosetta de novo structure prediction (Fig. 2.5d). Design10's predicted structure aligns to the wild-type GB1 crystal structure with 0.9Å Cα rmsd. Design10's actual structure is likely very similar to wild-type GB1 given their high sequence identity, similar circular dichroism spectra, and the small deviation between Design10's de novo predicted structure and the experimental GB1 structure. Inspection of Design10's predicted structure revealed that many of its mutations were concentrated near the IgG binding interface, and this may help to explain its large increase in IgG binding affinity. We also evaluated Rosetta models for Design20 to Design50 and found no obvious reasons why they failed to express.

2.3 Discussion

We have presented a supervised learning framework to infer the protein sequence-function mapping from deep mutational scanning data. Our supervised models work best when trained with large-scale datasets, but they can still outperform physics-based and unsupervised prediction methods when trained with only hundreds of sequence-function examples. Unsupervised methods remain appealing for proteins with very little or no sequence-function data available. Among the

supervised models, linear regression displayed the lowest performance due to its inability to represent interactions between multiple mutations. Despite that limitation, linear regression still performed fairly well because mutations often combine in an additive manner (Wells, 1990). The convolutional networks outperformed linear regression and fully connected networks when trained with fewer training examples and when performing mutational extrapolation. The parameter sharing inherent to convolutional networks can improve performance by allowing generalization of the effects of mutations across different sequence positions. However, in the five datasets we tested, even the convolutional networks could not accurately generalize when entire sequence positions were excluded from the training data. It was surprising that graph convolutions that incorporate protein structure did not improve performance over sequence-based convolutions. The comparable performance could be the result of the networks' ability to compensate with fully connected layers, the lack of sequence diversity in the deep mutational scanning data, or the specific type of graph neural network architecture used. We are unable to determine which of these factors had the greatest influence.

Our analysis of how data quality influences the ability to learn the sequence-function mapping can be considered when designing future deep mutational scanning experiments. We found that a model's predictive performance is determined not only by the number of sequence-function training examples but also by the quality of the estimated functional scores. Therefore, in a deep mutational scanning experiment, it may be preferable to limit the total number of unique variants analyzed to ensure that each variant has sufficient sequencing reads to calculate

accurate functional scores. Any missing mutations can then be imputed with a convolutional network to overcome the smaller dataset size.

Recent studies have examined supervised learning methods capable of scaling to deep mutational scanning datasets. One study benchmarked combinations of supervised learning methods and protein sequence encodings (Xu et al., 2020). Consistent with our results, it found that sequence convolutional neural networks with amino acid property-based features tended to perform better than alternatives. Some algorithms specialize in modeling epistasis. Epistatic Net (Aghazadeh et al., 2020) introduced a neural network regularization strategy to limit the number of epistatic interactions. Other approaches focused on the global epistasis that arises due to a nonlinear transformation from a latent phenotype to the experimentally characterized function (Tareen et al., 2020; Otwinowski et al., 2018). Protein engineering with UniRep (Biswas et al., 2021) showed that general global protein representations can support training function-specific supervised models with relatively few sequence-function examples. ECNet pioneered an approach for combining global protein representations, local information about residue coevolution, and protein sequence features (Luo et al., 2020). Across tens of deep mutational scanning datasets, ECNet was almost always superior to unsupervised learning models and models based only on a global protein representation. Future work can explore how to best combine global protein representations, local residue coevolution features, and graph encodings of protein structure to learn predictive models for specific protein functions, including for proteins that have little experimental data available. Despite their similar performance to sequence convolutional net-

works in our study, graph convolutional networks that integrate three-dimensional structural information remain enticing because of successes on other protein modeling tasks (Fout et al., 2017; Gligorijevic et al., 2020; Strokach et al., 2020; Sanyal et al., 2020) and rapid developments in graph neural network architectures (Wu, Zonghan et al., 2020).

Another challenging future direction will be assessing how well trained models extrapolate to sequences with higher-order mutations (Mater et al., 2020; Bryant et al., 2021). As a proof of concept, we designed distant GB1 sequences with tens of mutations from the wild type. The 10-mutant design (Design10) had substantially stronger IgG binding affinity than wild-type GB1, but the four sequences with more mutations did not express as soluble proteins. The tremendous success of Design10 is encouraging considering how few designed sequences we tested and the many opportunities to improve upon our limited exploration of model-guided design. The model predictions can be improved through more sophisticated ensembling and uncertainty estimation. Our hill-climbing sequence optimization strategy can be replaced by specialized methods that allow supervised models to efficiently explore new parts of a sequence space (Angermueller et al., 2020; Fannjiang and Listgarten, 2020; Brookes et al., 2019; Linder and Seelig, 2021).

Machine learning is revolutionizing our ability to model and predict the complex relationships between protein sequence, structure, and function (Yang et al., 2019; Torrisi et al., 2020). Supervised models of protein function are currently limited by the availability and quality of experimental data but will become increasingly accurate and general as researchers continue to experimentally characterize protein

sequence space (Esposito et al., 2019). Other important machine learning advances relevant to protein engineering include generative modeling to sample nonnatural protein sequences (Hawkins-Hooker et al., 2020; Madani et al., 2020; Strokach et al., 2020), language models to learn protein representations from diverse natural sequences (Asgari and Mofrad, 2015; Yang et al., 2018; Alley et al., 2019; Rives et al., 2020), and strategies to incorporate machine learning predictions into directed evolution experiments (Biswas et al., 2018; Saito et al., 2018; Wittmann et al., 2020). These approaches are enabling the next generation of data-driven protein engineering.

2.4 Methods

Datasets

We tested our supervised learning approach on five deep mutational scanning datasets: avGFP (Sarkisyan et al., 2016), Bgl3 (Romero et al., 2015), GB1 (Olson et al., 2014), Pab1 (Melamed et al., 2013), and Ube4b (Starita et al., 2013). We selected these publicly available datasets because they correspond to diverse proteins and contain variants with multiple amino acid substitutions. The avGFP, Pab1, and Ube4b datasets were published with precomputed functional scores, which we used directly as the target scores for our method. For GB1 and Bgl3, we computed functional scores from raw sequencing read counts using Enrich2 (Rubin et al., 2017). We filtered out variants with fewer than five sequencing reads and ran Enrich2 using the "Log Ratios (Enrich2)" scoring method and the "Wild Type"

normalization method. Table 2.1 shows additional details about the datasets.

Protein Sequence Encoding

We encoded each variant's amino acid sequence using a sequence-level encoding that supports multiple substitutions per variant. Each amino acid is featurized with its own feature vector, and the full encoded variant consists of the concatenated amino acid feature vectors. We featurize each amino acid using a two-part encoding made up of a one-hot encoding and an amino acid index (AAIndex) encoding. One-hot encoding captures the specific amino acid at each position. It consists of a length 21 vector where each position represents one of the possible amino acids or the stop codon. All positions are zero except the position of the amino acid being encoded, which is set to a value of one. AAindex encoding captures physicochemical and biochemical properties of amino acids from the AAindex database (Kawashima et al., 2008). These properties include simple attributes, such as hydrophobicity and polarity, as well as more complex characteristics, such as average nonbonded energy per atom and optimized propensity to form a reverse turn. In total, there are 566 such properties that were taken from literature. These properties are partially redundant because they are aggregated from different sources. Therefore, we used principle component analysis to reduce the dimensionality to a length 19 vector, capturing 100% of the variance. We concatenated the one-hot and AAindex encodings to form the final representation for each amino acid. One benefit of this encoding is that it enables the use of convolutional networks, which leverage the spatial locality of the raw inputs to learn higher-level features via filters. Other

types of encodings that do not have a feature vector for each residue, such as those that embed full amino acid sequences into fixed-size vectors, would not be as appropriate for convolutional networks because they do not have locality in the input that can be exploited by convolutional filters.

Convolutional Neural Networks

We tested two types of convolutional neural networks: sequence convolutional and graph convolutional. These networks extract higher-level features from the raw inputs using convolutional filters. Convolutional filters are sets of learned weights that identify patterns in the input and are applied across different parts of the input. The filters can output higher or lower values depending on whether the given input matches the pattern that the filters have learned to identify. We implemented a sequence convolutional network where the input is a one-dimensional amino acid sequence. The network applies filters using a sliding window across the input sequence, integrating information from amino acid sequence neighbors. The network applies filters at all valid sequence positions and does not pad the ends of the sequence with zeros.

Graph convolutional neural networks are similar to traditional convolutional networks, except graph convolutional networks operate on arbitrary graph structures rather than linear sequences or two-dimensional grids. Graph filters still capture spatial relationships in the input data, but those relationships are determined by neighboring nodes in the graph rather than neighboring characters in a sequence or neighboring pixels in a two-dimensional grid. In our case, we use a graph derived

from the protein's wild-type three-dimensional structure. This allows the network to more easily learn features that correspond to patterns of amino acid residues that are nearby in physical space.

We use the order-independent graph convolution operator described by Fout et al. (Fout et al., 2017). It is considered order independent because it does not impose an ordering on neighbor nodes. In an order-dependent formulation, different neighbor nodes would have different weights, but in the order-independent formulation, all neighbor nodes are treated identically and share the same weights. Each filter consists of a weight vector for the center node and a weight vector for the neighbor nodes that is shared among the neighbor nodes. For a set of filters, the output z_i at a center node i is calculated using Equation 2.1, where W_C is the center node's weight matrix, W_N is the neighbor nodes' weight matrix, and b is the vector of biases, one for each filter. Additionally, x_i is the feature vector at node i, N_i is the set of neighbors of node i, and σ is the activation function.

$$z_i = \sigma\Big(W_C \cdot x_i + \frac{1}{|N_i|} \sum_{j \in N_i} (W_N \cdot x_j) + b\Big) \tag{2.1}$$

In this formulation, a graph consisting of nodes and edges is incorporated into each convolutional layer. Input features are placed at the graph's nodes in the first layer. Outputs are computed at the node level using input features from a given center node and corresponding neighbor nodes. Because output is computed for each node, graph structure is preserved between subsequent graph layers. The incoming signal from neighbor nodes is averaged to account for the variable numbers of neighbors. The window size of the filter is limited to the immediate

neighbors of the current center node. Information from more distant nodes is incorporated through multiple graph convolutional layers. The final output of the network is computed at the graph level with a single function score prediction for the entire graph.

Protein Structure as a Graph

We encoded each protein's wild-type structure as a graph and incorporated it into the architecture of the graph convolutional neural network (Fig. 2.1d). The protein structure graph is an undirected graph with a node for each amino acid residue and an edge between nodes if the residues are within a specified distance threshold in three-dimensional space. The distance threshold is a hyperparameter with a range of 4 to 10Å and was selected independently for each dataset during the hyperparameter optimization. We measure distances between residues via distances of the β-carbon atoms (Cβ) in angstroms. The protein structure graph for GB1 is based on Protein Data Bank (PDB) structure 2QMT. The protein structure graphs for the other four proteins are based on structures derived from Rosetta comparative modeling, using the RosettaCM protocol (Song et al., 2013) with the default options. For the comparative modeling, we selected template structures from PDB that most closely matched the reference sequence of the deep mutational scanning data. In addition to the standard graph based on the protein's structure, we tested four baseline graphs: a shuffled graph based on the standard graph but with shuffled node labels, a disconnected graph with no edges, a sequential graph containing only edges between sequential residues, and a complete graph

containing all possible edges (Fig. A.6). We used NetworkX (Hagberg et al., 2008) v2.3 to generate all protein structure and baseline graphs.

Complete Model Architectures

We implemented linear regression and three types of neural network architectures: fully connected, sequence convolutional, and graph convolutional. Linear regression is implemented as a fully connected neural network with no hidden layers. It has a single output node that is fully connected to all input nodes. The other networks all have multiple layers. The fully connected network consists of some number of fully connected layers, and each fully connected layer is followed by a dropout layer with a 20% dropout probability. Finally, there is a single output node. The sequence and graph convolutional networks consist of some number of convolutional layers, a single fully connected layer with 100 hidden nodes, a dropout layer with a 20% dropout probability, and a single output node. We also trained sequence and graph convolutional networks without the fully connected layer or dropout layer for the analyses in Figure A.7. We used the leaky rectified linear unit as the activation function for all hidden layers. A hyperparameter sweep determined the other key aspects of the model architectures, such as the number of layers, the number of filters, and the kernel size of filters (Fig. A.11). We used Python v3.6 and TensorFlow (Abadi et al., 2015) v1.14 to implement the models.

Model Training

We trained the networks using the Adam optimizer and mean squared error loss. We set the Adam hyperparameters to the defaults except for the learning rate and batch size, which were selected using hyperparameter sweeps. We used early stopping for all model training with a patience of 15 epochs and a minimum delta (the minimum amount by which loss must decrease to be considered an improvement) of 0.00001. We set the maximum possible number of epochs to 300. The original implementation overweighted the last examples in an epoch when calculating the tuning set loss. This could have affected early stopping but had little to no effect in practice. We trained the networks on graphics processing units (GPUs) available at the University of Wisconsin-Madison via the Center for High Throughput Computing and the workload management system HTCondor (Thain et al., 2005). We also used GPU resources from Argonne National Laboratory's Cooley cluster. The GPUs we used included Nvidia GeForce GTX 1080 Ti, GeForce RTX 2080 Ti, and Tesla K80.

Main experiment setup

We split each dataset into random training, tuning, and testing sets. The tuning set is sometimes referred to as the validation set and is used for hyperparameter optimization. This allowed us to train the models; tune hyperparameters; and evaluate performance on separate, nonoverlapping sets of data. The training set was 81% of the data, the tuning set was 9%, and the testing set was 10%. This strategy supports the objective of training and evaluating models that fully leverage

all available sequence-function data and make predictions for variants that have characteristics similar to the training data. There are other valid strategies that more directly test the ability of a model to generalize to mutations or positions that were not present in the training data, which we describe below.

We performed a hyperparameter grid search for each dataset using all possible combinations of the hyperparameters in Figure A.11. The hyperparameters selected for one dataset did not influence the hyperparameters selected for any other dataset. For each type of supervised model (linear regression, fully connected, sequence convolutional, and graph convolutional), we selected the set of hyperparameters that resulted in the smallest mean squared error on the tuning set. The selected hyperparameters are listed in Table A.3, and the number of trainable parameters in each selected model is listed in Table A.4. This is the main set of hyperparameters. For any subsequently trained models, such as those with reduced training set sizes, we performed smaller hyperparameter sweeps to select a learning rate and batch size, but all other hyperparameters that specify the network architecture were set to those selected in the main run.

To assess the robustness of the original train-tune-test splits, we created six additional random splits for each dataset. We tuned the learning rate and batch size independently for each split; however, the network architectures were fixed to those selected using the original split. There was a risk of overestimating performance on the new splits because the data used to tune the architectures from the original split may be present in the test sets of the new splits. However, the results showed no evidence of this type of overfitting. We report the performance on these new splits

in Figures 2.2b and A.1a. All other experiments used the original train-tune-test split.

For the random baseline in Figures A.8 and A.9, we generated 1,000 random rankings of the entire test set. Then, for each ranking threshold N, we selected the first N variants from each ranking as the prioritized variants. We computed the mean (Fig. A.8) and the maximum (Fig. A.9) of each random ranking's prioritized variants. Finally, we show the 95% confidence interval calculated as $\pm$ 1.96 times the SD.

Reduced Training Size Setup

For the reduced training size experiment, we used the same 9% tuning and 10% testing sets as the main experiment. The reduced training set sizes were determined by percentages of the original 81% training pool. For each reduced size, we sampled five random training sets of the desired size from the 81% training pool. These replicates are needed to mitigate the effects of an especially strong or weak training set, which could be selected by chance, especially with the smaller training set sizes. We reported the median metrics of these five replicates.

Mutational and Positional Extrapolation

We tested the ability of the models to generalize to mutations and positions not present in the training data using two dataset splitting strategies referred to as mutational and positional extrapolation. For each of these splitting strategies, we created six replicate train-tune-test splits and tuned the learning rate and batch size

independently for each split. We report the Pearson's correlation on the test set for each split in Figures 2.2d and A.5.

For mutational extrapolation, we designated 80% of single mutations present in the dataset as training and 20% as testing. We then divided the variants into three pools: training, testing, or overlap, depending on whether the variants contained only mutations designated as training, only mutations designated as testing, or mutations from both sets. We discarded the variants in the overlap pool to ensure there was no informational overlap between the training and testing data. We split the training pool into a 90% training set and a 10% tuning set. We used 100% of the variants in the testing pool as the test set.

For positional extrapolation, we followed a similar procedure as mutational extrapolation. We designated 80% of sequence positions as training and 20% as testing. We divided variants into training, testing, and overlap pools, depending on whether the variants contained mutations only in positions designated as training, only in positions designated as testing, or both in positions designated as training and testing. We discarded the variants in the overlap pool, split the training pool into a 90% training set and a 10% tuning set, and used 100% of the variants in the testing pool as the test set.

Comparison to EVmutation and DeepSequence

We generated multiple sequence alignments using the EVcouplings web server (Hopf et al., 2019) according to the protocol described for EVmutation (Hopf et al., 2017). We used Jackhmmer (Eddy, 2011) to generate an initial alignment with five

search iterations against UniRef100 (Suzek et al., 2015) and a sequence inclusion threshold of 0.5 bits per residue. If the alignment had $< 80\%$ sequence coverage, we increased the threshold in steps of 0.05 bits per residue until coverage was $\geqslant 80\%$. If the number of effective sequences in the alignment was < 10 times the length of the sequence, we decreased the threshold until the number of sequences was $\geqslant 10$ times the length. If the objectives were conflicting, we gave priority to the latter. We set all other parameters to EVcouplings defaults. We trained EVmutation via the "mutate" protocol from EVcouplings. We executed EVmutation locally using EVcouplings v0.0.5 with configuration files generated by the EVcouplings web server. We trained DeepSequence using the same fixed architecture and hyperparameters described in the original work (Riesselman et al., 2018). We fit a DeepSequence model to each alignment and calculated the mutation effect prediction using 2,000 evidence lower-bound samples.

Comparison to Rosetta

We computed Rosetta scores for every variant using Rosetta's FastRelax protocol with the talaris2014 score function (Rosetta v3.10). First, we created a base structure for each wild-type protein. We generated 10 candidate structures by running relax on the same structure used to generate the protein structure graph, described above. We selected the lowest-energy structure to serve as the base. Next, we ran mutate and relax to generate a single structure and compute the corresponding energy for each variant. We set the residue selector to a neighborhood of 10Å. We took the negative of the computed energies to compute the final score for each variant.

GB1 Resampling Experiment

We performed a resampling experiment on the GB1 dataset to assess how the quality of deep mutational scanning-derived functional scores impacts performance of supervised learning models. In this case, quality refers to the number of sequencing reads per variant used to estimate the fitness scores. The number of reads per variant depends on the number of variants and the total number of reads in the deep mutational scanning experiment. Raw deep mutational scanning data consist of two sets of variants: an input set and a selected set. Both sets have associated sequencing read counts for each variant, and the functional score for each variant is calculated from these read counts. We resampled the original GB1 data to generate datasets corresponding to 99 different combinations of protein library size and number of reads (Fig. A.12). The library size refers to the number of unique variants being screened in the deep mutational scan. Note that the final dataset may have fewer unique variants than the protein library. This occurs when there is a low number of sequencing reads relative to the size of the library. In that scenario, not all generated variants will get sequenced, even though they were screened as part of the function assay.

First, we created a filtered dataset by removing any variants with zero reads in either the input set or selected set of the original deep mutational scanning data. We generated Enrich2 scores for this filtered dataset using the same approach described in the above section on datasets. We randomly selected 10,000 variants from this dataset to serve as a global testing set. Next, we used the filtered dataset, minus the testing set variants, as a base to create the resampled datasets. For each library size

in the heat map in Figure 2.3, we randomly selected that many variants from the base dataset to serve as the library. Then, for each library, we created multinomial probability distributions giving the probability of generating a read for a given variant. We created these probability distributions for both the input and selected sets by dividing the original read counts of each variant by the total number of reads in the set. The multinomial distributions allowed us to sample new input and selected sets based on the read counts in the heat map in Figure 2.3. To determine how many reads should be sampled from the input set vs. the selected set, we computed the fraction of reads in the input set and selected set in the base dataset and sampled reads based on that fraction. Finally, we generated Enrich2 scores for each resampled dataset using the same approach described in the above section on datasets. To account for potential skewing from random sampling, we generated five replicates for each of the 99 combinations of library size and numbers of reads. Counting the replicates, we created 495 resampled datasets in total.

We trained the supervised learning models on each resampled dataset, as long as the dataset had at least 25 total variants in each of its five replicates. Out of the 99 combinations of library size and number of reads, 7 did not have enough variants across the replicate datasets and were thus excluded from this experiment. Although the libraries of these 7 combinations had more than 25 variants, there were not enough reads to estimate scores for all of them, and thus, the final datasets ended up with less than 25 variants. We split each resampled dataset into 80% training and 20% tuning sets. The tuning sets were used to select the learning rate and batch size hyperparameters. The network architectures and other parameters were set to

those selected during the main experiment described above. We evaluated each model using the held-out testing set with non-resampled fitness scores. This type of evaluation ensures that although the models are trained on resampled datasets with potentially unreliable fitness scores, they are evaluated on high-confidence fitness scores from the non-resampled dataset. We report the mean Pearson's correlation coefficient across the five replicates for each combination of library size and number of reads.

UMAP Projection of Latent Space

Each neural network encodes a latent representation of the input in its last internal layer before the output node. The last internal layer in the convolutional networks is a dense fully connected layer with 100 hidden nodes. Thus, the latent representation of each variant at this layer is a length 100 vector. We used UMAP (McInnes et al., 2020) to project the latent representation of each variant into a two-dimensional space to make it easier to visualize while still preserving spatial relationships between variants. We used the umap-learn package v0.4.0 to compute the projection with default hyperparameters (n_neighbors=15, min_dist=0.1, and metric="euclidean"). The two-dimensional visualization shows how the network organizes variants internally prior to predicting a functional score. We colored each variant by its score to show that the network efficiently organizes the variants. Variants grouped close together in the UMAP plot have similar functional scores. We also annotated a few key variants, such as the highest- and lowest-scoring variants.

Integrated gradients

To determine which input features were important for making predictions, we generated integrated gradients feature attributions (Sundararajan et al., 2017) for all variants. The attributions quantify the effects of specific feature values on the network's output. A positive attribution means the feature value pushes the network to output a higher score relative to the given baseline, and a negative attribution means the feature value pushes the network to output a lower score relative to the given baseline. We used the wild-type sequence as the baseline input. Integrated gradients attributions are computed on a per-variant basis, meaning attributions are specific to the feature values of the given variant. Due to nonlinear effects captured by the nonlinear models, a given feature value might have a positive attribution in one variant but a negative attribution in a different variant. We computed attributions for all variants in the training set. Examining the training set is analogous to other model interpretation techniques that compute attributions directly from the weights or parameters of models that were trained using training sets. We summed the attributions for all features at each sequence position, allowing us to see which mutations pushed the network to output a higher or lower score for each individual variant. We also summed the attributions across all the variants in the training set to see which sequence positions were typically tolerant or intolerant to mutations. We used DeepExplain (Ancona et al., 2018) v0.3 to compute the integrated gradients attributions with "steps" set to 100.

Model-Guided Design of GB1 Variants

We used a random-restart hill-climbing algorithm to design sequences with a set number of mutations (n) from wild-type GB1 that maximized the minimum predicted functional score from an ensemble of four models (linear regression, fully connected, sequence convolutional, and graph convolutional):

$$\arg\max_{x \in \mathbb{S}^n} \min_{model \in LR,FC,CNN,GCN} f_{model}(x),$$

where x is a sequence, $\mathbb{S}^n$ is the set of all sequences n mutations from wild-type, and $f_{model}(x)$ is a model's predicted score for sequence x. This design objective ensures that all models predict that the sequence will have a high functional score. We initialized a hill-climbing run with a randomly selected sequence containing n point mutations and performed a local search by exchanging each of these n mutations with each other possible single-point mutation. Exchanging mutations ensured that we only search sequences a fixed distance n from the wild type. We then moved to the mutation-exchanged variant with the highest objective, which became our new reference point, and repeated this hill climbing process until a local optimum was reached. We performed each sequence optimization with 10 random initializations and took the design with the highest overall objective value. We applied this procedure to design one sequence at each level of diversity, where $n = 10, 20, 30, 40, 50$. We visualized the sequence space using multidimensional scaling with the Hamming distance as the distance metric between sequences.

We predicted the three-dimensional structure of Design10 using Rosetta Abinitio

(Simons et al., 1999). We used the Rosetta Fragment Server to generate the fragments for the Design10 sequence. We generated 100 structures using Rosetta 3.12 and AbinitioRelax and selected the structure with the lowest total score. The predicted structure for Design10 aligns to the wild-type GB1 crystal structure with 0.9 Å C_α rmsd. The experimental methods to characterize these designs are described in SI Appendix.

Designed GB1 Variant Gene Synthesis and Protein Expression

We designed the genes encoding the designed GB1 variants by making codon substitutions into the base wild-type GB1 gene sequence. If there were multiple codon options for an amino acid, we chose the particular codon randomly from a set of 31 codons that are optimized for expression in E. coli (Boël et al., 2016). For our expression construct, we included an upstream bicistronic design (BCD) element to minimize any influence of mRNA secondary structure on protein expression (Mutalik et al., 2013) and also included an N-terminal 6x His-tag with a five-amino-acid linker for protein purification. We ordered wild-type GB1 and the five designed GB1 variants from Twist Biosciences cloned into the pET21(+) protein expression vector.

We expressed wild-type GB1 and the five designed GB1 variants using a standard T7 expression system. We transformed the six plasmids into BL21(DE3) E. coli cells. We expressed the GB1 variants by inoculating LB cultures containing 100 μg/mL carbenicillin with a 1:100 dilution of overnight cultures, incubating these cultures shaking at 37°C until they reached an OD600 of 0.4-0.6, and inducing

with 400 μM Isopropyl β-D-1-thiogalactopyranoside (IPTG). We then incubated these expression cultures overnight at 20°C while shaking, pelleted the cells by centrifuging at 3000 g for 20 minutes at 4°C, and stored the cell pellets at -80°C.

We determined the level of soluble protein expression using sodium dodecyl sulphate–polyacrylamide gel electrophoresis (SDS-PAGE). We thawed the cell pellets on ice and resuspended into 0.5 mL of Buffer A (20 mM sodium phosphate pH 7.3, 500 mM NaCl, 20 mM imidazole). We then added 2.5 mL of lysis buffer (Buffer A + 0.60x BugBuster + 2 U/mL DNaseI (Thermo Fischer) + 1 mg/mL hen egg white lysozyme) to each sample and incubated at room temperatures for 5 minutes to yield the total cell lysate. We obtained the soluble protein fraction by centrifuging at 21,000 g for 70 minutes and extracting the supernatant. We then ran samples of the total cell lysate and soluble fractions on a Novex 4-20% Tris-Glycine SDS-PAGE gel (Thermo Fischer). After staining, we analyzed the gels to qualitatively evaluate whether the expressed proteins were present in the soluble fraction

Protein Purification and Circular Dichroism Spectroscopy

We expressed wild-type GB1 and Design10 using the above protocol, with the exception that the expression cultures were incubated at 16°C for 24 hours. We thawed the cell pellets on ice, resuspended in 2.5 mL of Buffer A, sonicated for 1 minute with 5 second pulses spaced by 15 second resting periods, and centrifuged for 10 minutes at 21,000 g to obtain the soluble protein fraction. We then ran the soluble fraction over a Ni Sepharose 6 Fast Flow column (Cytiva Life Sciences) that

was equilibrated with Buffer A, washed with 3 column volumes of Buffer A, and eluted in 1.5 mL fractions of Buffer B (20 mM sodium phosphate pH 7.3, 500 mM NaCl, 500 mM imidazole). We ran the elution fractions over SDS-PAGE and pooled the fractions that contained the target protein. Finally, we aliquoted the purified protein, flash froze the aliquots in liquid nitrogen, and stored at -80°C.

For circular dichoism (CD) spectroscopy, we thawed the purified protein samples on ice and dialyzed overnight in 20 mM sodium phosphate pH 8.0 at 4°C to remove imidazole. We then determined the protein concentrations using a Nanodrop spectrophotometer. The CD measurements were then performed by UW-Madison's Biophysics Instrumentation Facility. They measured CD spectra using a 1 mm pathlength on an AVIV Model 420 Circular Dichroism Spectrometer at 4°C. The CD spectra for Design10 was normalized to the wild-type GB1 spectra at 222 nm.

GB1 Yeast Display Plasmid Construction and Flow Cytometric IgG Binding Affinity Titration

We synthesized wild-type and Design10 variant GB1 genes as yeast codon-optimized gBlocks (Integrated DNA Technologies, Coralville, IA). The gBlocks were ligated into the unique NheI and BamHI sites of the yeast surface display vector pCTCON2 (provided by Dane Wittrup, MIT). We synthesized A24Y and E19Q+A24Y variant GB1 genes as yeast-optimized gene fragments (Twist, San Francisco, CA). The gene fragments were ligated into a golden-gate compatible version of pCTCON2 at the NheI, BamHI sites. This yeast display vector fuses the Aga2p protein to the

N-terminus of GB1.

We transformed plasmid DNA into yeast display *Saccharomyces cerevisiae* strain EBY100 made competent using the Zymo Research Frozen EZ Yeast Transformation II kit with transformants grown on synthetic dropout (SD) -Trp (MP Biomedicals, Irvine, CA) agar plates for two days at 30°C. After two days, individual colonies were picked into 4 mL of low-pH Sabouraud Dextrose Casamino Acid media (20 g/L dextrose, 6.7 g/L yeast nitrogen base, 5 g/L casamino acids, 10.4 g/L sodium citrate, 7.4 g/L citric acid monohydrate) and grown overnight at 30°C and 250 rpm. For induction of GB1 display, we started a 5 mL Sabouraud Galactose Casamino Acid (8.6 g/L $NaH_2PO^*H_2O$, 5.4 g/L Na_2HPO_4, 20 g/L galactose, 6.7 g/L yeast nitrogen base, 5 g/L casamino acids) culture at an optical density, as measured at 600 nm, of 0.5 and shook overnight at 250 rpm and 20°C.

We harvested approximately 2×10^5 yeast cells for each titration data point by centrifugation after overnight incubation, washed them once in pH 7.4 Phosphate Buffered Saline (PBS) containing 0.2% (w/v) bovine serum albumin (BSA), and incubated them overnight at 4°C on a tube rotator at 18 rpm in between 100 μL and 800 μL of PBS/0.2% BSA containing various concentrations of mouse IgG2a (BioLegend, San Diego, CA) that had been conjugated with Alexa647 using NHS chemistry (Molecular Probes, Eugene, OR). Volumes of Alexa647 IgG-containing incubation solution were varied to prevent ligand depletion from occurring in the lowest IgG concentration incubation tubes. Following overnight incubation, yeast were washed once in PBS/0.2% BSA and resuspended in ice cold PBS for flow cytometric analysis. Analyses were performed using a Fortessa analyzer (Becton

Dickinson), and the mean of the fluorescence distribution was reported.

We performed duplicate fluorescence measurements for all nine IgG concentrations tested. We then fit a Hill function to the average of these duplicate measurements. We were able to determine the K_d of Design10 as 5 nM because the binding curve was beginning to display saturation. We were unable to determine the K_d of wild-type, A24Y, or E19Q+A24Y GB1 variants because the proteins were less than 50% bound at the highest IgG concentration tested.

Data Availability

We provide a cleaned version of our code that can be used to retrain the models from this article or train new models with different network architectures or for different datasets. We also provide pretrained models that use the latest code and are functionally equivalent to the ones from this article. The pretrained models can be used to make predictions for new variants. Our code is freely available on GitHub and is licensed under the MIT license (`https://github.com/gitter-lab/nn4dms`). The software is also archived on Zenodo (`https://doi.org/10.5281/zenodo.4118330`). Table A.5 shows software dependencies and their versions.

Acknowledgements

We thank Zhiyuan Duan for his assistance running Rosetta and Darrell McCaslin at the University of Wisconsin-Madison Biophysics Instrumentation Facility for his expertise in collecting and analyzing the circular dichroism spectra. This research

was supported by National Institutes of Health award R35GM119854, National Institutes of Health award R01GM135631, National Institutes of Health training grant T32HG002760, a Predoctoral Fellowship from the PhRMA Foundation, the John W. and Jeanne M. Rowe Center for Research in Virology at the Morgridge Institute for Research, and the Brittingham Fund and the Kemper K. Knapp Bequest through the Sophomore Research Fellowship at UW-Madison. In addition, this research benefited from the use of credits from the National Institutes of Health Cloud Credits Model Pilot, a component of the Big Data to Knowledge program, and used resources of the Argonne Leadership Computing Facility, which is a Department of Energy Office of Science User Facility supported under contract DE-AC02-06CH11357. The research was performed using the compute resources and assistance of the University of Wisconsin-Madison Center for High Throughput Computing in the Department of Computer Sciences.

APPENDIX A: SUPPLEMENTARY INFORMATION FOR CHAPTER 2

2.A Supplementary Methods

Designed GB1 Variant Gene Synthesis and Protein Expression

We designed the genes encoding the designed GB1 variants by making codon substitutions into the base wild-type GB1 gene sequence. If there were multiple codon options for an amino acid, we chose the particular codon randomly from a set of 31 codons that are optimized for expression in E. coli (Boël et al., 2016). For our expression construct, we included an upstream bicistronic design (BCD) element to minimize any influence of mRNA secondary structure on protein expression (Mutalik et al., 2013) and also included an N-terminal 6x His-tag with a five-amino-acid linker for protein purification. We ordered wild-type GB1 and the five designed GB1 variants from Twist Biosciences cloned into the pET21(+) protein expression vector.

We expressed wild-type GB1 and the five designed GB1 variants using a standard T7 expression system. We transformed the six plasmids into BL21(DE3) E. coli cells. We expressed the GB1 variants by inoculating LB cultures containing 100 μg/mL carbenicillin with a 1:100 dilution of overnight cultures, incubating these cultures shaking at 37°C until they reached an OD600 of 0.4-0.6, and inducing with 400 μM Isopropyl β-D-1-thiogalactopyranoside (IPTG). We then incubated these expression cultures overnight at 20°C while shaking, pelleted the cells by centrifuging at 3000 g for 20 minutes at 4°C, and stored the cell pellets at -80°C.

We determined the level of soluble protein expression using sodium dodecyl sulphate–polyacrylamide gel electrophoresis (SDS-PAGE). We thawed the cell pellets on ice and resuspended into 0.5 mL of Buffer A (20 mM sodium phosphate pH 7.3, 500 mM NaCl, 20 mM imidazole). We then added 2.5 mL of lysis buffer (Buffer A + 0.60x BugBuster + 2 U/mL DNaseI (Thermo Fischer) + 1 mg/mL hen egg white lysozyme) to each sample and incubated at room temperatures for 5 minutes to yield the total cell lysate. We obtained the soluble protein fraction by centrifuging at 21,000 g for 70 minutes and extracting the supernatant. We then ran samples of the total cell lysate and soluble fractions on a Novex 4-20% Tris-Glycine SDS-PAGE gel (Thermo Fischer). After staining, we analyzed the gels to qualitatively evaluate whether the expressed proteins were present in the soluble fraction

Protein Purification and Circular Dichroism Spectroscopy

We expressed wild-type GB1 and Design10 using the above protocol, with the exception that the expression cultures were incubated at 16°C for 24 hours. We thawed the cell pellets on ice, resuspended in 2.5 mL of Buffer A, sonicated for 1 minute with 5 second pulses spaced by 15 second resting periods, and centrifuged for 10 minutes at 21,000 g to obtain the soluble protein fraction. We then ran the soluble fraction over a Ni Sepharose 6 Fast Flow column (Cytiva Life Sciences) that was equilibrated with Buffer A, washed with 3 column volumes of Buffer A, and eluted in 1.5 mL fractions of Buffer B (20 mM sodium phosphate pH 7.3, 500 mM NaCl, 500 mM imidazole). We ran the elution fractions over SDS-PAGE and pooled

the fractions that contained the target protein. Finally, we aliquoted the purified protein, flash froze the aliquots in liquid nitrogen, and stored at -80°C.

For circular dichoism (CD) spectroscopy, we thawed the purified protein samples on ice and dialyzed overnight in 20 mM sodium phosphate pH 8.0 at 4°C to remove imidazole. We then determined the protein concentrations using a Nanodrop spectrophotometer. The CD measurements were then performed by UW-Madison's Biophysics Instrumentation Facility. They measured CD spectra using a 1 mm pathlength on an AVIV Model 420 Circular Dichroism Spectrometer at 4°C. The CD spectra for Design10 was normalized to the wild-type GB1 spectra at 222 nm.

GB1 Yeast Display Plasmid Construction and Flow Cytometric IgG Binding Affinity Titration

We synthesized wild-type and Design10 variant GB1 genes as yeast codon-optimized gBlocks (Integrated DNA Technologies, Coralville, IA). The gBlocks were ligated into the unique NheI and BamHI sites of the yeast surface display vector pCTCON2 (provided by Dane Wittrup, MIT). We synthesized A24Y and E19Q+A24Y variant GB1 genes as yeast-optimized gene fragments (Twist, San Francisco, CA). The gene fragments were ligated into a golden-gate compatible version of pCTCON2 at the NheI, BamHI sites. This yeast display vector fuses the Aga2p protein to the N-terminus of GB1.

We transformed plasmid DNA into yeast display *Saccharomyces cerevisiae* strain EBY100 made competent using the Zymo Research Frozen EZ Yeast Transformation

II kit with transformants grown on synthetic dropout (SD) -Trp (MP Biomedicals, Irvine, CA) agar plates for two days at 30°C. After two days, individual colonies were picked into 4 mL of low-pH Sabouraud Dextrose Casamino Acid media (20 g/L dextrose, 6.7 g/L yeast nitrogen base, 5 g/L casamino acids, 10.4 g/L sodium citrate, 7.4 g/L citric acid monohydrate) and grown overnight at 30°C and 250 rpm. For induction of GB1 display, we started a 5 mL Sabouraud Galactose Casamino Acid (8.6 g/L $NaH_2PO^*H_2O$, 5.4 g/L Na_2HPO_4, 20 g/L galactose, 6.7 g/L yeast nitrogen base, 5 g/L casamino acids) culture at an optical density, as measured at 600 nm, of 0.5 and shook overnight at 250 rpm and 20°C.

We harvested approximately 2×10^5 yeast cells for each titration data point by centrifugation after overnight incubation, washed them once in pH 7.4 Phosphate Buffered Saline (PBS) containing 0.2% (w/v) bovine serum albumin (BSA), and incubated them overnight at 4°C on a tube rotator at 18 rpm in between 100 μL and 800 μL of PBS/0.2% BSA containing various concentrations of mouse IgG2a (BioLegend, San Diego, CA) that had been conjugated with Alexa647 using NHS chemistry (Molecular Probes, Eugene, OR). Volumes of Alexa647 IgG-containing incubation solution were varied to prevent ligand depletion from occurring in the lowest IgG concentration incubation tubes. Following overnight incubation, yeast were washed once in PBS/0.2% BSA and resuspended in ice cold PBS for flow cytometric analysis. Analyses were performed using a Fortessa analyzer (Becton Dickinson), and the mean of the fluorescence distribution was reported.

We performed duplicate fluorescence measurements for all nine IgG concentrations tested. We then fit a Hill function to the average of these duplicate measure-

ments. We were able to determine the K_d of Design10 as 5 nM because the binding curve was beginning to display saturation. We were unable to determine the K_d of wild-type, A24Y, or E19Q+A24Y GB1 variants because the proteins were less than 50% bound at the highest IgG concentration tested.

2.B Supplementary Figures

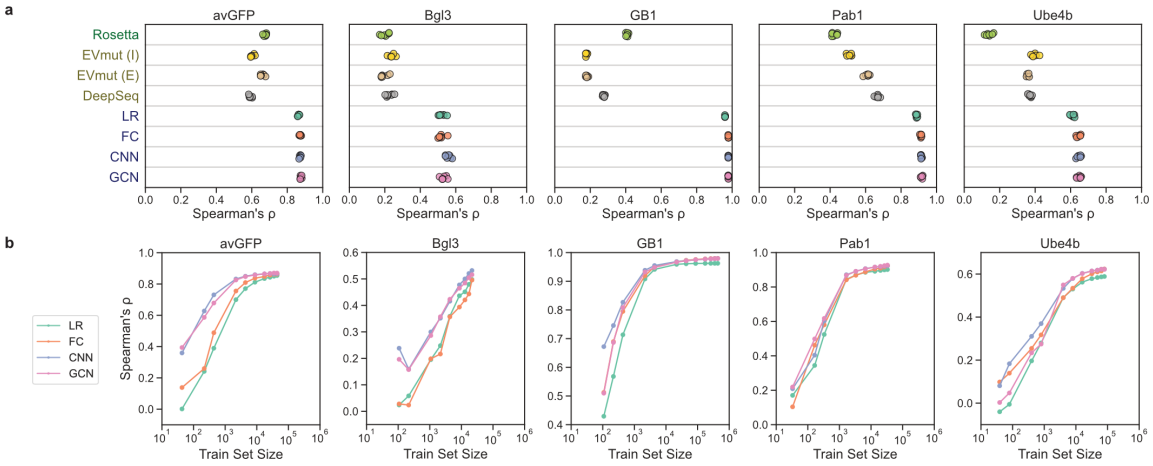

Figure A.1: **Model evaluation using Spearman's correlation coefficient.** (a) Spearman's correlation coefficient between true and predicted scores for Rosetta, EVmutation, DeepSequence, linear regression (LR), fully connected network (FC), sequence convolutional network (CNN), and graph convolutional network (GCN). EVmutation (I) refers to the independent formulation of the model that does not include pairwise interactions. EVmutation (E) refers to the epistatic formulation of the model that does include pairwise interactions. Each point corresponds to one of seven random train-tune-test splits. (b) Spearman's correlation performance of supervised models trained with reduced training set sizes.

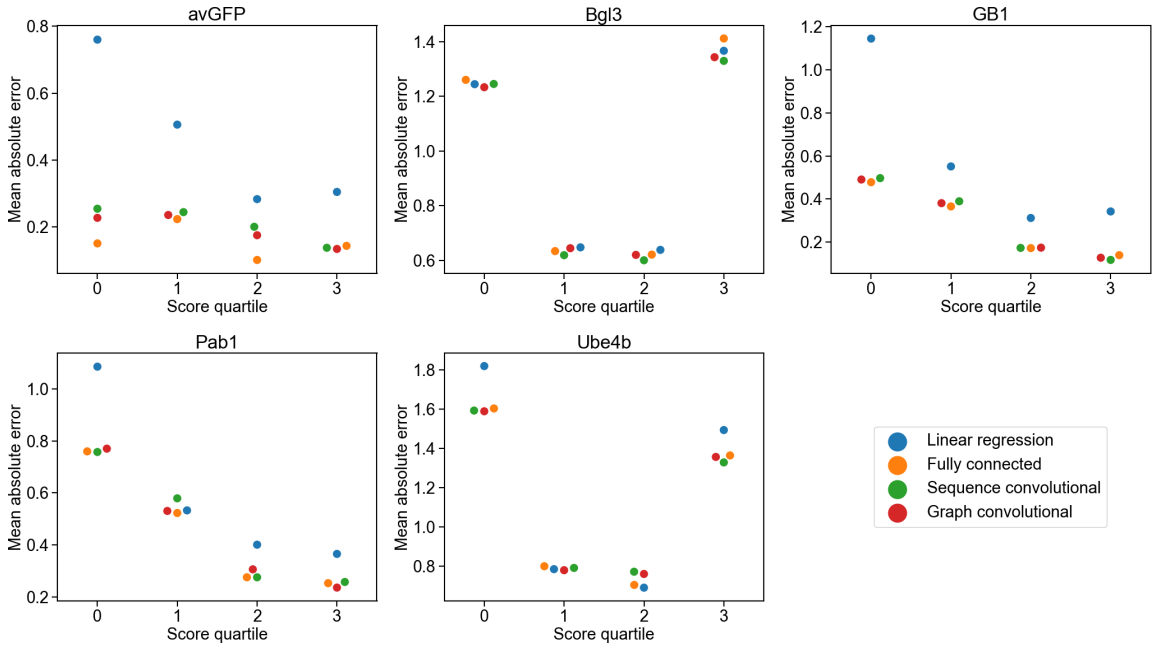

Figure A.2: **Mean absolute error vs. score quartile.** The mean absolute error in the models' predictions grouped by score quartile. Linear regression has a substantial jump in error for low-scoring variants compared to the other models in avGFP, GB1, Pab1, and Ube4b.

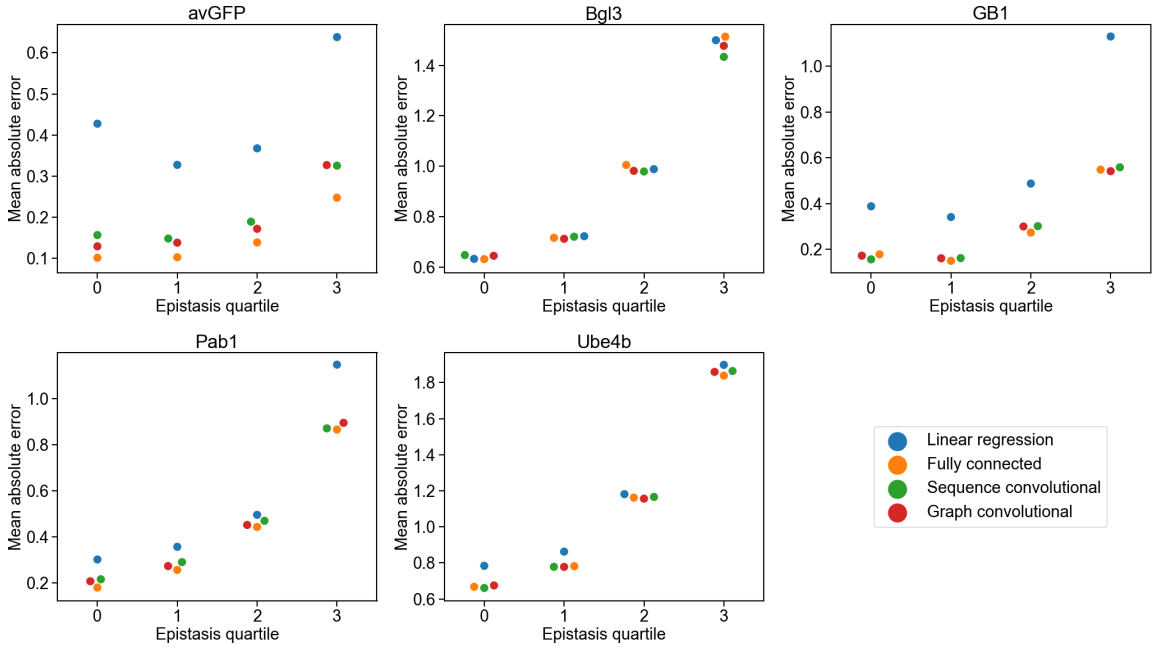

Figure A.3: **Mean absolute error vs. epistasis quartile.** The mean absolute error in the models' predictions grouped by absolute epistasis quartile. We compute epistasis by subtracting the expected score for the multi-mutant sequence from the true score. The expected score for the multi-mutant sequence is the sum of the corresponding single-mutant scores, truncated to the observed minimum or maximum in the dataset. Linear regression has a substantial jump in error for high-epistasis variants compared to the other models in avGFP, GB1, and Pab1.

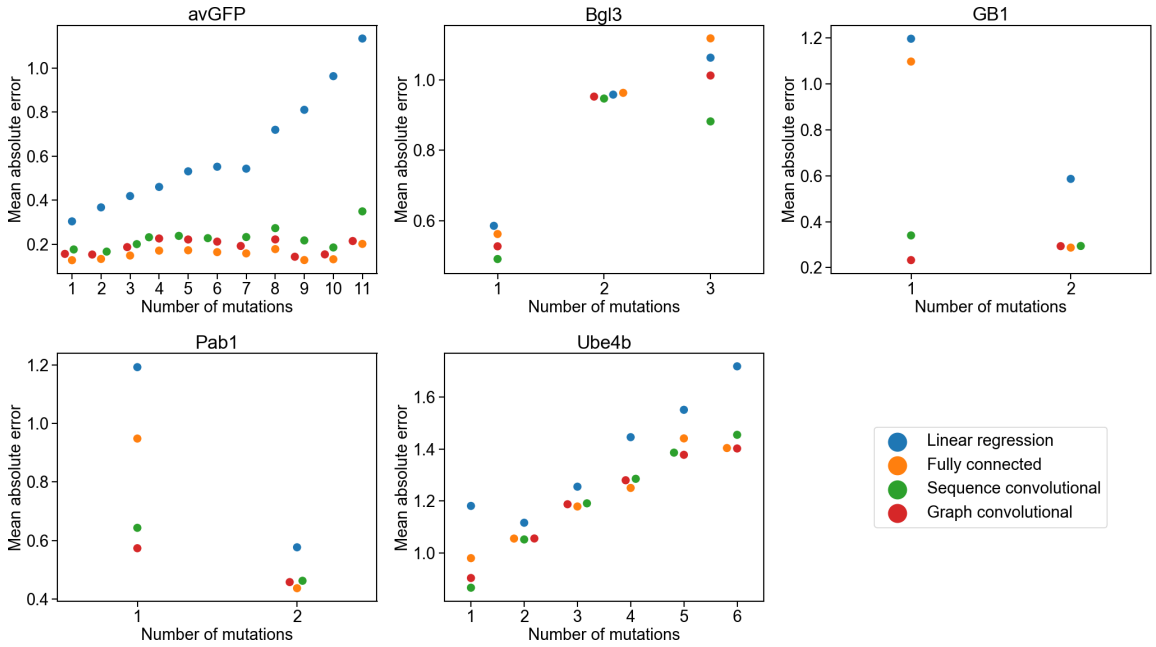

Figure A.4: **Mean absolute error vs. number of mutations.** The absolute error in the models' predictions for each variant grouped by the number of mutations in the variant. Linear regression struggles with increasing numbers of mutations in avGFP. The convolutional networks perform better than linear regression and the fully connected network on single-mutation variants in GB1 and Pab1.

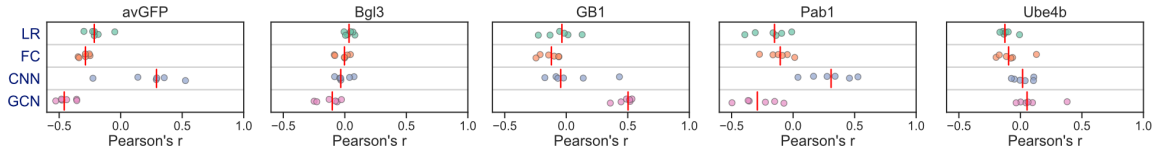

Figure A.5: **Positional extrapolation.** Model performance when making predictions for variants containing mutations in positions that were unmodified in the training data (positional extrapolation). Each point corresponds to one of six replicates, and the red vertical line denotes the median.

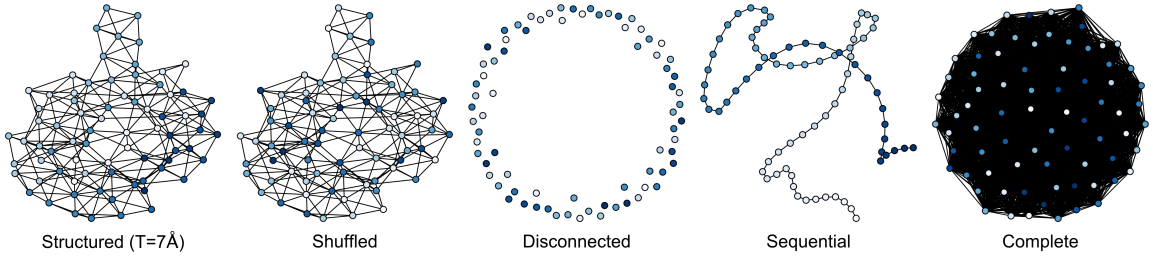

Figure A.6: **Protein structure graphs for Pab1.** The graph convolutional network uses a graph of the protein's structure to determine which residues are close together. In addition to the standard graph based on the protein's actual structure, we tested four baseline graphs: a shuffled graph based on the standard graph but with shuffled node labels, a disconnected graph with no edges, a sequential graph containing only edges between sequential residues, and a complete graph containing all possible edges. The graphs pictured are for the Pab1 dataset. The structured graph uses a distance threshold of 7Å to determine which residues should be connected with edges (selected by hyperparameter sweep). The nodes are colored according to each residue's sequence position, with light colors corresponding to residues at the start of the sequence and dark blue colors corresponding to residues at the end of the sequence.

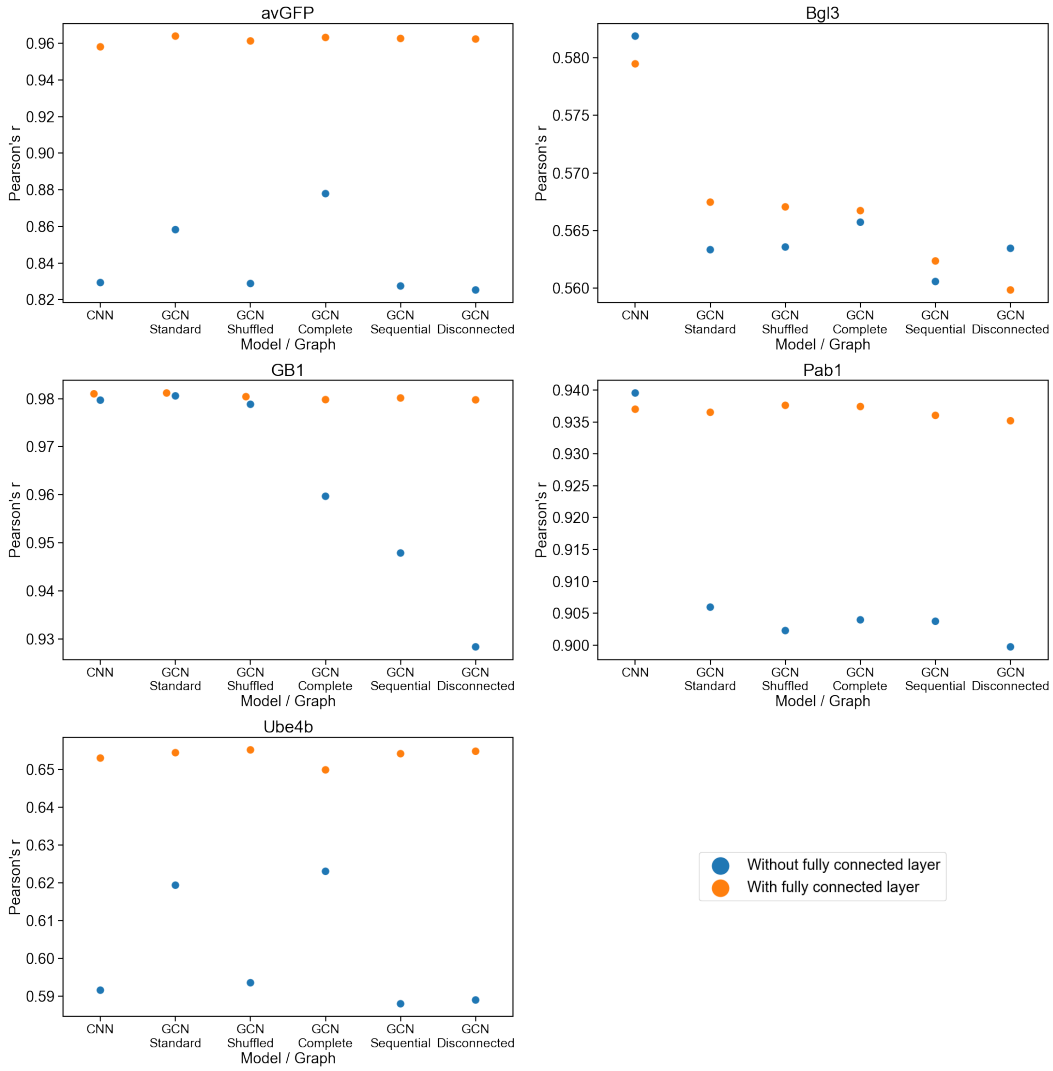

Figure A.7: **Convolutional networks with and without a fully connected layer.** The correlation performance of sequence convolutional and graph convolutional networks trained with various baseline structure graphs, with and without a final fully connected layer. The standard graph is based on the protein's actual structure. The shuffled graph is a version of the regular structured graph with shuffled node labels. The complete graph contains all possible edges between residues. The sequential graph only contains edges between sequential residues. The disconnected graph contains no edges. The fully connected layer at the end of the network compensates for apparent differences in performance caused by type of convolutional network or different graph structures.

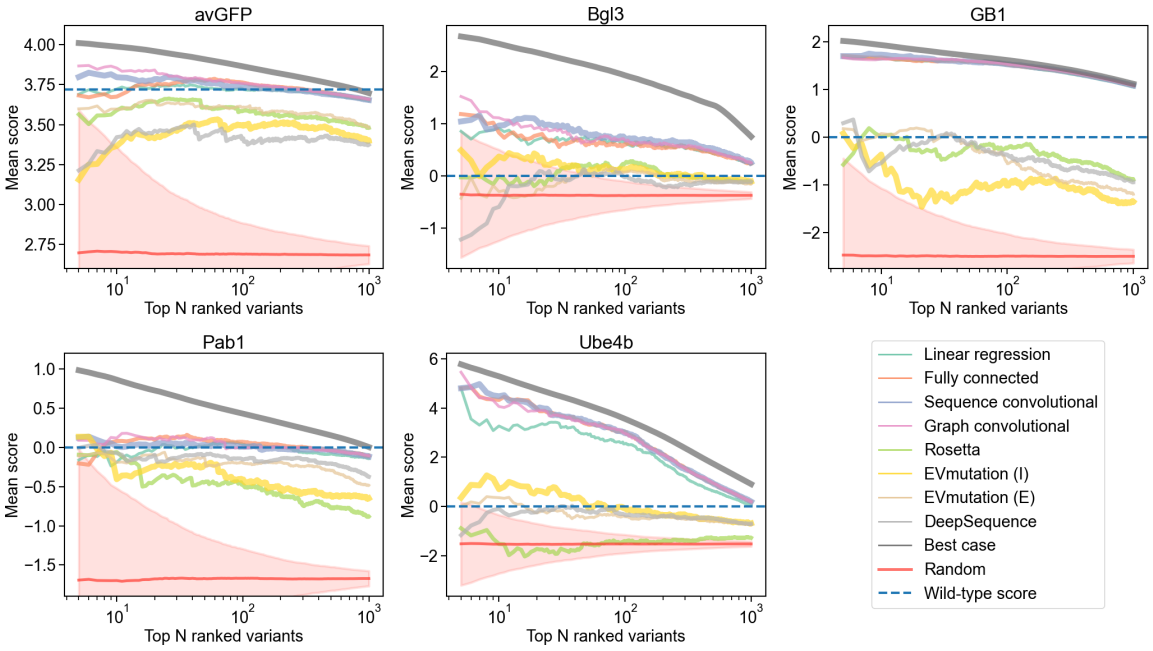

Figure A.8: **Mean score of highest ranked variants.** The mean score of each model's ranking of the highest scoring test set variants. For the most part, the supervised models prioritize variants whose average score is higher than the wild-type. The random baseline is shown with the mean and 95% confidence interval.

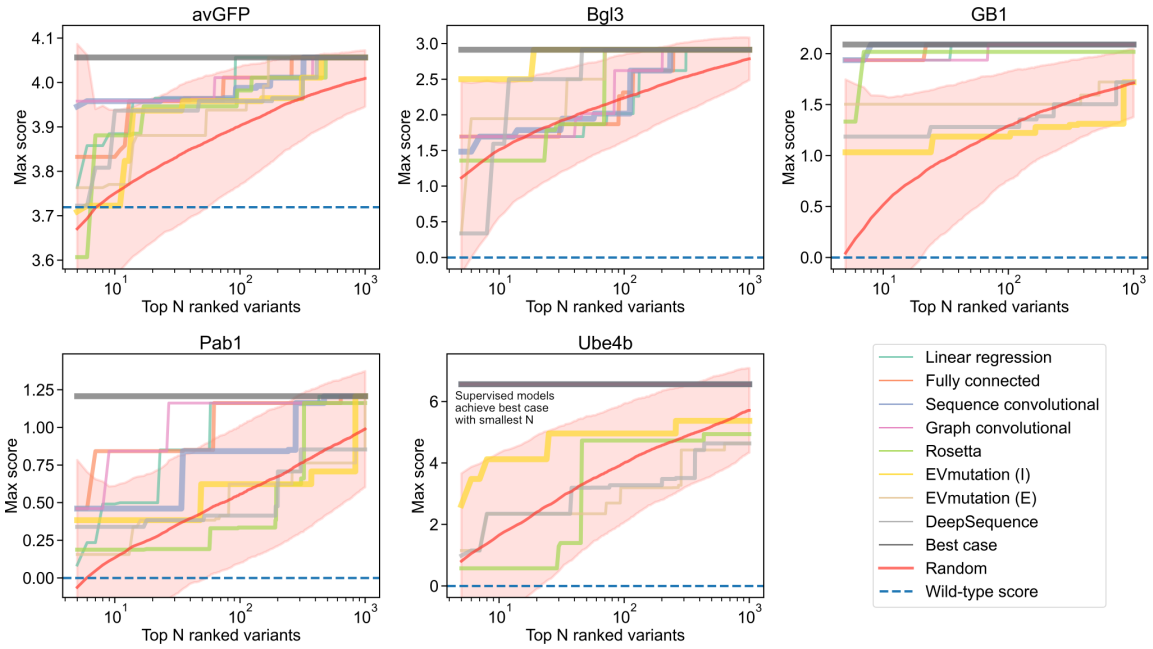

Figure A.9: **Max score of highest ranked variants.** The max score in each model's ranking of the highest scoring test set variants. For Ube4b, the supervised models prioritize a variant with the true max score with the smallest tested budget (N=5), thus all the lines corresponding to the supervised models are hidden behind the line for the true score. Nearly all models across all datasets prioritize variants whose max score is higher than the wild-type. The random baseline is shown with the mean and 95% confidence interval.

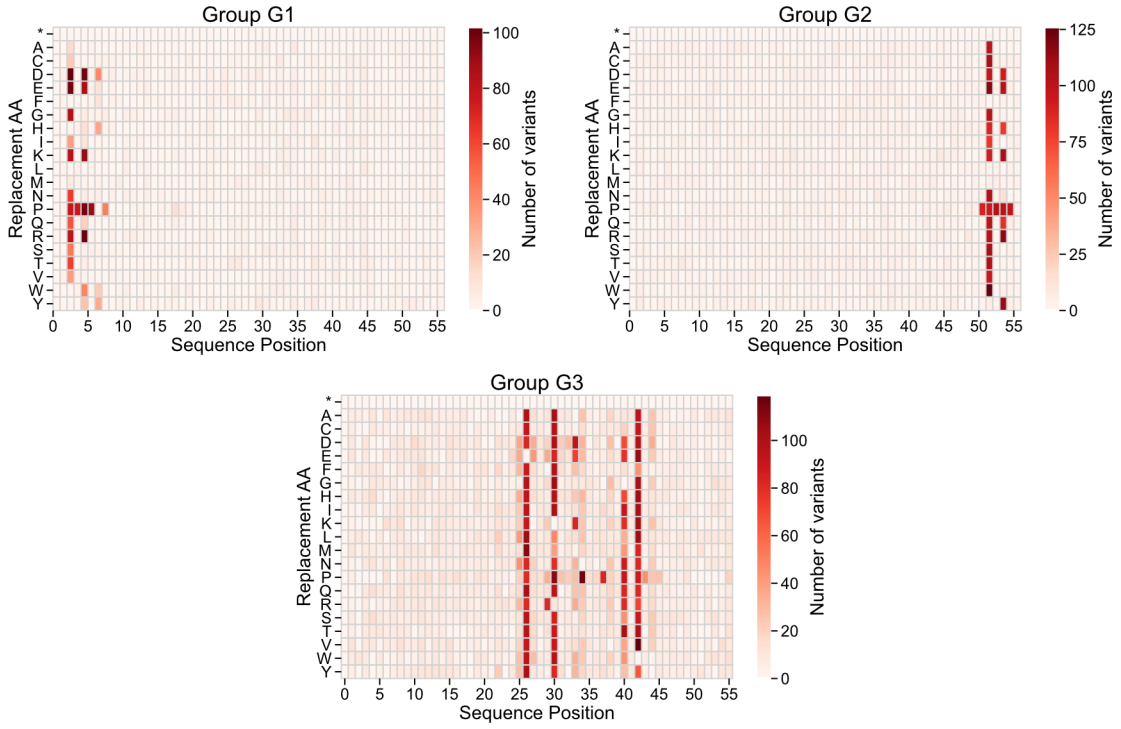

Figure A.10: **Mutations in GB1 latent space groups.** Heat maps showing the number of occurrences of mutations for each annotated group in the GB1 latent space in Figure 2.4a. Groups G1 and G2 contain variants with mutations at core residues near the start and end of the sequence, respectively. Group G3 contains variants with mutations at surface interface residues.

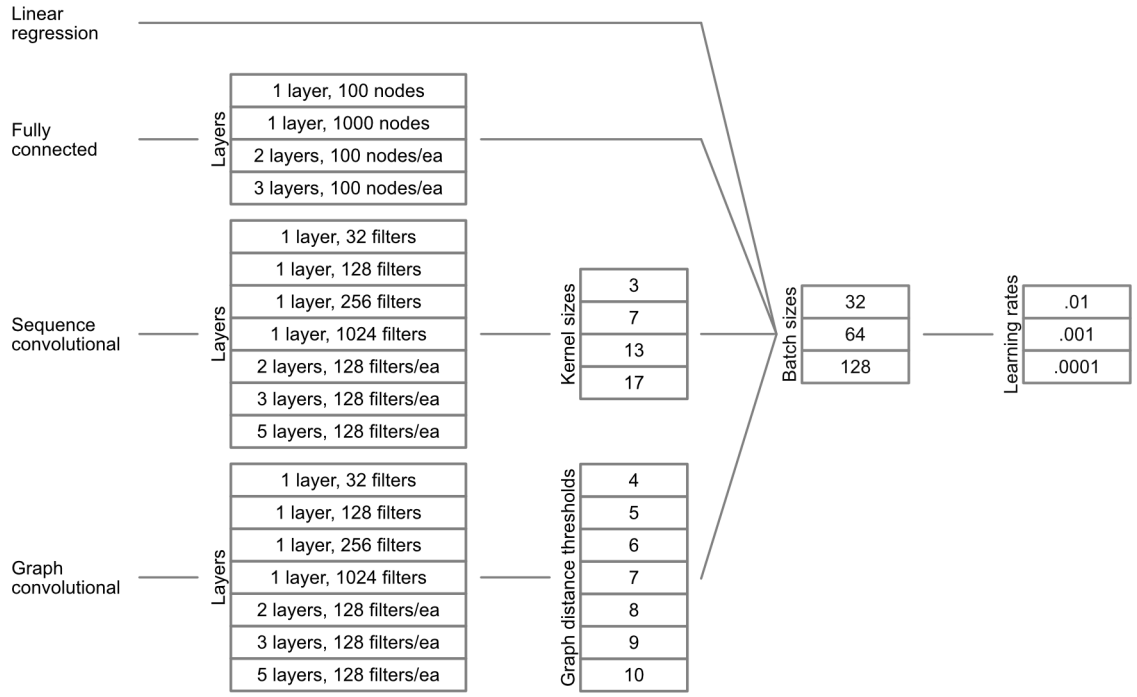

Figure A.11: **Hyperparameter sweep.** We performed an exhaustive hyperparameter sweep for each dataset and type of model using all possible combinations of these hyperparameters.

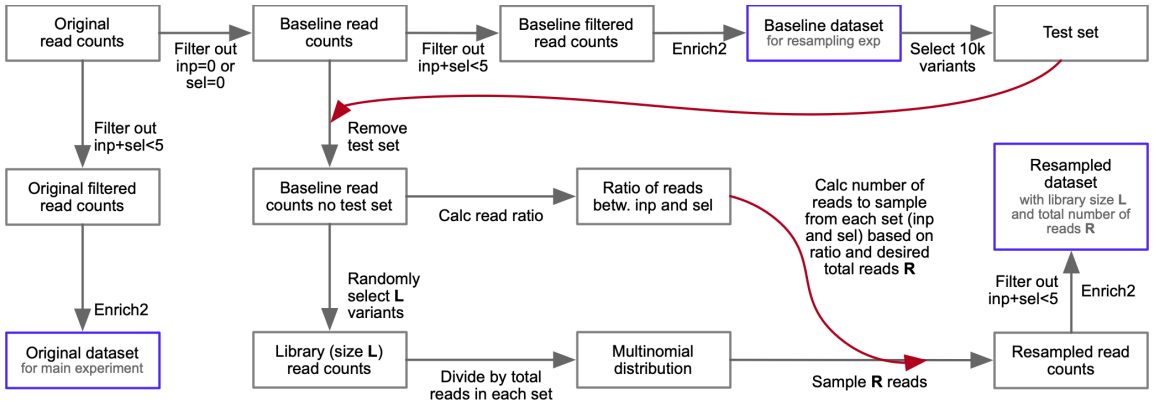

Figure A.12: **Generation of resampled GB1 datasets.** Flowchart showing how we created resampled GB1 datasets corresponding to different library sizes and numbers of reads.

Variant	Amino acid sequence
Wild-type	MQYKLILNGKTLKGETTTEAVDAATAEKVFKQYANDNGVDGEWTYDDATKTFTVTE
Design10	MQYKLILNGKTLKGETWTWGHDPYRAEKKFKLYANDNGVWGEWTYDDATKTFTVTE
Design20	MQYKLEANWKTLKGETFTIAVDDYRAEKHFKLMMNANNIYGLWTYDRATKTFGMTE
Design30	MQYKLETWHPWNAGERNRVAVVAYMAEKNFKNKLNANNWWGTWTIDWAGKTFGCTA
Design40	MAFKNEAWWPWWCEEINRVAHAAWWAEVNFKNKLNANNWFGCWADCWAHGIFGATT
Design50	MPHTCEANDWWNWEVVNWSRHAPYRAEIHHKNEAFSLNWLGTWQGIRVQDRFNFGT

Table A.1: **Designed GB1 sequences.** The GB1 wild-type sequence and the designed sequences with increasing numbers of mutations (10, 20, 30, 40, and 50) from wild-type.

Mutation	Frequency	Present in Design10
A24Y	0.93	Y
D40W	0.52	Y
D40Y	0.48	N
V29K	0.48	Y
Q32L	0.45	Y
E42Q	0.42	N
A34M	0.33	N

Table A.2: **Diversity in designed GB1 sequences.** We repeated our hill climbing protein design approach 100 times to generate 100 sequences with 10 mutations each. We found 27 of 100 design runs converged to the same sequence. The other 73 represent distinct local optima in the landscape. A number of mutations were observed across multiple designs, and some of these were present in Design10. This table lists mutations common across the designs and their frequencies.

Dataset	Model type	Key parts of architecture	Learning rate	Batch size	Epochs
avGFP	Linear regression	Linear regression	0.0001	128	90
	Fully connected	3 layers, 100 hidden units each	0.0001	32	134
	Sequence convolutional	5 layers, kernel size 3, 128 filters	0.001	64	113
	Graph convolutional	2 layers, 7Å threshold, 128 filters	0.0001	32	130
Bgl3	Linear regression	Linear regression	0.0001	128	164
	Fully connected	2 layers, 100 hidden units each	0.0001	32	187
	Sequence convolutional	1 layer, kernel size 17, 32 filters	0.0001	64	102
	Graph convolutional	1 layer, 6Å threshold, 32 filters	0.0001	128	129
GB1	Linear regression	Linear regression	0.0001	128	27
	Fully connected	1 layer, 1000 hidden units	0.0001	64	110
	Sequence convolutional	3 layers, kernel size 17, 128 filters	0.0001	32	27
	Graph convolutional	5 layers, 7Å threshold, 128 filters	0.0001	32	109
Pab1	Linear regression	Linear regression	0.001	128	47
	Fully connected	3 layers, 100 hidden units each	0.001	128	108
	Sequence convolutional	3 layers, kernel size 17, 128 filters	0.0001	128	42
	Graph convolutional	1 layer, 7Å threshold, 32 filters	0.0001	128	232
Ube4b	Linear regression	Linear regression	0.0001	64	73
	Fully connected	3 layers, 100 hidden units each	0.0001	64	124
	Sequence convolutional	5 layers, kernel size 3, 128 filters	0.0001	128	29
	Graph convolutional	3 layers, 7Å threshold, 128 filters	0.0001	128	92

Table A.3: **Selected hyperparameters.** The hyperparameters selected by a hyperparameter sweep for the main experiment. There are additional parts of the architecture that were not part of the hyperparameter sweep. For example, the fully connected networks have a dropout layer after every dense layer. The convolutional networks have a dense layer and a dropout layer before the output node. Experiments with reduced training set sizes and GB1 resampling used the same architectures selected for the main experiment, but they had their own sweeps for learning rate and batch size.

Model	avGFP	Bgl3	GB1	Pab1	Ube4b
Linear regression	9,481	20,041	2,241	3,001	4,081
Fully connected	968,401	2,014,301	2,242,001	320,401	428,401
Sequence convolutional	3,118,409	1,573,993	747,081	990,281	1,390,409
Graph convolutional	3,077,065	1,605,993	858,953	242,793	1,381,961

Table A.4: **Numbers of trainable parameters.** The number of trainable parameters in each model.

Library	Version
python	3.6.8
cudatoolkit	10.0.130
cudnn	7.6.0
tensorflow-gpu	1.14.0
gast	0.2.2
numpy	1.16.4
joblib	0.13.2
matplotlib	3.1.1
networkx	2.3
pandas	0.25.0
scikit-learn	0.21.2
scipy	1.3.0
seaborn	0.9.0
enrich2	1.2.1

Table A.5: **Main software packages.** The main libraries and version numbers used to train and evaluate models.

The work presented in this chapter was performed in collaboration with Bryce Johnson, Sameer D'Costa, Chase Freschlin, Philip A. Romero, and Anthony Gitter.

3.1 Introduction

Just as words combine to form sentences that convey meaning in human languages, the specific arrangement of amino acids in proteins can be viewed as an information rich language describing molecular structure and behavior. Protein language models (PLMs) harness advances in natural language processing to decode intricate patterns and relationships within protein sequences and have broad utility in protein engineering. These models learn meaningful representations that capture the semantic organization of protein space and can also be used in generative settings to create custom-made proteins with desired characteristics. PLMs' learned representations are low dimensional and can be related to specific protein properties such as enzyme activity or stability from limited training examples.

Protein language models such as UniRep and Evolutionary Scale Modeling (ESM) are trained on vast repositories of natural protein sequences distributed across the evolutionary tree. The training process often involves self-supervised autoregressive next token prediction or masked token prediction, and through this process, the PLMs learn context-aware representations of the amino acids within a protein. Training on examples of natural proteins produces PLMs that implicitly capture protein structure, biological function, and other evolutionary pressures.

While these models are powerful, they do not take advantage of our extensive knowledge of protein biophysics and molecular mechanisms acquired over the last century, and as a result are largely unaware of the underlying physical principles governing protein function.

In this work we introduce Mutational Effect Transfer Learning (METL), a pretraining strategy that integrates biophysical knowledge into protein language models. We use molecular modeling to generate large-scale synthetic data across diverse protein sequences and folds and pretrain a transformer-based PLM that encapsulates this biophysical knowledge. This biophysical prior is then refined through a fine-tuning process on experimental sequence-function data to produce a biophysics-aware model that can predict specific protein properties. We develop METL models in a global setting that are pretrained across protein folds and a local setting that are focused on a particular protein. We find biophysical pretraining produces models that better generalize to new regions of sequence space and can predict the effects of mutations and epistatic interactions not observed in the training data. Strong generalization allows the models to learn from limited sequence-function information. We demonstrate the ability of METL models to design functional GFP variants when trained on only 64 examples. METL provides a general framework for incorporating biophysical knowledge into protein language models and with the potential to become increasingly powerful with more advanced molecular modeling and simulation methods. The fusion of molecular mechanistic and deep learning models holds great potential for unraveling the intricacies of protein behavior and advancing drug design, disease understanding,

and synthetic biology.

3.2 Results

Mutational Effect Transfer Learning Overview

We introduce METL, a framework for training neural networks to predict experimentally derived functional scores for protein sequence variants, even when the available experimental training data is limited (Fig. 3.1). METL incorporates molecular simulations as a means to augment experimental datasets, operating in three main steps. First, we generate pretraining data via molecular simulations, employing Rosetta (Alford et al., 2017) to simulate protein variants and compute Rosetta score terms. The Rosetta scores include physical energies and statistical potentials and capture the stability and likelihood of protein conformations. Second, we pretrain a transformer encoder (Vaswani et al., 2017) to predict the Rosetta scores, allowing the model to learn relationships between amino acid substitutions and sequence positions and to form an internal representation of protein sequences based on the Rosetta scores. Finally, we fine-tune the pretrained transformer encoder to predict the functional scores from experimental sequence-function datasets.

Within the METL framework, we implement two pretraining strategies: METL-Local and METL-Global, which differ in what sequences are included in the pretraining data. The local strategy generates a protein representation targeted to a specific protein. For this strategy, we randomly generate up to 20M sequence variants of the target protein with a maximum of 5 amino acid substitutions per variant.

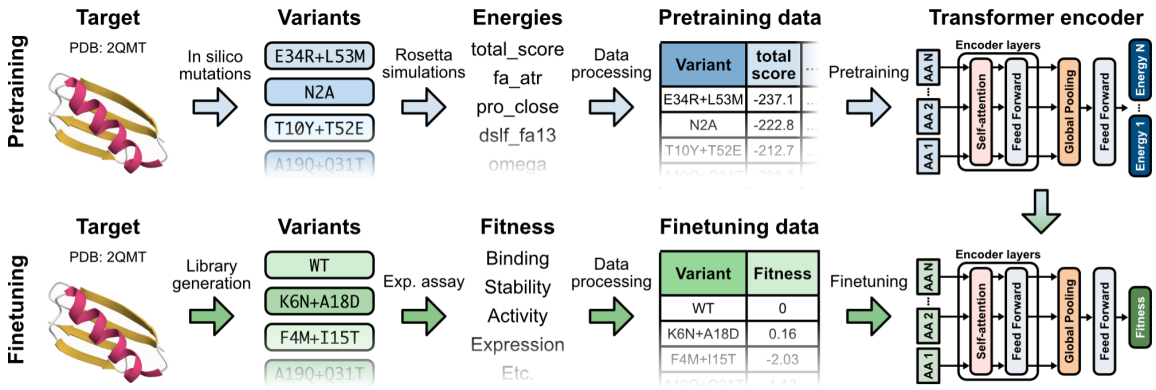

Figure 3.1: **Overview of Mutational Effect Transfer Learning (METL) with the local pretraining strategy.** In the pretraining phase (upper row), we start with a specific target protein (shown here as PDB:2QMT), randomly generate millions of sequence variants with up to 5 amino acid substitutions, and compute Rosetta score terms for each sequence variant. Then, we use the resulting data to pretrain a transformer encoder to predict the Rosetta scores. In the finetuning phase (lower row), we use experimental sequence-function data to fine-tune the pretrained neural network from the previous phase. The experimental sequence-function data may come from high-throughput experiments like deep mutational scanning or low throughput biological assays. It consists of variants of the same target protein and an associated functional score for each variant telling us how functional the variant is for the specific functional property measured in the experimental assay. The experimental functional score could measure properties such as binding, thermostability, and expression.

This provides coverage of the local sequence space surrounding the protein of interest. In contrast, the global strategy aims to create a general protein representation applicable to any target protein. For the global strategy, we select 148 diverse base proteins (Kosciolek and Jones, 2014) and score 200K sequence variants per base protein to generate a pretraining dataset consisting of approximately 30M variants. The global strategy provides greater diversity in the training data, which supports the learning of a more broadly applicable representation. For both strategies, we compute 55 Rosetta scores for each sequence variant, forming sequence-score pairs that serve as the pretraining data.

We utilize a transformer encoder architecture with a relative positional embedding (Shaw et al., 2018) based on three-dimensional protein structure. The relative position embedding enables the transformer to consider positional representations of the inputs in terms of three-dimensional distances between residues. The local models consist of 2M parameters, with 3 encoder layers and a 256 dimension embedding. The global models consist of 20M parameters, with 6 encoder layers and a 512 dimension embedding. We begin by pretraining the transformer encoder with the Rosetta data to predict the 55 Rosetta scores given input amino acid sequences. Then, we fine-tune the network with experimental data to transition it from predicting Rosetta scores to predicting the functional scores obtained from the experimental assay. We implement a dual-phase fine-tuning strategy (Kumar et al., 2022), first training a new prediction head while keeping other network weights frozen, followed by training the entire network at a reduced learning rate.

The primary purpose of pretraining is not to recapitulate Rosetta scores but rather to learn an information-rich protein representation that can serve as a starting point for finetuning on experimental data. Nonetheless, the pretrained models are capable of predicting Rosetta scores with remarkable speed and accuracy (Fig. B.1). METL-Local demonstrates strong predictive ability for Rosetta's *total score* energy term, achieving a mean Spearman correlation between true and predicted score of 0.96 across the 7 METL-Local source models we trained. With METL-Global, we observed a substantial difference in predictive ability for in-distribution PDBs (those included in the METL-Global pretraining data, Spearman correlation 0.85) and out-of-distribution PDBs (those not included, Spearman correlation 0.17). This

suggests METL-Global is overfitting to the PDBs present in the pretraining data. However, METL-Global still captures biologically relevant amino acid embeddings (Fig. B.2), which should be generally informative for any given protein sequence.

Evaluation With Challenging Generalization Tasks

Generalizing to new data can be challenging for neural networks trained with limited or biased datasets. This issue is relevant in protein engineering, where experimental datasets may have few training examples or skewed mutation distributions. These factors can impact the accuracy and utility of learned models when using them to prioritize variants for experimental characterization. We evaluated the predictive generalization performance of METL on eight experimental datasets, representing proteins of various sizes and functions: avGFP, DLG4-2022, GB1, GRB2-Abundance, GRB2-Binding, Pab1, TEM-1, and Ube4b (Table 3.1). To thoroughly evaluate performance with limited training data, we implemented comprehensive train, validation, and test splits, encompassing small training set sizes and difficult extrapolation tasks. We tested multiple replicate splits to account for variation in the selection of training examples. We measure performance via the test set Spearman correlation between true and predicted scores and present the median correlation across replicates.

For comparison, we also evaluated the predictive performance of several baselines and established methods, including linear regression with a one hot sequence encoding (Linear-OH), Rosetta's *total score* as a standalone prediction, ESM-2 (Lin et al., 2023), and EVE (Frazer et al., 2021). ESM-2 is a general protein language

	Description	Organism	Molecular Function	Selection	Length	Variants	Ref.
avGFP	Green fluorescent protein	A. victoria	Fluorescence	Brightness	237	51714	Sarkisyan et al. (2016)
DLG4-2022	Postsynaptic density protein 95 PDZ3 domain	H. sapiens	Synaptic organization	CRIPT binding	84	8251	Faure et al. (2022)
GB1	Protein G B1 domain	Streptococcus sp.	Antibody binding	IgG-Fc binding	56	536084	Olson et al. (2014)
GRB2-A	Growth factor receptor-bound protein 2 SH3 domain	H. sapiens	Signaling adaptor	Abundance	56	63366	Faure et al. (2022)
GRB2-B	Growth factor receptor-bound protein 2 SH3 domain	H. sapiens	Signaling adaptor	GAB2 binding	56	33441	Faure et al. (2022)
Pab1	Pab1 RNA recognition motif (RRM) domain	S. cerevisiae	Poly(A) binding	mRNA binding	75	37710	Melamed et al. (2013)
TEM-1	TEM-1 β-lactamase	E. coli	Antibiotic hydrolysis	Ampicillin resistance	286	12374	Gonzalez and Ostermeier (2019)
Ube4b	Ubiquitination factor E4B U-box domain	M. musculus	Ubiquitin activation	Ubiquitin ligase activity	102	88375	Starita et al. (2013)

Table 3.1: **Experimental datasets.** We evaluated METL on experimental datasets representing proteins of varying sizes, folds, and functions.

model that captures a rich representation of protein sequences based on underlying evolutionary signals. We used the 35M parameter version of ESM-2 and finetuned it with the same approach we used for METL. Like ESM-2, EVE captures underlying evolutionary signals, but EVE differs from ESM-2 in that it is trained on multiple sequence alignments that are specific to the target protein family. In addition to evaluating the unsupervised EVE score as a standalone prediction, we also tested it as an input feature to linear regression in combination with one hot encoded sequences (Linear-EVE) (Hsu et al., 2022).

Generalizing from small training sets

METL-Local and Linear-EVE consistently and substantially outperformed the other supervised methods for small training set sizes across most of the tested datasets (Fig. 3.2). METL-Local outperformed Linear-EVE on the avGFP and GB1 datasets, and Linear-EVE outperformed METL-Local on the other tested datasets. The differences between these two methods were sometimes minor, such as for the GRB2-Abundance dataset. METL-Global and ESM-2 were competitive with each other for small to mid-size training sets, with ESM-2 typically surpassing METL-Global for larger training set sizes. Interestingly, METL-Global and ESM-2 are both

general protein representation models, and they performed worse than METL-Local and Linear-EVE, which are protein family-specific models. Linear-OH performed worse than the other tested methods for small to mid-size training sets. However, given enough data, all of the supervised methods, including Linear-OH, converged to similar performance levels. The maximum performance varies by dataset, with Ube4b standing out as having substantially worse maximum performance than the other datasets.

The performance of METL-Local and Linear-EVE on small training set sizes is closely related to the performance of unsupervised Rosetta *total score* and EVE, respectively. With a training set size of 8 (the smallest we tested), Linear-EVE performs approximately the same as EVE. This suggests that the 8 experimental training examples are not contributing much over the standalone EVE score when used with the Linear-EVE regression framework. Conversely, METL-Local sometimes over-performs or under-performs the standalone Rosetta *total score* prediction, which may be due to the METL-Local finetuning framework or the fact that METL trains on numerous Rosetta score terms and not solely *total score*. In some cases, like for the avGFP dataset, it can take hundreds of experimental training examples for Linear-EVE to outperform unsupervised EVE. On the other hand, METL-Local can outperform Rosetta *total score* with as few as 8 training examples, and it consistently outperforms Rosetta *total score* with 16 examples, with the exception of the GRB2-Abundance dataset.

METL-Local performs especially well on the avGFP and TEM-1 datasets, which can be explained in part by the strong correlation between Rosetta's *total score* and

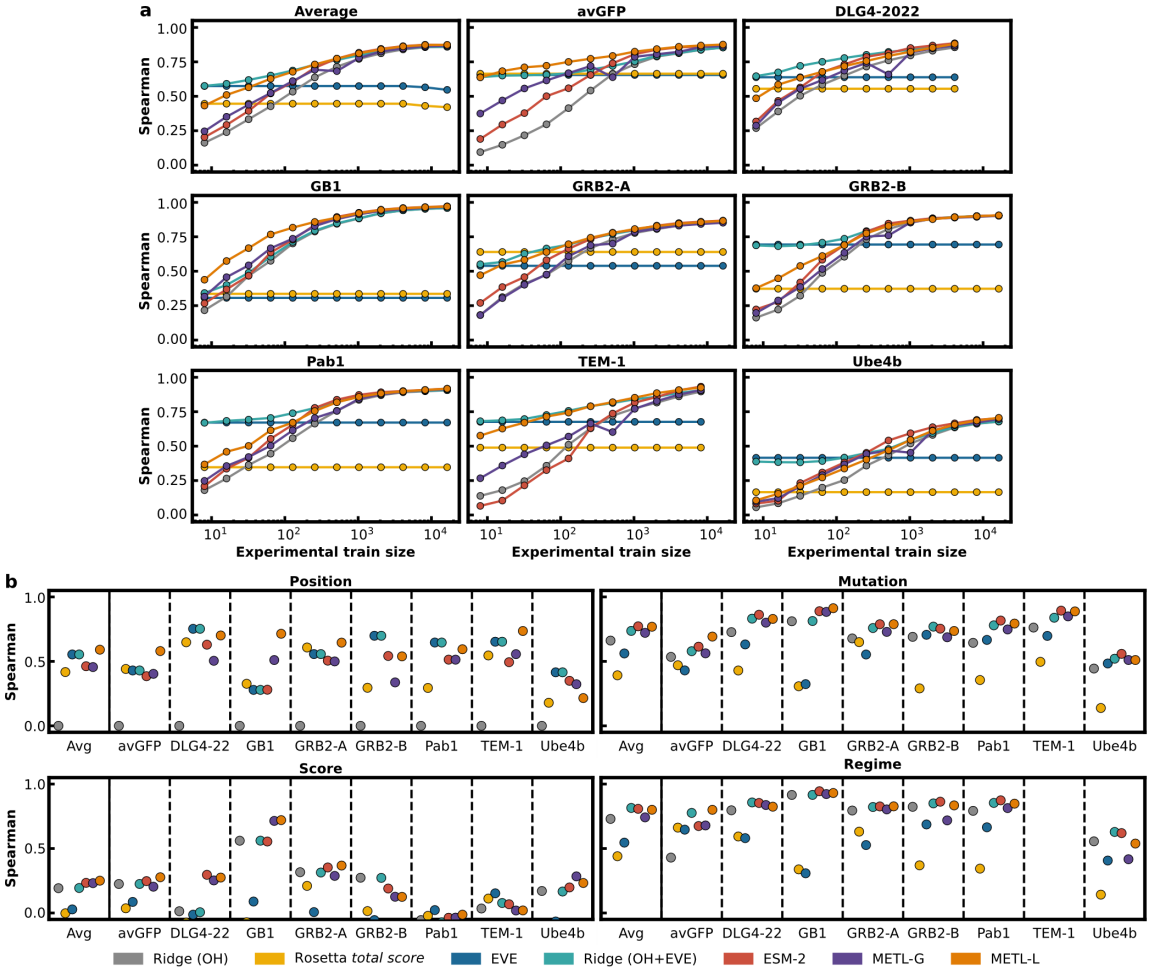

Figure 3.2: **Comparative performance of Linear-OH, Rosetta *total score*, EVE, Linear-EVE, ESM-2, METL-Global, and METL-Local across different training set sizes and extrapolation tasks.** (a) Learning curves for eight datasets showing the test set Spearman correlation between true and predicted scores across a number of training set sizes ranging from 8 to 16,384 examples. We tested multiple replicates for each training set size, starting with 101 replicates for the smallest train set size and decreasing to 3 replicates for the largest size. We show the median Spearman correlation across these replicates. The top left panel ("Average") shows the mean of the learning curves across all the pictured datasets. (b) Correlation performance of each method on position, mutation, score, and regime extrapolation. We tested 9 replicates for each type of extrapolation and show the median.

the experimental functional score for these datasets. In general, for small training set sizes, we observed that the stronger the correlation between Rosetta's *total score* and the experimental functional score, the stronger the METL-Local performance (Fig. B.3). However, once hundreds or thousands of training examples are available, METL-Local performance is dominated by dataset-specific effects rather than the correspondence between Rosetta *total score* and protein function. Furthermore, we found that at small training set sizes, the selection of experimental training examples can make a substantial difference in predictive performance (Fig. B.4), with the variance due to the selection of training set examples decreasing as the training set size increases (Fig. B.5).

Position extrapolation

We implemented a number of extrapolation tasks — position, mutation, score, and regime extrapolation — to simulate challenging scenarios that may arise in real protein engineering applications, such as datasets missing examples of mutations in certain positions, having biased score distributions with predominantly low-scoring variants, or consisting of solely single-substitution variants. Position extrapolation tests a model's ability to extrapolate to sequence positions that are not represented in the training data. This is a challenging task requiring the model to possess substantial prior knowledge or a structural understanding of the protein (Mater et al., 2020). Notably, METL-Local and the EVE-based methods demonstrated superior position extrapolation performance across the tested datasets, achieving average Spearman correlations of 0.59 and 0.55, respectively. METL-Global and

ESM-2 both achieved a lower average Spearman correlation of 0.46. METL-Local's advantage can be attributed, in part, to the local pretraining data, which includes mutations in all sequence positions of the target sequence, providing the model with prior knowledge of each sequence position. Linear-OH is unable to perform position extrapolation because the model cannot learn weights for sequence positions not represented in the training data. This logic also explains why EVE and Linear-EVE perform identically for position extrapolation. The one hot sequence features in Linear-EVE are not contributing to position extrapolation, so the model is relying solely on the EVE score.

Mutation extrapolation

Mutation extrapolation evaluates a model's ability to extrapolate to specific amino acid substitutions that are not present in the training data. Mutation extrapolation is relatively easier than position extrapolation because the model has access to examples of mutations in every sequence position, which provide information about the importance of those sequence positions. METL-Local and ESM-2 performed similarly in mutation extrapolation, achieving an average Spearman correlation across datasets of 0.77. Linear-EVE and METL-Global achieved average Spearman correlations of 0.74 and 0.72. Linear-OH performed the worst among the tested supervised methods with an average correlation of 0.66. The unsupervised Rosetta *total score* and EVE models performed about the same for mutation extrapolation as they did for small train sizes and position extrapolation. These models do not train on the experimental data, so their performance completely depends on the

test set. The test set used in mutation extrapolation has a similar distribution to the test sets used for small train sizes and position extrapolation.

Score extrapolation

Extrapolating from variants with lower-than-wild-type scores to variants with higher-than-wild-type scores proves to be a challenging task (Dallago et al., 2022). For most datasets, the maximum Spearman correlation for score extrapolation was less than 0.3, and for certain datasets like Pab1, all the Spearman correlations fell below 0.1. The GB1 dataset is an exception, where both METL-Local and METL-Global achieved relatively high Spearman correlations of 0.72 and 0.71. For comparison, Linear-EVE, Linear-OH, and ESM-2 achieved Spearman correlations of 0.56, 0.56, and 0.55 on the GB1 dataset. The difficulty of score extrapolation might be attributed to the information content of low-scoring variants. Many low-scoring variants have minimal function and exhibit instability, resulting in their low scores. Although the scoring method assigns a range of scores to these variants, there may be little meaningful distinction between their scores in reality. It is possible the information learned from these low scoring variants is not useful for ranking high scoring variants, thus resulting in the low score extrapolation for most datasets.

Moreover, the unsupervised methods Rosetta *total score* and EVE, which do not train on the experimental data, performed worse on score extrapolation than the other generalization tasks we evaluated. This is because the test set in score extrapolation is highly biased, containing only variants that function better than wild-type. One reason as to why the unsupervised methods would perform worse when eval-

uating only on high-scoring variants is that at least some of the performance of these unsupervised methods in the other generalization tasks is due to their ability to distinguish between low-scoring variants and high-scoring variants. This reasoning could also help explain the weak performance of the supervised methods, in addition to the information content of low-scoring variants. Given the focus of protein engineering on identifying variants with high functional scores, these results highlight the importance of having training examples with higher-than-wild-type scores.

Regime extrapolation

Experimental datasets often contain only single amino acid substitution variants, lacking examples of higher-order mutations and non-additive (epistatic) interactions. Regime extrapolation tests a model's ability to extrapolate to variants with 2 or more amino acid substitutions when trained on variants containing only 1 amino acid substitution. All of the supervised models achieved relatively high performance across the datasets, with the exception of the Ube4b dataset. Interestingly, Linear-OH, which inherently captures only additive interactions, still achieved strong Spearman correlations on several datasets, suggesting many of the interactions in these datasets are additive. The unsupervised Rosetta *total score* and EVE performed worse than the supervised models in most cases. The avGFP dataset stands out as it exhibits a larger distinction between the models than the other datasets, with METL-Local, Linear-EVE, METL-Global, ESM-2, and Linear-OH achieving Spearman correlations of 0.8, 0.78, 0.68, 0.67, and 0.43, respectively.

Furthermore, the avGFP and Ube4b datasets contain variants with up to 15 and 10 amino acid substitutions, respectively. This enables us to test a second type of regime extrapolation, where we train the model on variants with 1 or 2 amino acid substitutions and evaluate on variants with 3 or more substitutions (Fig. B.6). Notably, on the avGFP dataset, METL-Local's and Linear-EVE's performance remained relatively consistent regardless of their training on singles or both singles and doubles. Conversely, the other models exhibited substantial improvements when trained on singles and doubles. This suggests METL-Local and Linear-EVE may be capturing knowledge about interactions, even when trained with just singles, whereas the other models do not capture the same information unless trained with singles and doubles. The Ube4b dataset shows a different pattern, with all models improving when trained with singles and doubles.

Additional baselines

We evaluated several additional baselines, including METL-Local with random initialization, linear regression with Rosetta's *total score* as an input feature (Linear-RTS), and sequence convolutional networks. METL-Local with random initialization performed substantially worse than its pretrained counterpart (Fig. B.7), showing the considerable impact of pretraining. Incorporating Rosetta's *total score* as an input feature for linear regression, in combination with one hot encoding, greatly improved performance over solely using one hot encoding features (Fig. B.8). This baseline is similar in concept to Linear-EVE, but it uses Rosetta's *total score* instead of the EVE score as an additional feature. While Linear-RTS sometimes

matched METL-Local's performance and even exceeded it on the GRB2-A dataset, METL-Local still outperformed it on average. Furthermore, it should be noted that Linear-RTS requires running Rosetta to compute the *total score* for every variant, even during inference. We further tested sequence convolutional networks and fully connected networks (Fig. B.9), and while sequence convolutional networks offered some improvement over Linear-OH in extrapolation tasks, they did not approach METL-Local in challenging small train set size or extrapolation scenarios.

Comparative analysis of relative embeddings, feature extraction, and model size

We examined several components of the METL architecture to investigate their effects on model performance, including one-dimensional (sequence-based) versus three-dimensional (structure-based) relative position embeddings (Fig B.10), feature extraction versus finetuning (Fig B.11), and global models with 20M versus 50M parameters (Fig B.13). While METL-Local did not benefit much from three-dimensional embeddings over one-dimensional embeddings (except for the TEM-1 dataset), METL-Global showed consistent improvement with three-dimensional embeddings. This suggests that spatial information from three-dimensional structures provides useful context when diverse structures are involved, such as in the METL-Global pretraining data. In particular, METL-Global's position extrapolation performance benefited substantially from incorporating three-dimensional relative position embeddings, suggesting 3D structure information plays an important role in predicting behavior of unseen sequence positions.

We compared the efficacy of fine-tuning to feature extraction for both METL and ESM-2 models (Fig. B.11). To perform feature extraction, we saved outputs from the appropriate internal layer of each model and then used those features as inputs to train linear regression. Fine-tuning consistently outperformed feature extraction for both METL-Local and METL-Global. In contrast, for ESM-2, there were several instances where feature extraction substantially outperformed fine-tuning when applied to small training set sizes, namely for the DLG4-2022, GRB2-B, Pab1, and TEM-1 datasets. Notably, for DLG4-2022 and Pab1 with small training set sizes, the performance of ESM-2 feature extraction exceeded that of METL-Local finetuning. However, this is only the case for small training set sizes, and ESM-2 finetuning outperforms ESM-2 feature extraction given enough experimental training data, even for these datasets.

In addition to the 35M parameter ESM-2 model, we also tested feature extraction with the 150M and 650M parameter ESM-2 models (Fig. B.12). The 150M parameter model consistently outperformed the 35M parameter model, with the exception of the DLG4-2022 dataset, where the 35M parameter model actually performed better. Surprisingly, the 650M parameter model performed worse than both the 35M and 150M parameter models for small training set sizes with the avGFP, DLG4-2022, and Pab1 datasets. In other cases, such as for GB1, GRB2-A, and GRB2-B, the 650M parameter model offered some improvement over the 35M and 150M parameter models, at least for larger training set sizes.

Beyond the 20M parameter METL-Global models used for our main results, we also tested METL-Global models with 50M parameters (Fig. B.13). The performance

of the 50M parameter model was similar on average to the 20M parameter version, with the 50M parameter model offering minor improvements for some datasets like GRB2-B but also performing slightly worse for other datasets like Ube4b. Examining the pretrained source models, we observed that the 50M parameter METL-Global overfits more than the 20M parameter METL-Global when predicting Rosetta's *total score* on in-distribution PDBs and generalizes worse to out-of-distribution PDBs (Fig. B.14). This overfitting suggests that simply increasing the number of parameters may not necessarily yield a better representation or more generalizable representation without making other changing to the training strategy.

Relationship Between Experimental and Simulated Data

Experimental assays allow for the measurement of specific functions, but they can be complicated, expensive, and time-consuming. Conversely, molecular simulations are simpler to execute, but they do not necessarily model the exact phenotype of interest. Moreover, running millions of simulations can also take considerable time and computational resources. To quantify the tradeoff between experimental and simulated data, we measured the performance of METL-Local, pretrained and finetuned with varying amounts of simulated and experimental data, respectively, for the GB1 dataset (Fig. 3.3).

As expected, increasing the amount experimental data improves performance. Furthermore, increasing the amount of simulated data also improves performance, and simulated data can partially compensate for a lack of experimental data. For instance, a model pretrained with 1K simulated examples achieves a Spearman's

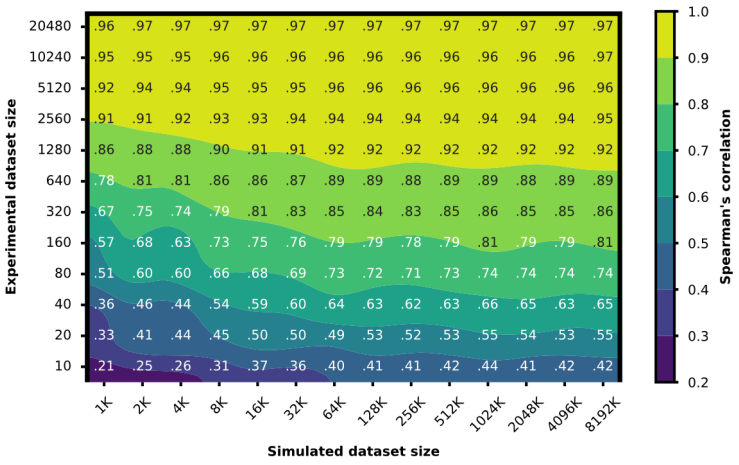

Figure 3.3: **Relationship between experimental and simulated data for the GB1 dataset.** The contour plot illustrates the test set Spearman's correlation resulting from training METL-Local with varying amounts of simulated (pretraining) and experimental (finetuning) data. The plot displays a grid of Spearman's correlation values on the same test dataset corresponding to discrete combinations of experimental and simulated dataset sizes. The model benefits from larger quantities of experimental and simulated data, with the latter producing diminishing returns after approximately 128K examples.

correlation of 0.36 when fine-tuned with 40 experimental examples and 0.51 when fine-tuned with 80 experimental examples. However, by increasing the amount of pretraining examples to 128K, the model achieves a Spearman's correlation of 0.53 with just 20 experimental examples. There are, however, diminishing returns for adding additional simulated data beyond a certain threshold. The performance improvements with additional pretraining data are marginal after about 128K examples, at least for this relatively small 2M parameter model.

We ran our simulations on servers across the OSG Open Science Pool (OSG, 2006), with each compute job requiring only 1 CPU core and 2GB of memory. The average runtime for a single GB1 variant was approximately 59 seconds. Given this

average runtime, simulating 128K GB1 variants would take around 87 compute-days. If distributed across 96 cores, such as on a 96-core server processor, running all 128K GB1 simulations would take under one day. Simulations for larger proteins took longer, with an average of approximately 152 seconds for an avGFP variant. On a faster M2 Max MacBook Pro, the average runtime is approximately 19 seconds for a GB1 variant and approximately 50 seconds for an avGFP variant. These results suggest it is possible to achieve sufficient pretraining with a limited number of simulations, within a reasonable computational and time budget.

Customized Biophysical Simulations Improve METL-Local Performance

The METL framework supports pretraining on function-specific molecular simulations, such as binding-specific scores, that more closely align with the target biological functions or experimental assays. Similarity between pretraining and target tasks is important to achieve strong performance and avoid detrimental effects in transfer learning (Wang et al., 2019). Indeed, for small experimental training set sizes, we observed that the stronger the correlation between Rosetta's *total score* and the experimental functional score, the stronger the METL-Local performance (Fig. B.3).

To demonstrate how function-specific simulations can improve the relevance of the pretrained METL model and its performance after finetuning, we conducted function-specific simulations for the GB1 dataset. The function measured by this assay is the binding affinity between the GB1 domain of streptococcal protein G and

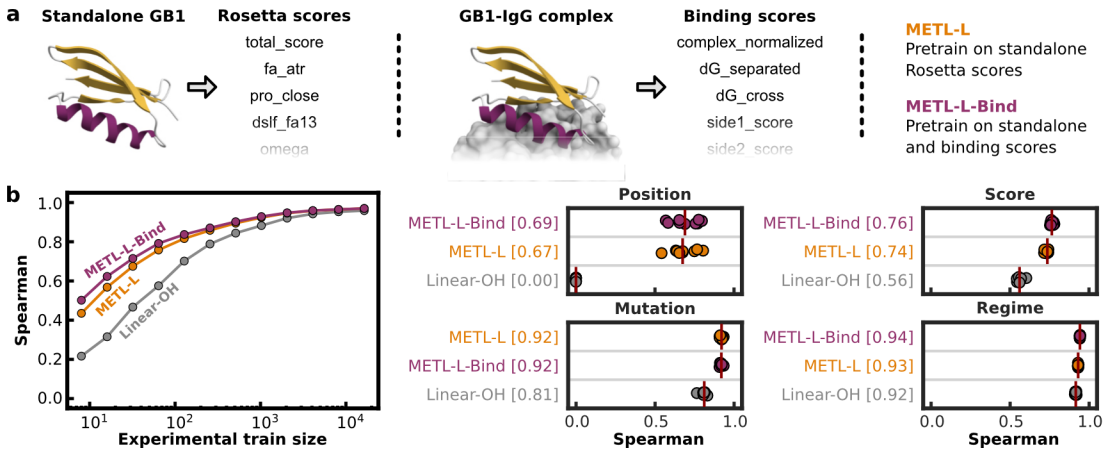

Figure 3.4: **Customized binding simulations improve METL-Local performance for GB1 dataset.** (a) METL-Local pretrains on general stability-related Rosetta scores from the standalone GB1 structure. METL-Local-Bind pretrains on both general Rosetta scores from the standalone GB1 structure and binding-specific scores from the GB1-IgG complex structure. (b) Learning curves and extrapolation performance for Linear-OH, METL-L, and METL-L-Bind on the GB1 dataset. We pretrained METL-L and METL-L-Bind on identical datasets, differing only in the target Rosetta score terms. We used the same finetuning dataset splits and replicates as the results in Figure 3.2. Vertical red bar denotes the median of the extrapolation replicates.

mammalian IgG. To match this experimentally assayed function, we implemented an enhanced Rosetta pipeline to compute binding-specific scores based on the GB1-IgG protein complex. The Rosetta binding scores include new energy terms such as *per_residue_energy_int*, which captures the average energy of each residue at the interface, and *dG_separated*, which captures the change in energy when the chains are separated versus bound (Table B.5). The GB1 experimental score correlates more strongly with several of the binding-specific Rosetta scores than it does with the strongest-correlated Rosetta score from the standalone GB1 structure (Fig. B.15). This suggests the simulated binding-specific scores are more aligned with the underlying binding signal from the experimental assay.

We pretrained a standard METL-Local (METL-L) model and a modified METL-L-Bind model using the standard 55 Rosetta scores and those scores plus the binding scores, respectively (Fig. 3.4). The models were identical besides the tasks in the pretraining dataset, enabling us to isolate the effects of pretraining on the additional binding scores. We evaluated the performance of METL-L and METL-L-Bind using the same small training size and extrapolation tasks used in our main experiments. METL-L-Bind outperformed the standard METL-L when finetuned on small amounts of data. On the smallest training set sizes of 8, 16, and 32, METL-L-Bind achieved test set Spearman correlations of 0.50, 0.62, and 0.71, while METL-L performed worse with Spearman correlations of 0.43, 0.57, and 0.67. METL-L-Bind's improvement over METL-L on the position, score, and regime extrapolation tasks was negligible. These results suggest that protein function-specific simulations can be worthwhile when used within the METL pretraining framework.

Low-N avGFP Design

In protein engineering, computational models that predict protein variant fitness can be used to prioritize high-functioning variants for experimental characterization. However, these models often face the challenge of making predictions based on limited training data or extrapolating to unexplored regions of sequence space. To demonstrate METL's potential for real protein engineering applications, we tested METL-Local's ability to prioritize fluorescent avGFP variants in these challenging design scenarios. We used METL-Local to design 20 avGFP sequences that were not part of the original DMS dataset, and we experimentally validated the resulting

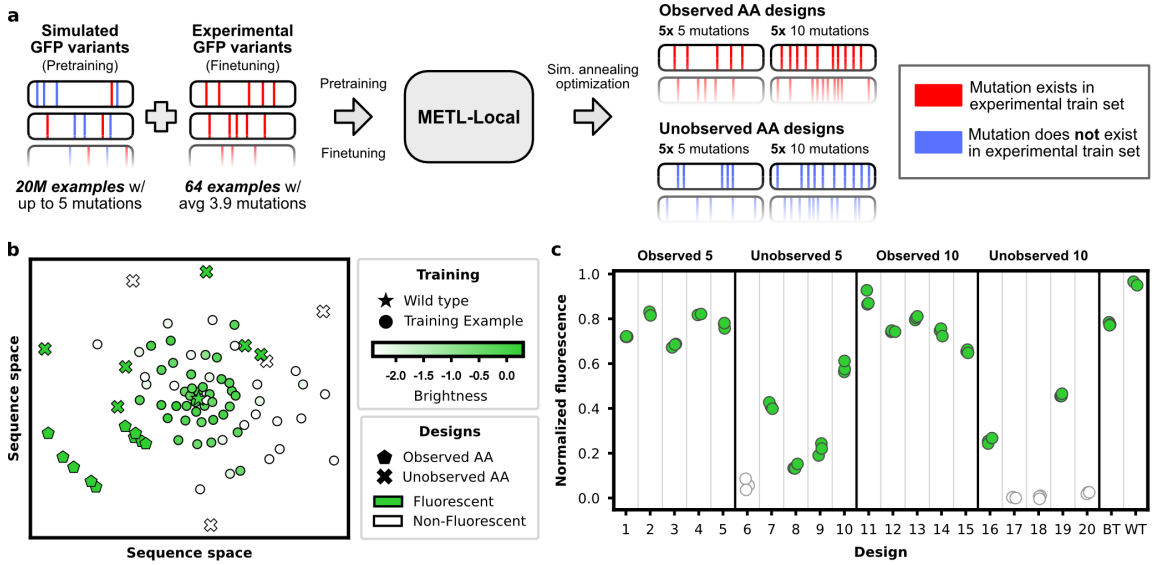

Figure 3.5: **Low-N avGFP Design.** (a) Overview of GFP design experiment. We tested 2 different design constraints: *Observed AA*, where sequences contain only amino acid substitutions found in the training set, and *Unobserved AA*, where sequences exclude any amino acid substitutions found in the training set. (b) Multidimensional scaling (MDS) sequence space visualization of the wild-type avGFP sequence, the 64 avGFP training sequences, and the 20 designed proteins. The designed sequences contain either 5 or 10 amino acid substitutions from wild-type. Training set sequences are colored on a gradient according to their experimental brightness score. Designed sequences are colored according to whether they exhibited fluorescence. (c) Experimentally characterized fluorescence brightness (multiple replicates) of the designed sequences, the best training set sequence (BT), and the wild-type sequence (WT).

avGFP variants to measure their fluorescence brightness (Fig. 3.5 and Table B.1).

We incorporated several constraints to create a challenging protein engineering scenario. We emulated low-N protein engineering by using only $N = 64$ randomly-sampled variants from the full DMS dataset as the METL-Local training set. The brightness distribution of the sampled variants roughly matched that of the full DMS dataset, and the average number of amino acid substitutions per variant in the sample was 3.9. We designed variants with either 5 or 10 amino acid substitutions

away from wild-type, forcing the model to perform regime extrapolation. Furthermore, we tested two different constraints on the sequence design process: *Observed AA* and *Unobserved AA*. In the Observed design scenario, the designed sequences can contain only amino acid substitutions found in the training set. Conversely, in the Unobserved scenario, the designed sequences must exclude any amino acid substitutions found in the training set. The Unobserved constraint is similar to mutation extrapolation in that the model must make predictions for amino acid substitutions that are not present in the training data. We designed 20 avGFP sequences in total: 10 Observed and 10 Unobserved with 5 sequences with 5 amino acid substitutions and 5 sequences with 10 substitutions in each scenario. We used simulated annealing to search over avGFP sequence space for designs that maximize the fitness predicted by METL-Local and clustered the simulated annealing solutions to select diverse sequences.

To identify variants with improved fluorescence due to increased brightness rather than improved expression, we expressed avGFP as a fusion protein with mKate2, emulating the conditions used to collect the training dataset (Sarkisyan et al., 2016). Briefly, avGFP is expressed as a fusion protein with mKate2 using a rigid alpha-helical linker. This minimizes FRET activity and limits the potential for highly unstable or misfolded avGFP variants from significantly affecting the folding of mKate2. We calculated a brightness value for each avGFP by normalizing its fluorescence with respect to mKate2.

Of the 20 designed avGFP sequences, 16 exhibited fluorescence (Fig. 3.5c). We achieved a 40% (2/5) hit rate in the most restrictive design scenario, Unob-

served with 10 amino acid substitutions. The percentage increased to 80% (4/5) when considering sequences with 5 amino acid substitutions. In the Observed design scenario, our hit rate was 100% (10/10). Although none of the designed sequences expressed higher brightness than wild type, several matched or exceeded the brightness of the best training set variant. Six variants exhibited greater avGFP fluorescence than wild-type avGFP, and 12 variants exhibited greater mKate2 fluorescence than that measured in the wild-type avGFP construct. Because mKate2 is constant across all variants, the differences in mKate2 fluorescence may be driven by avGFP stability. In the case of variants that express in the Unobserved scenario, we theorize improved stability compensates for reduced brightness such that the total avGFP fluorescence is greater than wild-type avGFP.

3.3 Discussion

Motivated by decades of research into biophysics, molecular dynamics, and protein simulation (Hollingsworth and Dror, 2018; Alford et al., 2017), we present METL, which leverages transfer learning from molecular simulations to improve protein variant fitness prediction with small or biased datasets. Unlike existing protein language models or multiple sequence alignment-based methods that train on natural sequences (Alley et al., 2019; Rives et al., 2021; Bepler and Berger, 2021; Frazer et al., 2021; Elnaggar et al., 2022; Yang et al., 2023), METL captures different underlying signals, which are generated through molecular simulations rather than natural selection and evolution. We implemented METL with pretraining

on general Rosetta score terms, which broadly capture aspects of protein stability. The METL framework also supports pretraining on function-specific molecular simulations, such as binding-specific scores, that can be tailored to specific biological functions and assays. METL's pretraining strategy incorporates both functional and non-functional examples from regions of sequence space close to the protein of interest, providing valuable context for predicting functional effects of amino acid substitutions.

Our comprehensive evaluation of METL's predictive generalization across various experimental datasets revealed that METL-Local substantially outperformed other methods for certain datasets such as avGFP and GB1. Due to biophysics-based knowledge embedded within the model and insights into how mutations relate to each other in the pretraining data, METL-Local excelled in tasks like generalizing from small training set sizes and position extrapolation. However, Linear-EVE outperformed METL-Local for many of the tested datasets. The performance differences between METL-Local and Linear-EVE can be explained, at least in part, by their reliance on protein stability and evolutionary information, respectively. Our in-depth analysis uncovered other notable patterns, such as that the protein family-specific methods, METL-Local and Linear-EVE, outperformed the general protein representation methods, METL-Global and ESM-2. METL-Global performed similarly to ESM-2 35M when finetuned on experimental data. However, we found METL-Global was overfitting to PDBs in the pretraining data, suggesting there is more work to be done in improving the pretraining strategy for the global model.

Our results provide insights into the performance of METL, EVE, and other

related methods, but they also raise broader questions about the merits of evolutionary versus biophysics-based data in predicting protein variant fitness. Evolutionary data captures protein fitness information through natural selection and evolution, encompassing protein stability and functions under evolutionary pressures, such as binding. In contrast, biophysics-based data predominantly captures protein stability. Evolutionary information often correlates better with experimental datasets than protein stability information alone (Høie et al., 2022). Consistent with these findings, we observed that for many of our tested datasets and challenging generalization tasks, EVE (evolutionary-based) outperformed Rosetta *total score* (stability-based). By extension, Linear-EVE outperformed METL-Local. Regardless, certain datasets lent themselves better to modeling with Rosetta and METL, such as GB1 and avGFP. The question of whether to rely on evolutionary or biophysics-based information depends on how closely the experimental function of interest relates to these different signals. Additionally, there are confounding factors, such as some target proteins not having very deep alignments, which provide less signal for evolutionary-based methods, or certain proteins being harder to simulate, which makes the biophysics-based signal noisier. Finally, protein stability and evolutionary data may provide complementary information to protein fitness prediction, and future work can focus on how to best combine these sources of information.

Prior models have integrated biophysics and machine learning either by using biophysics-based features as input to machine learning models or approximating biophysics simulations with machine learning. Rosetta and FoldX stability and energy terms have been provided as features for an augmented linear regres-

sion model (Hsu et al., 2022), a 2D CNN (Harmalkar et al., 2023), and on nodes and edges in a graph neural network (Wang et al., 2022a). Function-value-prior augmented-Bayesian Neural Networks can incorporate Rosetta stability as a prior on protein function prediction in regions where a Bayesian Neural Network has high epistemic uncertainty (Nisonoff et al., 2022). Nordquist et al. (Nordquist et al., 2023) include both Rosetta- and molecular dynamics-derived features in their supervised learning models of big potassium channels. Unlike a fine-tuned METL-Local model, all of these approaches must run the biophysics calculations for each sequence variant, which could limit their scalability in protein engineering applications. Alternative protein design and engineering strategies use machine learning to approximate the biophysics calculations, similar to METL's pretraining. These include the Epistasis Neural Network that has been used to engineer xylanases (Lipsh-Sokolik et al., 2023) and GFP variants (Weinstein et al., 2023), molecular dynamics approximations to minimize energy and match a target structure (Omar et al., 2023), and learning to predict Rosetta protein-ligand binding energy to speed up variant scoring (Ramírez-Palacios and Marrink, 2023). Predicting biophysics scores for protein engineering is related to the long-standing problem of predicting stability (Capriotti et al., 2005; Folkman et al., 2016; Cao et al., 2019; Chen et al., 2020; Li et al., 2020; Wang et al., 2022b; Blaabjerg et al., 2023; Hummer et al., 2023; Zhou et al., 2023; Dieckhaus et al., 2023; Boyer et al., 2023; Sun et al., 2023).

Examples across diverse scientific domains have demonstrated the power of combining simulations and machine learning (Cranmer et al., 2020), spanning topics such as gene regulatory network reconstruction (Wu and Sinha, 2023), chemical

foundation model pretraining (Ahmad et al., 2022), climate emulation (Yu et al., 2023), and quantum chemistry approximation (Eastman et al., 2023a,b). METL fits within this broader trend and represents a significant step toward effectively integrating biophysics insights with machine learning-based protein fitness prediction. The METL framework pretrains protein language models on molecular simulations, capturing underlying signals present in the simulated data. METL can pretrain on general stability terms or more specific function-related scores, offering the potential to model protein functions that can be simulated but are not highly evolutionarily constrained. As the field of biophysics and molecular simulation continues to evolve, METL stands to benefit from faster and more accurate simulations. Biophysics-based pretraining can help overcome key challenges in protein engineering, such prioritizing protein variants for experimental analysis with limited training data. Consequently, METL emerges as a promising tool for protein engineering with a distinct approach from the many existing methods rooted in evolutionary information.

3.4 Methods

Generating Rosetta pretraining data

The Rosetta pretraining data consists of protein sequences and their corresponding score terms, computed by modeling the sequences with Rosetta. The data used for local and global source models differs in what sequences are included. Rosetta data for local source models contains protein variants within the local sequence

space surrounding the protein of interest. Rosetta data for global source models contains protein variants from a diverse range of base sequences and structures.

We generated local Rosetta datasets for each target protein from the experimental datasets. We acquired the necessary structures for these target proteins from RCSB Protein Data Bank (Berman et al., 2000) and AlphaFold Protein Structure Database (Varadi et al., 2022). For cases where the acquired structure did not match the reference sequence of the target protein, we used Rosetta comparative modeling or truncated the acquired structure to match the reference sequence. For each local dataset, we generated ≈20M protein sequence variants with a maximum of 5 amino acid substitutions. See Table B.3 for additional details regarding local Rosetta dataset structures and variants, including exceptions to the above.

We generated the global Rosetta dataset based on 150 diverse protein structures identified in Supplementary Table 1 of Kosciolek and Jones (Kosciolek and Jones, 2014). We downloaded the 150 structures from RCSB Protein Data Bank (Berman et al., 2000). Some structures contained modified or missing residues. We replaced modified residues with canonical amino acids and used the RosettaRemodel application to fill in the structure of missing residues. We were unable to remodel PDB IDs 1J3A and 1JBE, thus we excluded these structures from the final dataset. For each of the remaining 148 structures, we generated ≈200K variants with a maximum of 5 amino acid substitutions, for a total of ≈30M variants.

We used a custom sub-variants sampling algorithm to generate the variants for both the local and global datasets. The algorithm iteratively samples a random variant with 5 amino acid substitutions from the wild-type sequence then generates

all possible 4-, 3-, 2- and 1-substitution sub-variants with the same amino acid substitutions as the 5-substitution variant. Duplicate variants generated through this process are discarded. The iteration terminates when the target number of variants is reached. For the global dataset, we used the sub-variants sampling algorithm to generate all of the ≈200K variants per base sequence. For the local datasets, we first generated all possible 1-substitution or 1- and 2-substitution variants, and then we used the sub-variants sampling algorithm to generate the remainder of the ≈20M variants per target protein (Table B.3).

Once variants were generated, we used Rosetta to compute energy terms for each variant sequence. We first prepared each base PDB file for use with Rosetta by following the recommendation in the Rosetta documentation. We ran Rosetta's *clean_pdb.py* and relaxed the structure with all-heavy-atom constraints. We generated 10 structures and selected the lowest energy structure to serve as the base structure for subsequent steps.

We used Rosetta v3.13 (Alford et al., 2017) to compute full-atom energy terms (ref2015 score function), centroid-atom energy terms (score3 score function), and custom filter-based terms. For each variant, we introduced the variant's mutations to the corresponding base structure using a Rosetta resfile. Then, to generate the full-atom energy terms, we used FastRelax to relax the mutated structure using the ref2015 score function, only repacking residues within 10Å of the mutated residues, with 1 repeat. To generate the centroid-atom energy terms, we used score_jd2 to score the resulting structure using the score3 score function. Finally, we calculated custom filter-based energy terms. See Table B.4 for a list and description of each

term.

Preprocessing Rosetta pretraining data

Prior to training neural networks, we preprocessed the raw Rosetta data by dropping variants with NaN values for any of the energies, removing duplicates by randomly selecting one of the duplicates to keep, and filtering out variants with outlier $\texttt{total\_score}$ values. We grouped variants by base PDB and removed outliers independently for each group using a modified z-scores method, which uses the median and median absolute deviation instead of the mean and standard deviation. For each data point i, we calculated the modified z-score using the following equation:

$$s_i = \frac{|x_i - \tilde{x}|}{\text{MAD}}, \tag{3.1}$$

where s_i is the modified z-score, x_i is the Rosetta $\texttt{total\_score}$, $\tilde{x}$ is the median $\texttt{total\_score}$ of the group, and MAD is the Median Absolute Deviation, defined as $\text{MAD} = \text{median}(|x_j - \tilde{x}|)\ \forall\ x_j \in \{x\}$, or the median of the absolute deviations of all data points from the median of the group. We removed variants with a modified z-score > 6.5 from the dataset.

Additionally, we standardized the Rosetta scores to equalize the contribution of each score term to the model's loss function and to ensure score terms are comparable across different base PDBs in the global dataset. Once again, we grouped variants by base PDB, and then we standardized each group and score term independently by subtracting the mean and dividing by the standard deviation. We

calculated the mean and standard deviation using only the training set data. This process scales the score terms to have zero mean and a standard deviation of one.

We excluded the following score terms from the final dataset because the values were zero for a large portion of base PDBs: dslf_fa13 (from ref2015 score function), linear_chainbreak and overlap_chainbreak (from score3 score function), and filter_total_score (custom filter term). We also discarded res_count_all (custom filter term that counts the residues in the protein) because it did not vary among variants of a single base PDB. After these removals, the total number of remaining score terms was 55 (Table B.4).

METL source model architecture

The source model architecture accepts amino acid sequences as input and outputs predictions for each of the 55 Rosetta score terms. The main component of the source model architecture is a transformer encoder based on the original transformer architecture (Vaswani et al., 2017), with the notable differences being the use of a relative positional embedding (Shaw et al., 2018) instead of a sinusoidal positional encoding and pre-layer normalization instead of post-layer normalization (Xiong et al., 2020). Local source models total ≈2.5M parameters and have transformer encoders consisting of a 256 embedding size, 3 encoder layers, 4 attention heads, a 1024 feed forward hidden size, and 0.1 dropout. Global source models total ≈20M parameters and have transformer encoders consisting of a 512 embedding size, 6 encoder layers, 8 attention heads, a 2048 feed forward hidden size, and 0.1 dropout. We also evaluated a global source model with ≈50M parameters,

consisting of a similar architecture as the 20M parameter global source model but with 16 encoder layers instead of 6 encoder layers. After the transformer encoder, source models implement an additional layer normalization layer, a global average pooling layer, a nonlinear fully-connected layer, and a linear output layer with 55 output nodes corresponding to the 55 Rosetta score terms. The global average pooling layer computes the mean of the per-residue encodings, which are output from the encoder, to produce a sequence-level representation of the same size as the embedding dimension. This sequence-level encoding is fed into a fully-connected layer with 256 hidden nodes for the local model and 512 hidden nodes for the global model. We used the rectified linear unit (ReLU) activation function for the transformer encoder and final fully connected layer.

We implemented relative position embeddings as described by Shaw et al. (Shaw et al., 2018). In contrast to the absolute position encoding used in the original transformer architecture (Vaswani et al., 2017), the relative position embedding enables the network to consider positional representations of the inputs in terms of distances between sequence positions. We consider two distinct ways to encode relative distances, generating what we refer to as 1D positional embeddings and 3D positional embeddings. In the 1D approach, relative distances are based on the protein amino acid sequence alone. This approach is identical to the implementation of relative position embeddings described by Shaw et al. In the 3D approach, relative distances are based on the 3D protein structure.

In the 1D approach, we calculate relative distances by determining the offset between each pair of sequence positions (i, j) in the input. The relative distance is

defined as $d = j - i$, representing how far sequence position j is relative to position i. A negative value signifies that j precedes i in the sequence, and a positive value signifies that j succeeds i. We map each of the possible relative distances to a pair of learnable embedding vectors, corresponding to attention keys and values. When calculating attention between sequence positions i and j, we add the key and value positional embedding vectors to the keys and values, respectively. As was hypothesized by Shaw et al., precise relative position information might not be useful beyond a certain distance. Thus, we clipped the possible relative distances to ± 8.

In the 3D approach, we calculate relative distances using the protein 3D structure instead of the amino acid sequence. When using 3D relative position embeddings, the model requires a protein structure in the form of a PDB file, corresponding to the base protein that the input variant sequence is based on. We first represent the protein structure as an undirected graph, where each node corresponds to a residue. We place an edge between any pair of nodes if the beta carbon atoms $(C\beta)$ of the residues are within 8Å of each other in the 3D space. We define the relative distance between residues (i, j) as the minimum path length from node i to node j in the graph. Unlike the 1D approach, relative distances computed using the 3D approach cannot be negative values. We clip the 3D relative distances at 3, effectively transforming distances greater than 3 into a relative distance of 3. A relative distance of 0 represents a node with itself, 1 signifies direct neighbors, 2 signifies second degree neighbors, and 3 encapsulates any other node not covered by the previous categories. As in the 1D approach, each possible relative distance

in the 3D approach is mapped to a pair of embedding vectors corresponding to keys and values. These vectors are learned during training and are added to keys and values during the attention calculation.

METL source model training

We split the Rosetta source data into randomly sampled train, validation, test, and withheld sets. For each dataset, we first withheld 5% of the data, to be used as a final test set. We split the remaining data into 80% train, 10% validation, and 10% test sets.

We trained source models for 30 epochs using the AdamW optimizer with a learning rate of 0.001. We applied a linear warm-up learning rate scheduler, with a warm-up period of 2% of the total training steps. Additional AdamW hyperparameters were $weight\_decay = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. We computed mean squared error loss independently for each of the 55 prediction tasks (corresponding to the 55 Rosetta energy terms) and took the sum to compute the final loss for the network. We applied gradient norm clipping with a max norm of 0.5. We employed distributed data parallel (DDP) training with 4 GPUs. We trained local source models with an effective batch size of 2048 (512 x 4 GPUs) and global source models with an effective batch size of 1024 (256 x 4 GPUs).

The global source data contains variants of 148 base sequences, with most having different sequence lengths. This complicates the process of encoding data into a single fixed-length batch. Padding is a commonly employed approach in such scenarios. However, incorporating different sequence lengths and base structures

in a single batch would negatively impact efficiency of computing attention with our implementation of relative position embeddings. Thus, we implemented a PDB-based data sampler that ensures each batch only contains variants from a single base PDB structure. Due to the use of DDP training with 4 GPUs, each aggregated training batch effectively contains variants from 4 base PDBs.

Experimental datasets

We evaluated our method on experimental datasets representing proteins of varying sizes, folds, and functions: avGFP (Sarkisyan et al., 2016), DLG4 (Nedrud et al., 2021), DLG4-2022 (Faure et al., 2022), GB1 (Olson et al., 2014), GRB2-Abundance (Faure et al., 2022), GRB2-Binding (Faure et al., 2022), Pab1 (Melamed et al., 2013), TEM-1 (Gonzalez and Ostermeier, 2019), and Ube4b (Starita et al., 2013) (Table 3.1). We acquired raw datasets from published manuscript supplements, MaveDB (Esposito et al., 2019), and NCBI GEO (Barrett et al., 2013). We transformed raw data into a standardized format, making sure that functional scores were log-transformed, normalized so that the wild-type score is 0, and rounded to 7 decimal places. We removed variants with mutations to stop codons and converted variant indexing to be 0-based. For DLG4 and GB1, we filtered variants to ensure a minimum number of reads. See Table B.2 for additional details about dataset transformations. We opted to use the DLG4-2022 dataset instead of the DLG4 dataset in our main analysis due to potential problems with the reliability of the functional scores in the DLG4 dataset.

We used GB1 as an exploratory dataset during method development to make

modeling decisions such as at what size validation set to enable model selection, where to place the prediction head on the source model, whether to use a linear or nonlinear prediction head, and others. Due to this, there is potential we overfit to GB1 and that our final results are optimistic for GB1. That said, we took precautions to limit the potential impact of using GB1 as our development dataset. The results presented for the small training set size experiment use a test set that was completely held out, even during method development. The randomly sampled train and validation sets used to generate the final results are also different splits than the ones we used during method development. Additionally, the results presented for the extrapolation experiments use different splits than the ones we used to test extrapolation during method development.

We adjusted the avGFP dataset preprocessing after seeing early small training set size results. Performance was lower than expected, which led us to realize that the dataset scores were not normalized so wild-type is 0. We modified the avGFP dataset to normalize the scores and set wild-type to 0 by subtracting the wild-type score from all the scores. All our other datasets were already normalized so wild-type is 0.

METL target model architecture

Target models are made up of a backbone and a head. The backbone contains network layers from the source model, pre-trained to predict Rosetta energies. The head is a new, randomly-initialized linear layer placed on top of the backbone to predict experimental functional scores. We also added a dropout layer with dropout

rate 0.5 between the backbone and the head. For local source models, we attach the head immediately after the final fully connected layer. For global source models, we attach the head immediately after the global pooling layer. Target models have a single output node corresponding to the experimental functional score prediction.

METL target model training

We implemented two training strategies for target models: feature extraction and finetuning. Feature extraction is a training strategy where only the head is trained, and the backbone weights are not updated during the training process. In contrast, finetuning is a training strategy where both the backbone and head weights are updated during training. For feature extraction, we trained the head using scikit-learn ridge regression with $\mathtt{alpha} = 1.0$. This provides a closed-form solution for the ridge regression weights.

For finetuning, we implemented a dual-phase finetuning strategy (Kumar et al., 2022). In the first phase, we froze the backbone and trained only the head for 250 epochs. In the second phase, we trained both the backbone and the head for an additional 250 epochs at a reduced learning rate. We used the AdamW optimizer with a learning rate of 0.001 in the first phase and 0.0001 in the second phase. We applied a learning rate scheduler with linear warm-up and cosine decay to each phase, with a warm-up period of 1% of the total training steps. Additional AdamW hyperparameters were set as follows: $\mathtt{weight\_decay} = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. We used a batch size of 128 and mean squared error loss. We applied gradient norm clipping with a max norm of 0.5.

After the full training period, we selected the model from the epoch with the lowest validation set loss. We only performed model selection if the validation set size was $\geqslant$ 32 for METL-L and $\geqslant$ 128 for METL-G and ESM. We found the optimization was more stable for METL-L than METL-G and ESM, thus smaller validation sets were still reliable. For validation sets smaller than those thresholds, we did not perform model selection, and instead we used the model from the last epoch of training. We determined these thresholds using the GB1 dataset, which we designated as our development dataset, by selecting the dataset size along the learning curve where using model selection started to outperform not using model selection. In retrospect, these thresholds were too low for other datasets.

Target model dataset splits

We created comprehensive train, validation, and test splits to evaluate performance with small training set sizes and a range of extrapolation tasks, including position, mutation, score, and regime extrapolation. For small training set sizes, we first sampled a random 10% test set from each full dataset. Then, from the remaining data, we sampled datasets of sizes 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240, and 20480. To account for especially easy or difficult training sets that may be sampled by chance, we generated multiple replicates for each dataset size. The number of replicates decreases as the dataset size increases: 101 replicates for the smallest dataset size, followed by 23, 11, 11, 11, 11, 7, 7, 5, 5, 3, and 3 replicates for the largest dataset size. We split the sampled datasets into 80% train and 20% validation sets. We used the same test set across all dataset sizes and replicates. We

report median performance metrics across replicates.

Whereas the small dataset splits are sampled randomly, the extrapolation splits are specially designed to assess the models' ability to generalize to more challenging test sets. For position, mutation, and score extrapolation, we randomly resampled any datasets with > 50000 variants down to 50000 variants before generating the extrapolation splits. To account for chance, we generated 9 replicate splits for each extrapolation type. We report the median across the 9 replicates.

Position extrapolation tests the ability of a model to generalize to sequence positions not present in the training data. To generate position extrapolation splits, we first randomly designated 80% of sequence positions as train and the other 20% as test. Then, we divided variants into training and testing pools depending on whether the variants contain mutations only in positions designated as train or only in positions designated as test. We discarded variants that had mutations in both train and test positions. To create the final train, validation, and test sets, we split the train pool into a 90% train set and a 10% validation set. We used the entire test pool as the test set.

Mutation extrapolation tests the ability of a model to generalize to mutations not present in the training data. To generate mutation extrapolation splits, we followed a similar procedure as position extrapolation, except with mutations instead of sequence positions. We randomly designated 80% of mutations present in the dataset as train and the other 20% as test. We divided variants into training and testing pools depending on whether the variants contain only mutations designated as train or only designated as test. We split the train pool into a 90% train and a

10% validation set and used the entire test pool as the test set.

Score extrapolation tests the ability of a model to generalize from low-scoring variants to high-scoring variants. We divided variants into train and test pools depending on whether the variant had a score less than wild-type (train pool) or greater than wild-type (test pool). We split the train pool into a 90% train and a 10% validation set and used the entire test pool as the test set.

Regime extrapolation tests the ability of the model to generalize from lower numbers of amino acid substitutions to higher numbers of amino acid substitutions. For datasets with single and double substitution variants, we divided the variants into a train pool comprising of the single substitution variants and a test pool comprising of the double substitution variants. We split the train pool into into an 80% train and a 20% validation set. We sampled a 10% test set from the test pool. For datasets containing greater than double substitution variants, we also implemented a regime extrapolation split where the train pool was comprised of single and double substitution variants and the test pool was comprised of variants with three or more substitutions.

Baseline models

We implemented and evaluated additional baselines, including: Linear-OH, a fully connected network (FCN), a sequence convolutional network (CNN), METL-Local with random initialization, Rosetta's *total score* as a standalone prediction, and Linear-RTS.

Linear-OH is a linear regression model that uses one hot encoded sequences

as inputs. One hot encoding captures the specific amino acid at each sequence position. It consists of a length 21 vector where each position represents one of the possible amino acids or the stop codon. All positions are zero except the position of the amino acid being encoded, which is set to a value of one. Note that we removed variants containing mutations to the stop codon during dataset preprocessing, so this was feature not used in our analysis. We implemented linear regression using scikit-learn's ridge regression class, which incorporates L2 regularization. We set the solver to *cholesky* to calculate a closed-form solution for the ridge regression weights. We set *alpha*, the constant that controls regularization strength, to the default value of 1.0. We set all other parameters to the default scikit-learn values.

For baseline neural networks, we tested an FCN, a CNN, and a transformer encoder with a similar architecture as METL-Local, but with a random initialization. The FCN and CNN used one hot encoded sequences as input. The FCN consisted of 1 hidden layer with 1024 nodes followed by a dropout layer with a dropout rate of 0.2. The CNN consisted of 1 convolutional layer with kernel size 7, 128 filters, and zero-padding to ensure the output has the same shape as the input (padding mode "same" in PyTorch's Conv2d class). Following the convolutional layer, we placed a fully connected layer with 256 nodes and a dropout layer with a dropout rate of 0.2. We used the ReLU activation function for both models. In addition to the FCN and CNN, we tested a randomly initialized transformer encoder neural network with a similar architecture as METL-Local. Unlike METL-Local, this randomly initialized version was set up with a single output node corresponding to the experimental functional score instead of multiple output nodes corresponding to Rosetta scores.

We trained the FCN, CNN, and randomly initialized METL-Local for 500 epochs using the AdamW optimizer with a base learning rate of 0.001. We applied a learning rate scheduler with linear warm-up and cosine decay, with a warm-up period of 2% of the total training steps. Additional AdamW hyperparameters were set as follows: $weight\_decay = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. We used a batch size of 128 and mean squared error loss. We applied gradient norm clipping with a max norm of 0.5. Similar to METL-Local finetuning, we selected the model from the epoch with the lowest validation loss when the validation set size was $\geqslant 32$. Otherwise, we used the model from the last epoch of training.

We evaluated Rosetta's *total score* as a standalone, unsupervised prediction, as well as an additional input feature for linear regression, which we refer to as Linear-RTS. By default, the lower Rosetta's *total score*, the more stable the structure is predicted to be. Thus, when using Rosetta's *total score* as an unsupervised prediction, we multiplied it by -1 before computing correlation with the experimental functional score. We also tested Rosetta's *total score* as part of a supervised learning framework. Linear-RTS is identical to Linear-OH, but it uses Rosetta *total score* as an additional input feature in combination with the one hot encoded sequence. We standardized the *total score* for use as an input feature by first calculating its mean and standard deviation in the train set. Then, we subtracted the mean and divided by the standard deviation.

Comparison to ESM

We used the ESM-2 (Lin et al., 2023) 35M parameter model with identifier esm2_t12_35M_UR50D as our default ESM model. We incorporated several additional layers to match the METL architecture, including a global mean pooling layer, a dropout layer with dropout rate 0.5, and a linear prediction head. We attached these additional layers immediately after layer 12. We trained the ESM models using the same training procedures we used for the METL models. We also explored feature extraction with larger 150M and 650M parameter ESM-2 models with identifiers esm2_t30_150M_UR50D and esm2_t33_650M_UR50D. For these larger models, we attached the additional layers after layers 30 and 33, respectively.

Comparison to EVE

We obtained a multiple sequence alignment for each target protein through the EVcouplings web server (Hopf et al., 2019), using search parameters consistent with EVMutation (Hopf et al., 2017). The multiple sequence alignments were found on the 11/22 release of UniRef100. We trained EVE using the default training parameters of 40,000 training iterations and the average of 20,000 evolutionary indices. In addition to using the EVE score as a standalone unsupervised method, we incorporated the EVE score into a supervised learning framework (Hsu et al., 2022). The Linear-EVE model is identical to the Linear-OH model described above, but it uses the EVE score as an additional input feature in combination with the one hot encoded protein sequence. We standardized the EVE score for use as an input feature by first calculating its mean and standard deviation in the train set.

Then, we subtracted the mean and divided by the standard deviation.

avGFP sequence design

We finetuned a pretrained METL-Local model on 64 randomly sampled variants from the avGFP dataset. The selected variants had 1 to 11 mutations, and their experimental score distribution was bimodal, similar to the distribution of the full avGFP dataset. We refer to the fine-tuned model as METL-L-avGFP.

We performed in-silico optimization with METL-L-avGFP to design a total of 20 variants distributed evenly across for 4 different design criteria. These criteria are the product of 2 primary design categories: the number of mutations (either 5 or 10), and the constraints on mutation selection (either Observed or Unobserved). In the Observed constraint, the designed sequences contain only amino acid substitutions found in the 64-variant training set. Conversely, in the Unobserved constraint, the designed sequences exclude any amino acid substitutions found in the 64-variant training set. The intersection of these categories resulted in the 4 design criteria: Observed 5-mutant, Unobserved 5-mutant, Observed 10-mutant, and Unobserved 10-mutant. We designed 5 sequences for each criteria, resulting in a total of 20 designed sequences.

To perform the in-silico optimization, we ran simulated annealing 10,000 times for each design criterion. For each iteration, we changed the seed of the random number generator, which proposed new variants, and executed the Monte Carlo optimization for 10,000 steps. Each step consisted of suggesting a mutation for the currently sampled variant and deciding whether to accept the new variant according

to the Metropolis-Hastings criteria. We decreased the optimization temperature according to a logarithmic gradient beginning at 10^1 and ending at 10^{-2}. The initial temperature was chosen by randomly sampling 10,000 variants, predicting their fitness with METL-L-avGFP, and calculating the absolute value of the difference between the lowest and highest predicted fitness, rounded to the nearest power of 10. The final temperature was determined by calculating the absolute value of the smallest difference in predicted fitness between any two variants, rounded to the nearest power to 10. The initial temperature encouraged acceptance of all variants, while the final temperature, set close to the resolution of the model's predictions, meant that only variants better than the current ones would be accepted near the end of the run. The simulation began by randomly selecting a variant with the necessary number of mutations depending on the design criterion. We determined how many mutations to change (mutation rate) at each step by sampling from a Poisson distribution.

The optimization process described above yielded 10,000 designs for each criterion, which we downsampled to 5 designs for each criterion via clustering. Our downsampling approach prioritized diversity and was predicated on the idea that repeated convergence to similar sequences was correlated with higher observed fitness values, as these regions of the fitness landscape would have broader peaks and allow more room for error in the model predictions or optimization process. We clustered the 10,000 designs using scikit-learn's agglomerative clustering with complete linkage and a BLOSUM62-based distance metric. After preliminary results indicated that selecting between 10, 20, or 50 clusters did not substantially impact

the diversity of selected mutations (our primary metric for evaluating algorithm design), we arbitrarily chose 20 as the initial number of clusters. We filtered out clusters that contained less than 100 sequences (1% of the simulated annealing solutions).

To further downsample to 5 clusters, we employed an iterative, greedy approach. First, we selected a representative sequence for each cluster, choosing the one with the lowest average BLOSUM62-based distance to all other sequences within the same cluster. Next, we selected the cluster containing the highest number of sequences. We then proceeded iteratively, selecting additional clusters one at a time. In each iteration, we calculated the distance between the representative sequences of the already selected clusters and the remaining unselected clusters. We kept the cluster with the largest mean distance to the already selected clusters.

Cloning and experimental validation of avGFP variants

We modeled our expression system on that used in Sarkysian et al. (Sarkisyan et al., 2016), which uses a pQE-30 vector (Qiagen) to express avGFP as a fusion protein with mKate2. To generate the expression construct, we used the vector backbone from a related pQE-30 system that expresses KillerOrange (Addgene 74748) and ordered the mKate2-avGFP fusion protein as a gene fragment from Twist Biosciences. We first removed a BsaI restriction site in the AmpR gene from the backbone using site directed mutagenesis (NEB: M0554S), then used Golden Gate cloning to replace KillerOrange with the fusion protein. We incubated (1 hr, 37 C) the backbone and insert with BsaI (15 U, NEB: R3733), T4 Ligase (1,000

U, NEB: M0202) and T4 Ligase Buffer (NEB B0202) to assemble the vector. The assembly was cleaned up with a PCR Clean and Concentrate column (Zymogen, D4003) and transformed into in-house DH5a cells. Plasmid stock was purified from an overnight culture starting from a single colony using a Qiagen Miniprep kit (Qiagen, 27104), and the vector was on-boarded with Twist Biosciences. All avGFP variants were ordered as clonal genes from Twist Biosciences wherein the wild-type avGFP sequence was replaced with the variant sequence. For each variant, the nucleotide sequence was kept the same as the wild-type sequence except at mutated residues. We selected new codons for mutated residues based on an E. coli codon usage index (Boël et al., 2016) to mitigate poor expression due to rare codons.

Clonal genes ordered from Twist Biosciences were transformed into NEBExpress Iq Competent *E. coli* (NEB: C3037I) cells and plated on Luria Broth (LB) plates with carbenecillin selection (0.1 mg/mL). Proteins were expressed as previously described in Sarkysian et al. (Sarkisyan et al., 2016). Briefly, freshly plated transformants were incubated overnight at 37 C and then moved to 4 C the following day. After 24 hours, plates were washed with 4 mL LB and normalized to 1 OD. This wash was used to create triplicate expression cultures where protein expression was induced for 2 hours with 1 mM IPTG at 23 C. An empty pQE-30 vector was used as a negative expression control.

To prepare cultures for fluorescence measurement, expression cultures were pelleted (3,000xg, 5 mins) and re-suspended in sterile 1X PBS to a concentration of 1 OD. Cells were diluted 2-fold into 96-well plates to measure fluorescence and culture density with the Tecan Spark 10M. Measurements for avGFP (ex. 405 nm,

em. 510 nm), mKate2 (ex. 561 nm, em. 670 nm), and OD600 (abs. 600 nm) were collected.

Fluorescence was reported as the ratio of avGFP fluorescence to mKate2 fluorescence averaged across replicates. First, fluorescent measurements were normalized to 1 OD based on the OD600 value. Background fluorescence was subtracted out of each sample. Background fluorescence for avGFP and mKate2 was calculated by averaging avGFP and mKate2 fluorescence in the normalized negative fluorescent control. The avGFP/mKate2 expression ratio was calculated for each sample by dividing the normalized avGFP fluorescence by normalized mKate2 fluorescence. At this point, a single fluorescence ratio was calculated for each design by averaging across the three replicates.

Acknowledgements

This research was supported by National Science Foundation awards 2226383 and 2226451, National Institutes of Health award R01GM135631, the John W. and Jeanne M. Rowe Center for Research in Virology at the Morgridge Institute for Research, and the University of Wisconsin–Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation. We thank Ben Gelman for insightful discussions regarding the transformer architecture, attention mechanism, and effects of data normalization. The research was performed using the compute resources and assistance of the University of Wisconsin-Madison Center for High Throughput Computing (Center for High

Throughput Computing, 2006) and services provided by the OSG Consortium (Pordes et al., 2007; Sfiligoi et al., 2009; OSG, 2006), which is supported by the National Science Foundation awards 2030508 and 1836650.

APPENDIX B: SUPPLEMENTARY INFORMATION FOR CHAPTER 3

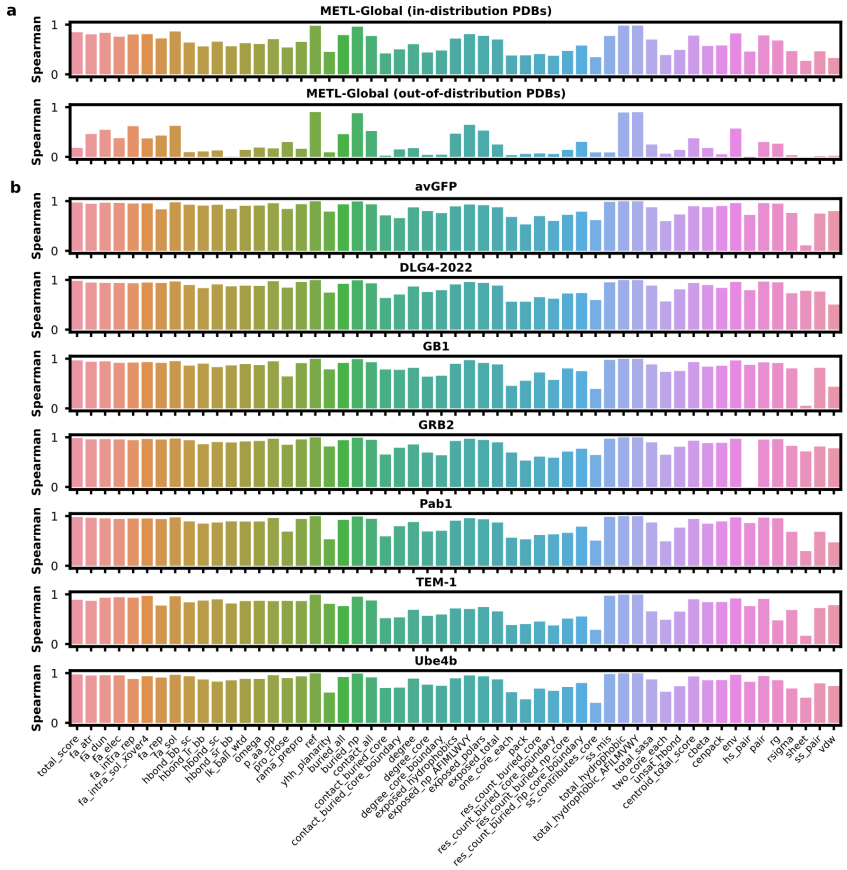

Figure B.1: **Performance of pretrained METL source models in predicting Rosetta scores.** This figure shows Spearman correlations between true and predicted Rosetta scores for each of the 55 Rosetta score terms. (a) Performance of METL-Global in predicting Rosetta scores for protein variants originating from in-distribution base PDBs (those included in METL-Global pretraining) and out-of-distribution base PDBs (those not included). We show the mean Spearman correlation across base PDBs. To evaluate in-distribution PDBs, we used variants in the pretraining data test set. To evaluate out-of-distribution PDBs, we used variants from the eight DMS datasets included in this study. METL-Global performs substantially better for in-distribution PDBs, suggesting there is overfitting to the PDBs present in the training data. (b) Correlation performance of METL-Local models predicting Rosetta energy terms for the local pretraining data test sets.

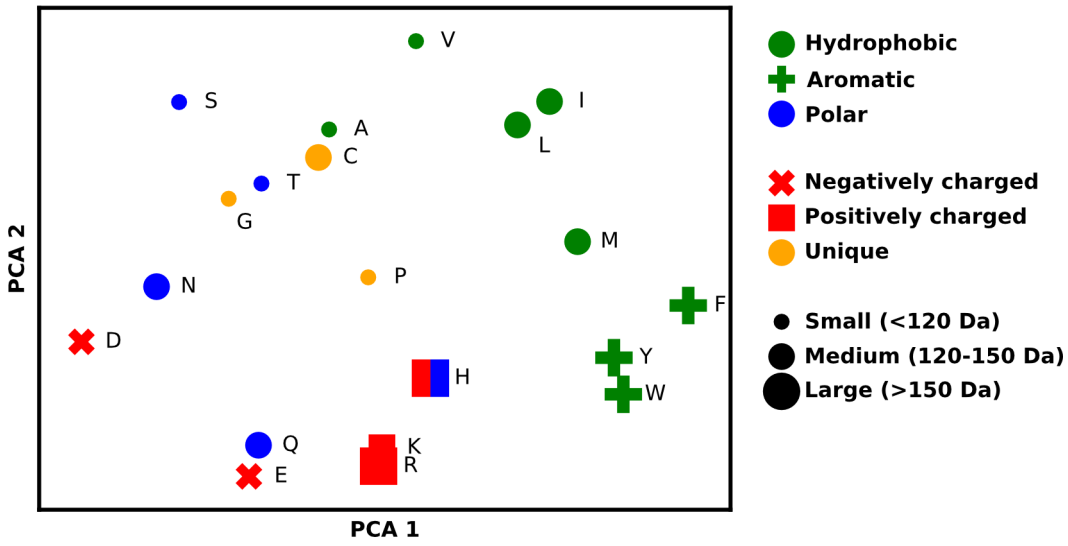

Figure B.2: **METL-Global amino acid embeddings** We applied principle component analysis (PCA) to reduce the METL-Global length 512 amino acid embeddings down to 2 dimensions, capturing 33% of the variance in data. This scatter plot of the 2-dimensional amino acid embeddings is annotated with amino acid properties. Amino acids with similar properties are grouped together in the embedding space.

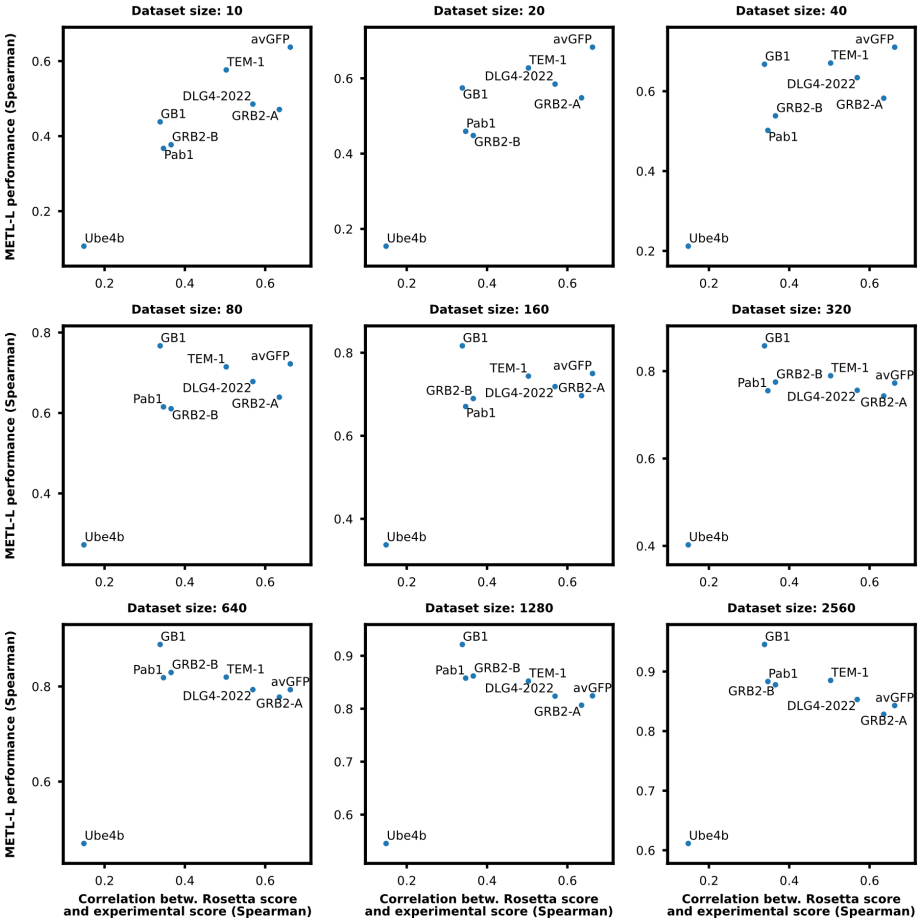

Figure B.3: **Relationship between METL-Local performance and the relatedness of Rosetta and experimental scores.** The figure displays a series of scatterplots showing the relationship between METL-Local performance and the relatedness of Rosetta and experimental scores, across multiple experimental datasets and training set sizes. The x-axis shows the Spearman correlation between Rosetta total score and the experimental functional score for the entire dataset, representing the relatedness or similarity between the Rosetta total score and the experimental functional score. The y-axis shows the METL-Local performance for the respective training set size, as determined by the Spearman correlation on the test set. Notably, as the similarity between Rosetta total score and the experimental functional score increases, so does the METL-Local performance, at least for small training set sizes. However, with increasing experimental training set sizes, the similarity between Rosetta total score and experimental functional score becomes less important to the METL-Local performance, suggesting a shift in METL-Local away from the Rosetta pretraining data and more toward the experimental finetuning data.

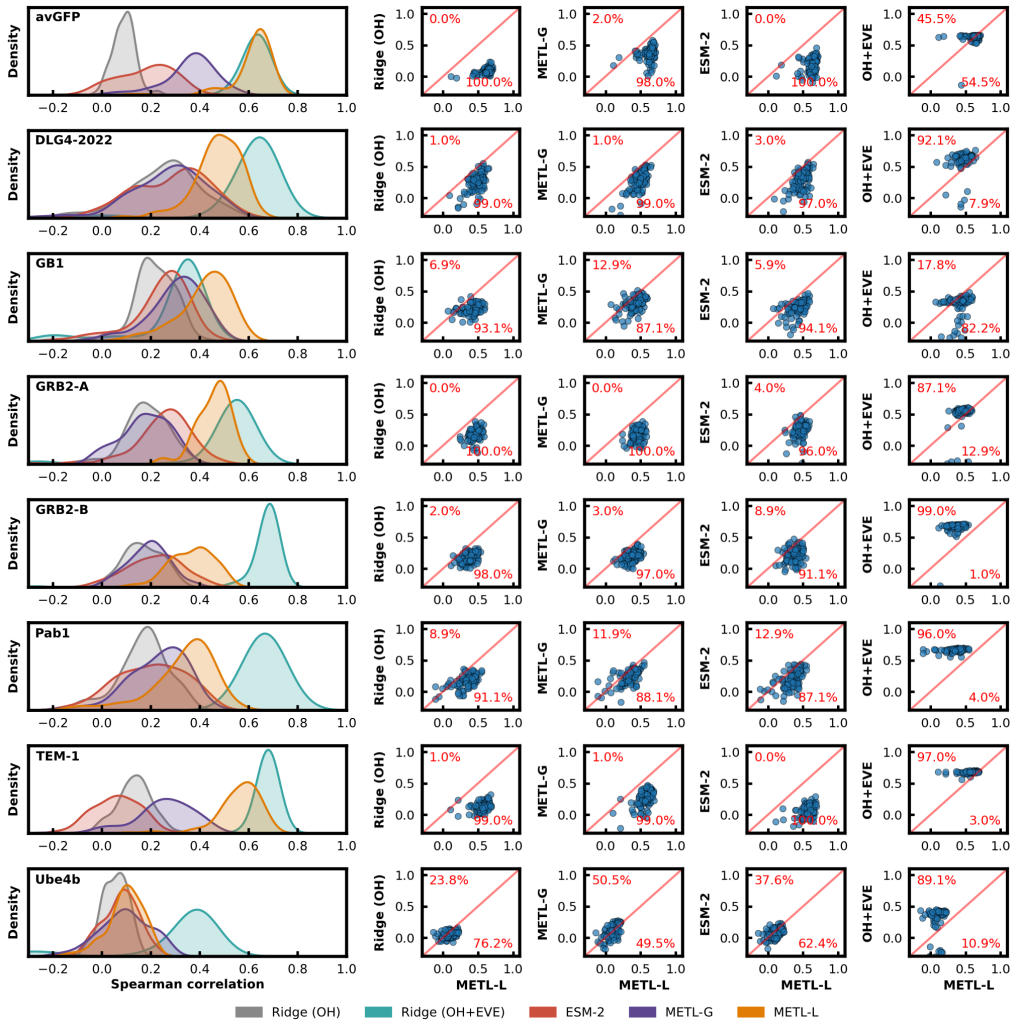

Figure B.4: **Performance of 101 training set replicates for training set size 8.** The left panel consists of kernel density estimation plots showing the distribution of test set performance (Spearman correlation) of 101 training set replicates for training set size 8. The selection of training set examples can have a substantial impact on performance for this small training set size. The right panel consists of scatterplots showing individual training set replicates with the performance of METL-L (Spearman correlation) on the x-axis and the performance of the other methods (Spearman correlation) on the y-axis. We annotated the scatterplots with the line of equivalence and the percentage values showing the fraction of replicates for which METL-L has stronger performance (bottom right quadrant) versus the fraction of replicates for which the other respective method has stronger performance (top left quadrant).

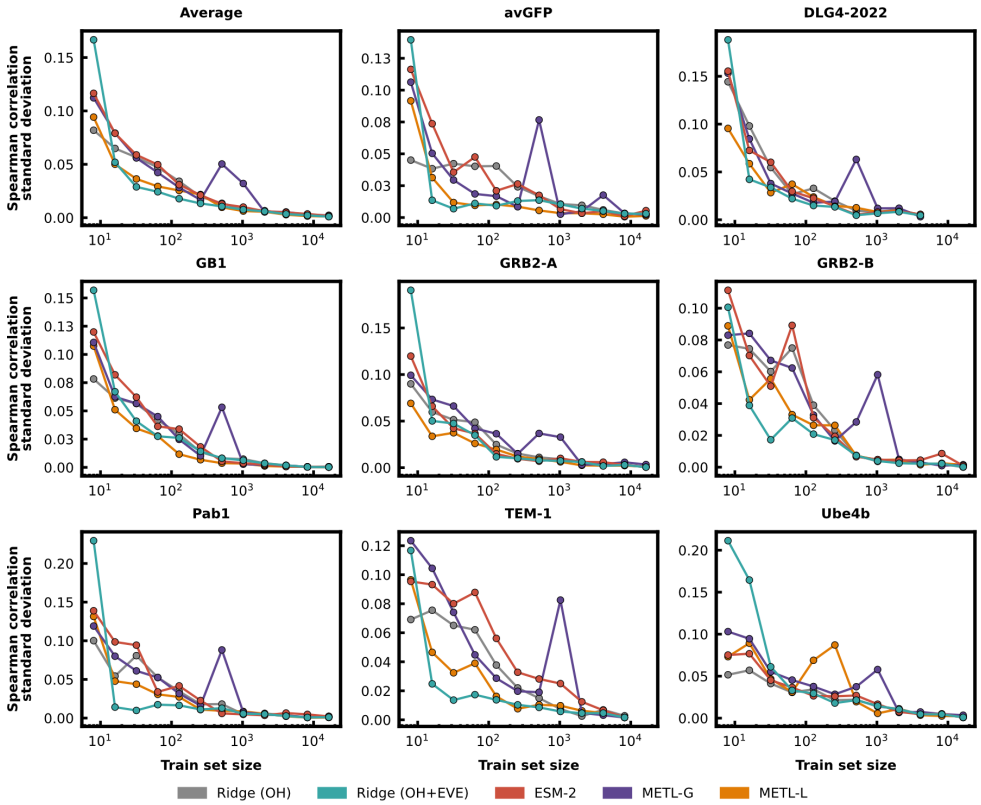

Figure B.5: **Standard deviation of performance across training set replicates.** We tested numerous replicates for each train set size: 101 replicates for the smallest train set size, followed by 23, 11, 11, 11, 11, 7, 7, 5, 5, 3, and finally 3 replicates for the largest train set size. This figure shows the standard deviation of performance, as measured by the test set Spearman correlation between true and predicted scores, across the train set replicates. As expected, the standard deviation decreases as the size of the training set increases. We observe that for small training set sizes (> 8), METL-Local and Ridge (OH+EVE) tend to have a smaller standard deviation than the other methods, signifying they are less sensitive than the other methods to the selection of train set examples. We note that METL-Global exhibits a spike in standard deviation at train set size 512, which is the train set size at which we enable early stopping based on the validation set loss. At this train set size, the validation set size is 128. Given our finetuning approach and the instability of training the 20M parameter METL-Global model, this validation set size may not be large enough to use reliably for early stopping, resulting in higher standard deviations in performance of the trained models. The Ube4b dataset also shows this spike for METL-Local at train set size 128, which is the train set size at which we enable early stopping for METL-Local, with a validation set size of 32.

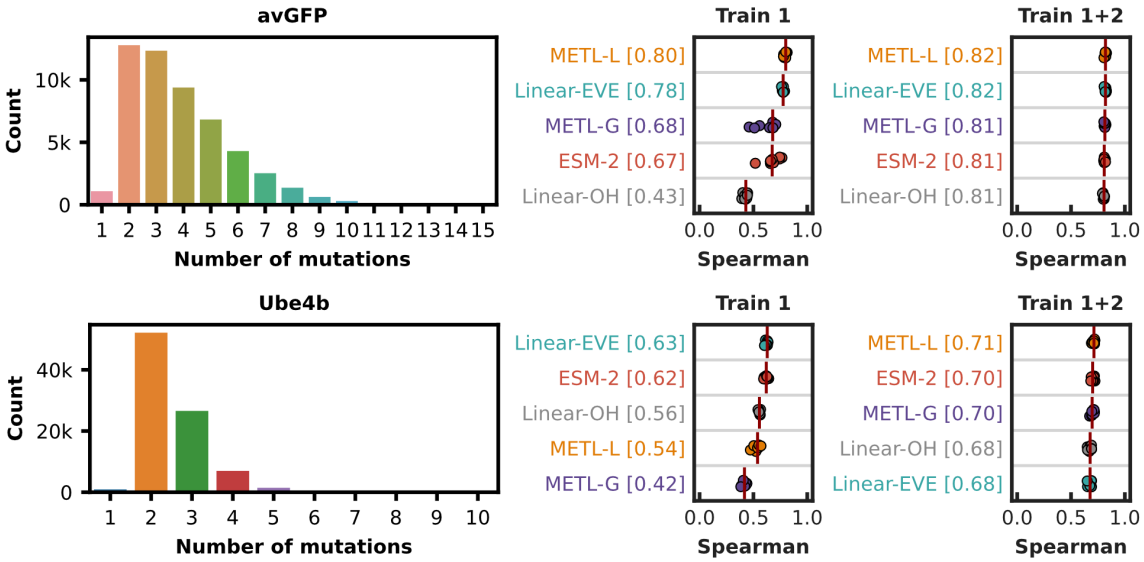

Figure B.6: **Regime extrapolation for avGFP and Ube4b datasets.** The avGFP and Ube4b datasets contain variants with higher order mutations, enabling us to test two types of regime extrapolation: Train 1 and Train 1+2. The bar plots (left) show the count of variants with the specified number of mutations for each dataset. The strip plots (right) show the performance of regime extrapolation for Train 1, where we train on single substitution variants and evaluate on variants with 2+ substitutions, and Train 1+2, where we train on variants with single or double substitutions, and evaluate on variants with 3+ substitutions. The strip plots show the performance of 9 test set replicates, and the red vertical line denotes the median.

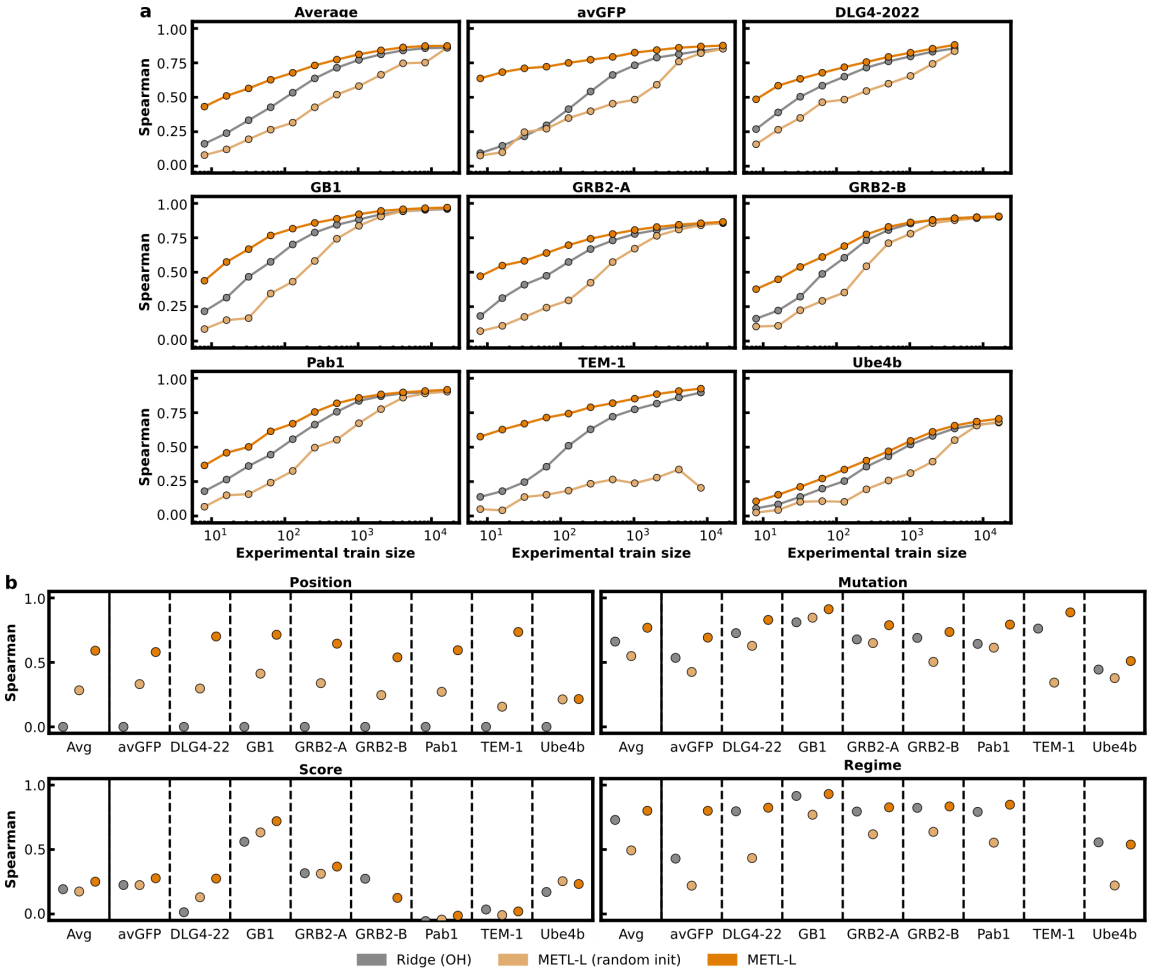

Figure B.7: **Performance of METL-Local with and without pretraining.** These plots show the correlation performance of Ridge (OH), METL-Local (random init), and METL-Local. METL-Local (random init) is a model with the same architecture as METL-Local but without pretraining on Rosetta scores. (a) The learning curves show that METL-Local (random init) substantially underperforms both Ridge (OH) and pre-trained METL-Local, emphasizing the impact pretraining on Rosetta scores has on this transformer-based architecture. Given enough experimental training data, METL-Local (random init) converges to the performance of the other models for most datasets. (b) METL-Local (random init) outperforms Ridge (OH) for position extrapolation due to the fact that Ridge (OH) is not able to perform position extrapolation. For the other types of extrapolation, METL-Local (random init) performs about the same or worse than Ridge (OH).

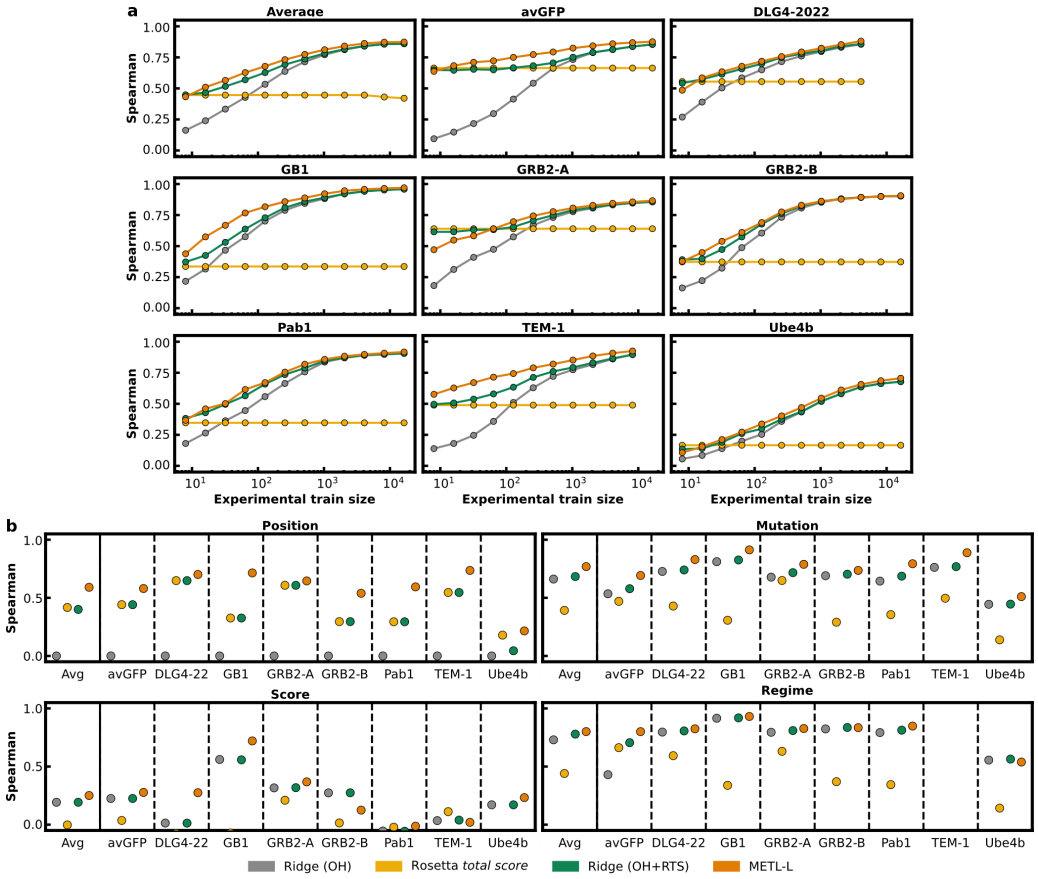

Figure B.8: **Performance of baseline models directly using Rosetta's total score.** Rosetta *total score* is the score term from Rosetta with no supervised training on experimental data. Ridge (OH+RTS) is a linear ridge regression trained on experimental data with one hot encoding features augmented with the Rosetta *total score* as an additional input feature. Both of these models require running Rosetta to compute the *total score* for every variant, even during inference. For comparison, this figure also shows the performance of Ridge (OH) and METL-Local. (a) For small training set sizes, incorporating Rosetta *total score* as an additional input feature for ridge regression greatly improved performance over solely using one hot encoding features, as demonstrated by the difference in performance between Ridge (OH) and Ridge (OH+RTS). While Ridge (OH+RTS) sometimes matched METL-Local's performance and even exceeded it on the GRB2-A dataset, METL-Local still outperformed Ridge (OH+RTS) on average. (b) METL-Local outperformed Ridge (OH+RTS) across most datasets and extrapolation tasks. The performance differences were sometimes substantial, such as for position extrapolation with GB1. In other cases, the performance differences were much smaller, such as for regime extrapolation.

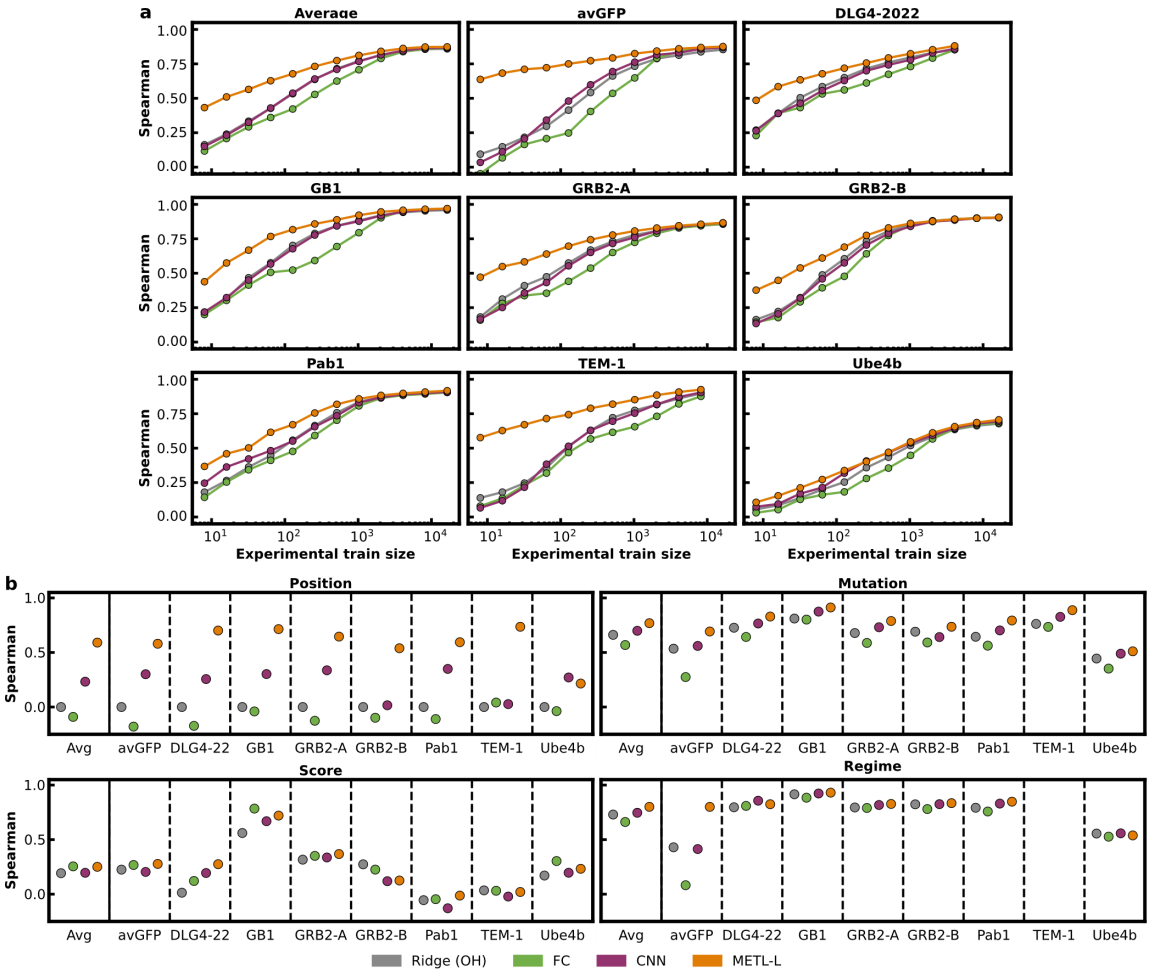

Figure B.9: **Performance of additional baseline models.** Correlation performance of Ridge (OH), fully connected networks (FC), sequence convolutional networks (CNN), and METL-Local. (a) The CNN performed about the same as Ridge (OH) across different sized training sets. The fully connected network typically performed about the same or worse than Ridge (OH), especially for mid-size training sets. (b) The CNN performed about the same or better than Ridge (OH) across most extrapolation tasks and datasets. The fully connected network performed worse than Ridge (OH), with some outliers, like for GB1 score extrapolation, where it performed better than any of the other tested models.

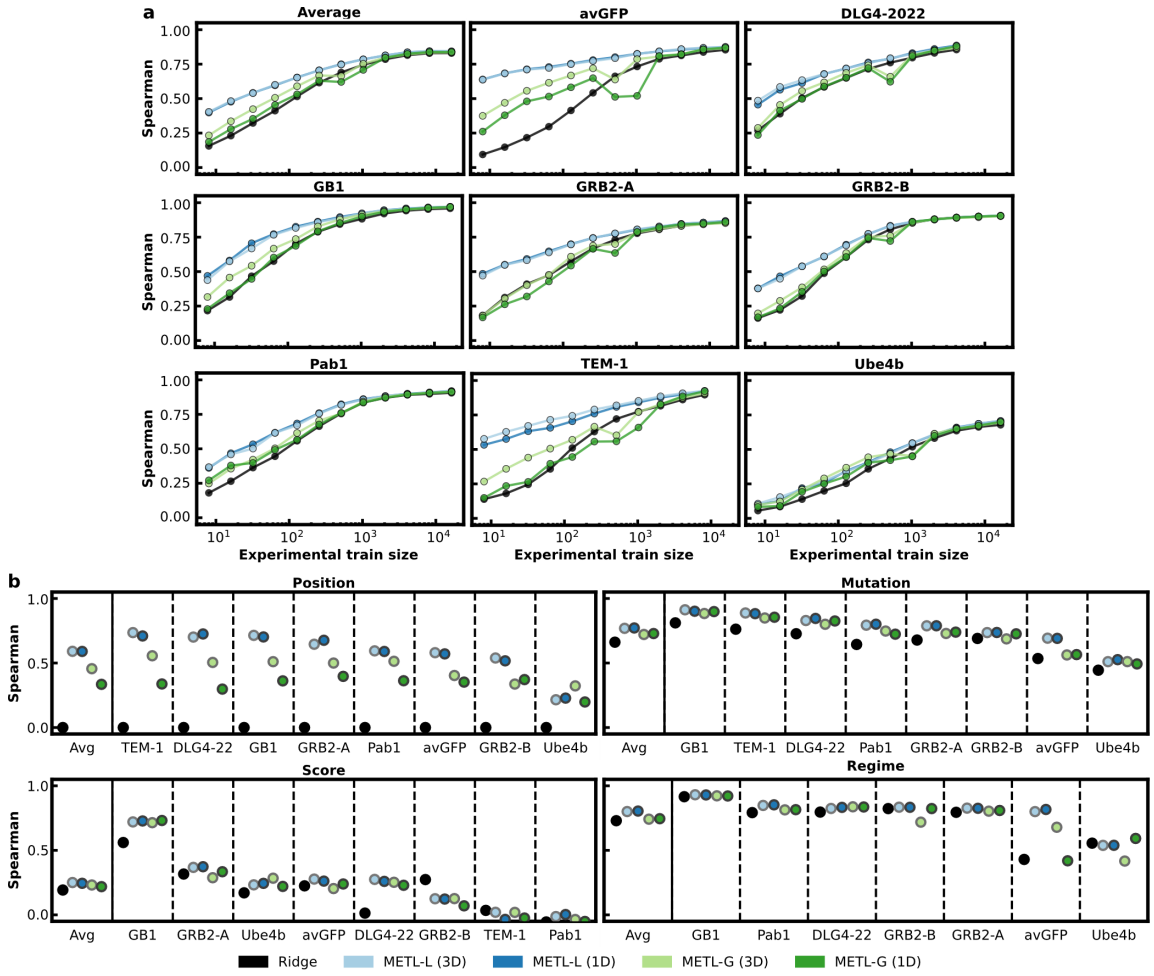

Figure B.10: **Performance of one-dimensional and three-dimensional relative position embeddings.** This figure shows the performance of METL-Local and METL-Global with one-dimensional (1D), sequence-based and three-dimensional (3D), structure-based relative position embeddings. (a) Learning curves showing Spearman correlation between true and predicted scores across a range of training set sizes. (b) Spearman correlation between true and predicted scores for position, mutation, score, and regime extrapolation. For both panels, the "Average" or "Avg" represents the mean across all datasets. Overall, METL-Local does not benefit much from three-dimensional embeddings over one-dimensional (except for the TEM-1 dataset), while METL-Global shows consistent improvement with the three-dimensional embeddings.

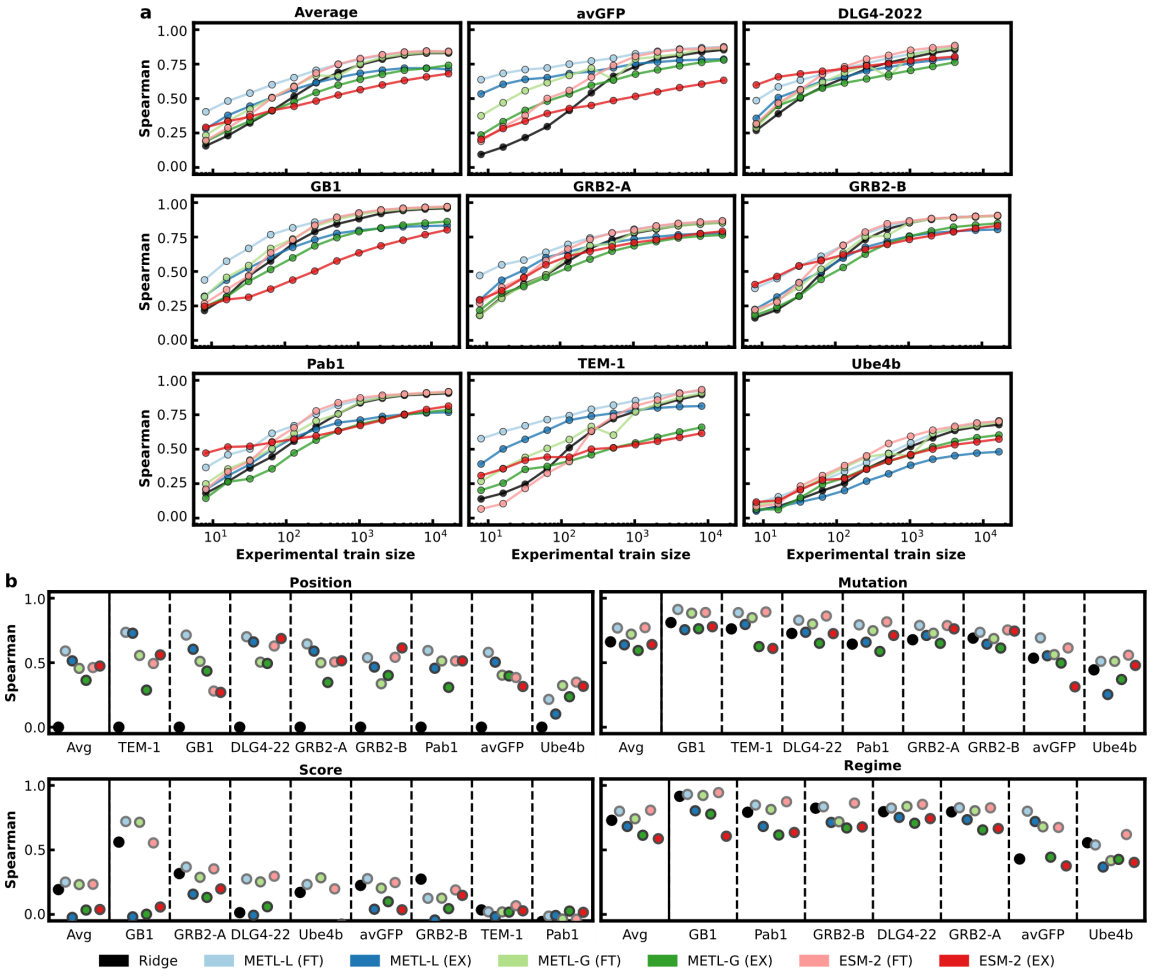

Figure B.11: **Performance of finetuning and feature extraction.** This figure shows the performance of METL-Local, METL-Global, and ESM-2 with both finetuning (FT) and feature extraction (EX). To perform feature extraction, we saved outputs from the appropriate internal layer of each model and then used those features as inputs to train linear ridge regression. Finetuning consistently outperformed feature extraction for METL-Local and METL-Global across (a) different training set sizes and (b) extrapolation tasks. For ESM-2, there were several instances where feature extraction substantially outperformed fine-tuning when applied to (a) small training set sizes, namely for the DLG4-2022, GRB2-B, Pab1, and TEM-1 datasets. Notably, the performance of ESM-2 feature extraction exceeded the performance METL-Local finetuning for DLG4-2022 and Pab1 with small training set sizes. For (b) extrapolation tasks, ESM-2 finetuning generally performed better than feature extraction.

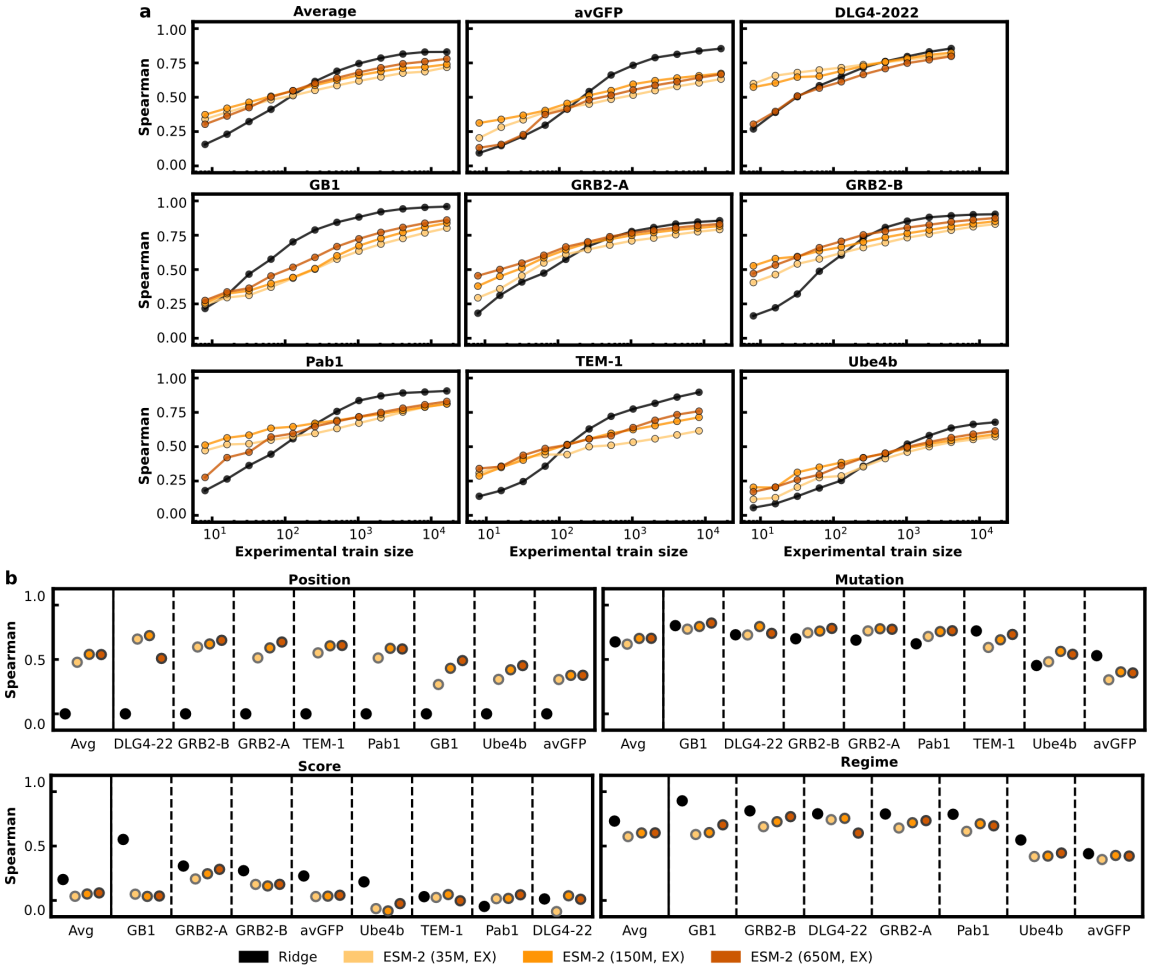

Figure B.12: **Feature extraction performance of ESM-2 models with 35M, 150M, and 650M parameters.** (a) Across the range of training set sizes, the 150M parameter model consistently outperformed the 35M parameter model, with the exception of the DLG4-2022 dataset, where the 35M parameter model actually performed better. Surprisingly, for small training set sizes, the 650M parameter model performed worse than both the 35M and 150M parameter models with the avGFP, DLG4-2022, and Pab1 datasets. For larger training set sizes, the 650M parameter model offered some improvement over the 35M and 150M parameter models with the GB1, GRB2-A, and GRB2-B datasets. (b) Across extrapolation tasks, the 35M parameter model tended to perform worse than the 150M and 650M parameter models. The 650M parameter model often performed the best, but not in all instances, and the differences between the models were minor in some cases.

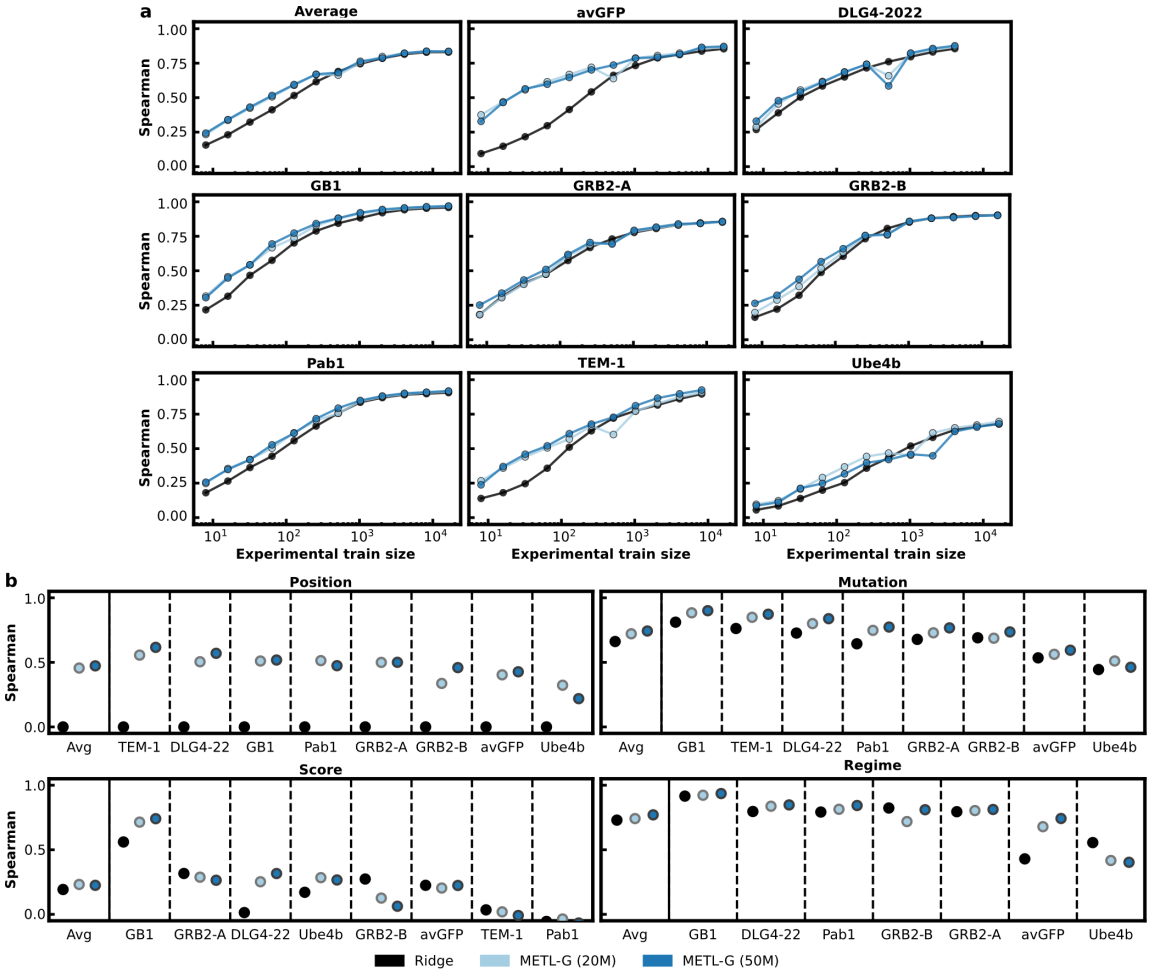

Figure B.13: **Performance of METL-G with 20M and 50M parameters** (a) Across different training set sizes, the 50M parameter model performed similarly to the 20M parameter model on average, with the 50M parameter model offering minor improvements for some datasets like GRB2-B but also performing slightly worse for other datasets like Ube4b. (b) For position, mutation, and regime extrapolation, the 50M parameter model performed slightly better on average than the 20M parameter model. For score extrapolation, the two models performed similarly on average.

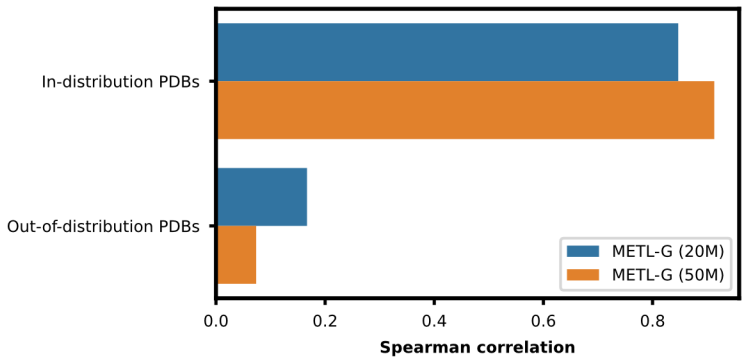

Figure B.14: **Performance of METL-G source models predicting Rosetta's *total score*.** This figure shows the performance of 20M and 50M parameter METL-G source models on predicting Rosetta's *total score* for both in-distribution and out-of-distribution PDBs. In-distribution PDBs are the ≈150 PDBs that were used as part of the METL-G pretraining data, while out-of-distribution PDBs consist of the experimental dataset PDBs, which were not used for METL-G pretraining. The 50M parameter METL-G model overfits more than the 20M parameter model when predicting Rosetta's *total score* on in-distribution PDBs, and it generalizes worse to out-of-distribution PDBs.

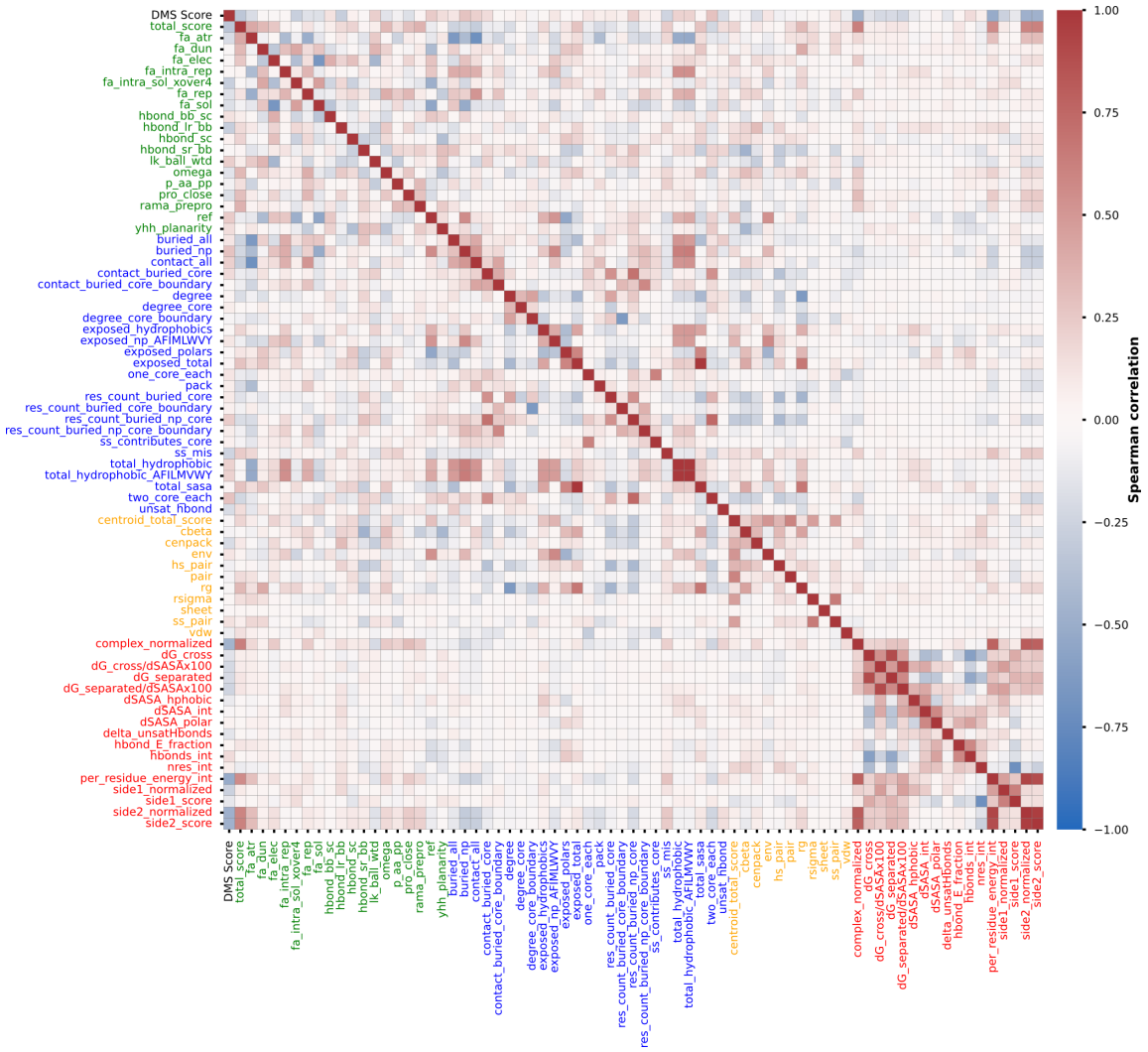

Figure B.15: **Pairwise correlations between GB1 DMS score and Rosetta scores.** Heatmap showing pairwise Spearman correlations between the GB1 experimental functional score (DMS Score) and Rosetta score terms. Rosetta scores are color coded, with green representing all-atom REF15 scores, blue representing filter scores, orange representing centroid score3 scores, and red representing InterfaceAnalyzer binding scores. Correlations were computed using the GB1 DMS variants.

ID	Constraint	# Muts	Mutations
1	Observed	5	S26R,K164R,Q175L,N196Y,G226W
2	Observed	5	S26R,I126V,K164R,Q175L,D195Y
3	Observed	5	K164R,Q175L,N196Y,A204T,G226R
4	Observed	5	S26R,S70G,Q175L,G226W,Y235H
5	Observed	5	K164R,Q175L,N196Y,G226R,I227F
6	Unobserved	5	N103I,I150W,V161I,G230K,K236R
7	Unobserved	5	L42V,N162R,L176W,D195C,L219F
8	Unobserved	5	D34W,F97Y,L176R,N183V,G230R
9	Unobserved	5	I12L,S26T,Q175V,A225R,G226K
10	Unobserved	5	P11R,N103I,V161M,G230W,L234M
11	Observed	10	S26R,I121V,I126V,K164R,Q175L,N196Y,S200N,S203T,A225G,G226W
12	Observed	10	S26R,F97S,N103S,I121V,K164R,Q175L,N196Y,G226R,I227F,Y235H
13	Observed	10	S26R,S70G,D100G,K105E,I126V,Q175L,N196Y,S203T,G226R,Y235H
14	Observed	10	S26R,K39R,S70G,I126V,Q175L,S200N,S203T,A225G,G226W,I227F
15	Observed	10	S26R,K105E,I126T,E140V,K164R,Q175L,N196Y,S203T,G226W,Y235H
16	Unobserved	10	F97W,V161M,S173E,Q175Y,Q182R,S200M,A204C,L219I,V222L,G230R
17	Unobserved	10	V9I,S28R,F97Y,N103I,L176R,N183V,H197F,L219W,A225W,G230Q
18	Unobserved	10	V9R,V91M,E93W,K105R,N162R,T184V,L193M,S203Q,G230M,L234W
19	Unobserved	10	D34W,I126E,L139M,E140R,Q175V,L193F,A204W,T228S,G230N,E233Q
20	Unobserved	10	P11H,E15N,S26E,S28I,I96W,S173K,Q175M,H197F,A225R,G230R

Table B.1: **avGFP designed sequences.** The sequences designed in the avGFP low-N design experiment.

	Acquired from	Files / URN / Accession	Variant filtering	Score transformation	Ref.
avGFP	Paper supplement	amino_acid_genotypes_to_brightness.tsv	Drop variants with mutations to stop codons	Normalized to WT by subtracting WT score	Sarkisyan et al. (2016)
DLG4	MaveDB	urn:mavedb:00000053-a	Keep if (inp >= 200) or (inp > 10 and sel >= 1)	N/A	Nedrud et al. (2021)
DLG4-2022	NCBI GEO	GSE184042	N/A	Normalized to WT by subtracting WT score	Faure et al. (2022)
GB1	Paper supplement	mmc2.xlsx	Keep if input_count + sel_count >= 5	Computed from read counts using Enrich2 Rubin et al. (2017)	Olson et al. (2014)
GRB2-Abundance	NCBI GEO	GSE184042	N/A	Normalized to WT by subtracting WT score	Faure et al. (2022)
GRB2-Binding	NCBI GEO	GSE184042	N/A	Normalized to WT by subtracting WT score	Faure et al. (2022)
Pab1	Paper supplement	Supplementary tables 2 and 5	N/A	Converted to log scores by taking log base 2	Melamed et al. (2013)
TEM-1	Paper supplement	mmc2.xlsx	N/A	Converted to log scores by taking log base 2	Gonzalez and Ostermeier (2019)
Ube4b	MaveDB	urn:mavedb:00000004-a-3	Drop variants with mutations to stop codons	N/A	Starita et al. (2013)

Table B.2: **Experimental datasets.** This table specifies the experimental datasets used in this study, where we acquired them from, and any filtering or transformations we applied to standardize the dataset format.

Dataset	Structure Acquired From	Structure Notes	Num Variants
avGFP	RosettaCM		18,681,329
DLG4	PDB: 6QJI	Not truncated	20,270,692
DLG4-2022	PDB: 6QJI	Not truncated	22,221,845
GB1	PDB: 2QMT		12,556,374
GRB2-A/B	AlphaFold DB: AF-P62993-F1-model_v4	Truncated to match DMS sequence	20,294,793
Pab1	RosettaCM		19,667,539
TEM-1	AlphaFold DB: AF-Q6SJ61-F1-model_v4		19,441,290
Ube4b	RosettaCM		19,734,229

Table B.3: **Local Rosetta datasets.** Information about the datasets used to train the local Rosetta source models, including PDB origin and the final number of variants in each dataset.

Rosetta score term	Description	Rosetta score term	Description
total_score	REF15	exposed_np_AFIMLWVY	Custom
fa_atr	REF15	exposed_polars	Custom
fa_dun	REF15	exposed_total	Custom
fa_elec	REF15	one_core_each	Custom
fa_intra_rep	REF15	pack	Custom
fa_intra_sol_xover4	REF15	res_count_buried_core	Custom
fa_rep	REF15	res_count_buried_core_boundary	Custom
fa_sol	REF15	res_count_buried_np_core	Custom
hbond_bb_sc	REF15	res_count_buried_np_core_boundary	Custom
hbond_lr_bb	REF15	ss_contributes_core	Custom
hbond_sc	REF15	ss_mis	Custom
hbond_sr_bb	REF15	total_hydrophobic	Custom
lk_ball_wtd	REF15	total_hydrophobic_AFILMVWY	Custom
omega	REF15	total_sasa	Custom
p_aa_pp	REF15	two_core_each	Custom
pro_close	REF15	unsat_hbond	Custom
rama_prepro	REF15	centroid_total_score	Centroid
ref	REF15	cbeta	Centroid
yhh_planarity	REF15	cenpack	Centroid
buried_all	Custom	env	Centroid
buried_np	Custom	hs_pair	Centroid
contact_all	Custom	pair	Centroid
contact_buried_core	Custom	rg	Centroid
contact_buried_core_boundary	Custom	rsigma	Centroid
degree	Custom	sheet	Centroid
degree_core	Custom	ss_pair	Centroid
degree_core_boundary	Custom	vdw	Centroid
exposed_hydrophobics	Custom		

Table B.4: **Rosetta score terms.** The Rosetta score terms used to train METL.

Rosetta score term	Description
complex_normalized	InterfaceAnalzyer
dG_cross	InterfaceAnalzyer
dG_cross/dSASAx100	InterfaceAnalzyer
dG_separated	InterfaceAnalzyer
dG_separated/dSASAx100	InterfaceAnalzyer
dSASA_hphobic	InterfaceAnalzyer
dSASA_int	InterfaceAnalzyer
dSASA_polar	InterfaceAnalzyer
delta_unsatHbonds	InterfaceAnalzyer
hbond_E_fraction	InterfaceAnalzyer
hbonds_int	InterfaceAnalzyer
nres_int	InterfaceAnalzyer
per_residue_energy_int	InterfaceAnalzyer
side1_normalized	InterfaceAnalzyer
side1_score	InterfaceAnalzyer
side2_normalized	InterfaceAnalzyer
side2_score	InterfaceAnalzyer

Table B.5: **Binding score terms.** The Rosetta binding score terms, calculated on the GB1-IgG complex structure and used in addition to the standard score terms to train METL-L-Bind.

4 DISCUSSION

4.1 Contributions

Protein engineering enables us to harness the immense potential of proteins for human benefit, with wide-ranging applications from creating life-saving therapeutics to enhancing the sustainability of industrial processes. An inherent challenge in protein engineering is the vastness and complexity of the protein sequence space. It is not feasible to experimentally characterize all possible protein variants, and it is hard to know how a particular amino acid substitution will affect function without testing it.

Computational methods can assist with protein engineering by extracting insights from data and prioritizing variants for experimental characterization. With new sources of data, advancements in machine learning, and ever increasing computational power, computational methods promise to have a transformative impact on protein engineering. Yet, computational methods are not without their own challenges, and there remain many questions in how to accurately and efficiently model protein sequence-function relationships.

In this dissertation, I explored the complex problem of learning protein sequence-function relationships for protein engineering. Specifically, my research focused on training neural networks to predict protein variant functional scores from experimental data. In exploring this topic, I conducted original research yielding theoretical, methodological, and practical contributions. Chapter 2 presented a framework for training neural networks on deep mutational scanning data and tested several

network architectures, including graph convolutional networks incorporating protein structure. Chapter 3 introduced a method for improving performance with limited experimental datasets by pretraining transformer-based neural networks on molecular simulations. The following sections highlight some of the substantive contributions from these chapters.

Chapter 2: Neural Nets for Deep Mutational Scanning Data

- Demonstrated that neural networks can effectively learn the protein sequence-function mapping when given enough data
- Compared linear regression, fully connected networks, sequence convolutional networks, and graph convolutional networks, showing the benefits of using nonlinear, parameter-sharing networks
- Developed a framework for integrating protein 3D structure into a graph convolutional network via a protein structure graph
- Demonstrated that protein structure did not notably improve performance in this scenario, potentially due to factors like the lack of sequence diversity in the data or the specific network architecture used
- Showed the benefits of training on experimental data compared to evolutionary-based zero shot methods and physics-based scores
- Showed the neural network latent space clusters variants based on different molecular mechanisms of function, suggesting that the networks are learning

biologically meaningful information

- Simulated datasets of varying quality, showing the detrimental effect that poor quality datasets can have on supervised learning methods
- Published an accessible framework for training neural networks on deep mutational scanning: `https://github.com/gitter-lab/nn4dms`

Chapter 3: Mutational Effect Transfer Learning

- Introduced Mutational Effect Transfer Learning (METL), a pretraining framework based on biophysical simulations
- Developed two approaches within the METL framework: METL-Local and METL-Global, targeting local and global representations of the sequence space, respectively
- Showed that pretrained models can predict Rosetta energy terms accurately, although with overfitting on the global dataset
- Explored dynamics between experimental and simulated data in the METL framework, showing how simulated data can compensate for lack of experimental data
- Explored the importance of evolutionary conservation signals versus stability signals for learning protein sequence-function relationships from experimental data

- Developed a protein 3D structure-based relative position embedding (RPE) for transformer models and demonstrated it performs better than a 1D sequence-based RPE
- Developed frameworks for running molecular simulations and training METL, to be released publicly in the near future
- Pretrained models on molecular simulations, to be released publicly in the near future

Implications

The research presented in this dissertation contributes to the ongoing innovation in the field of protein engineering and bridges methodological advancements with practical outcomes. While Chapters 2 and 3 present methodological improvements, they also describe practical efforts to engineer protein variants. In Chapter 2, we used our trained models to engineer a GB1 variant that binds to IgG with substantially higher affinity than wild-type, demonstrating the ability of the models to generalize beyond the training data. In Chapter 3, we engineered avGFP variants using models trained on just 64 examples. The majority of engineered variants demonstrated some degree of fluorescence, showing the potential of our transfer learning approach to perform protein engineering with small labeled datasets. Beyond our practical engineering efforts, I created accessible and open code frameworks to enable others to reproduce our work and apply our methods to their own data (Wang and Gamazon, 2022). These contributions represent progress toward a

future where protein engineering is both more effective and accessible.

Limitations

It is important to address limitations that are inherent in any scientific endeavor. The field of computational methods for protein engineering is rapidly evolving. Due to the consistent influx of new data and new methods, there's a risk that findings can become outdated quickly. Further, with the volume of publications, related work is not always immediately evident, presenting the risk of potential oversights. Constraints on computational resources and time can limit the depth of hyperparameter sweeps, the number of baselines and datasets that can be tested, and other important research attributes. Additionally, this research is highly interdisciplinary, and it is possible an expert in a particular field might find additional oversights.

4.2 Future Work

There are many opportunities to further explore computational methods for protein engineering, both in terms of improving the methods presented in this dissertation and for the field in general. This section touches on several potential research directions that may yield interesting or useful results.

Improving METL

Chapter 3 introduced METL and laid the groundwork for transfer learning from biophysical simulations to improve the modeling of experimental sequence-function

data. There are several potential avenues to further expand upon METL.

Targeted Molecular Simulations

Transfer learning works best when the source task closely aligns with the target task. METL's source task of predicting Rosetta score terms primarily captures protein stability. On the other hand, experimental datasets often measure more specific functions such as binding affinity, fluorescence, enzymatic activity, and abundance. As shown in Chapter 3, protein stability can be a useful signal and related to these experimentally measured functions. However, Rosetta presents the potential to emulate target functions more closely through simulations that can capture docking and binding. Training METL with more specific, experimentally-related simulations could align the source and target tasks and potentially improve METL's performance. We explored this idea in Chapter 3 for the GB1 dataset, which measures binding affinity to IgG. We computed interface energies for the GB1-IgG complex structure and used those additional energies as pretraining tasks for METL-L. The resulting METL-L-Bind model outperformed the standard METL-L across small experimental training set sizes. This result is promising and suggests additional exploration in the area of tailored simulations is warranted. Furthermore, the general simulations described in Chapter 3 were optimized for speed over quality to achieve the goal of simulating millions of protein variants. Rosetta is capable of more detailed general stability simulations that may provide a stronger or cleaner protein stability signal.

Combining Global, Local, and Targeted Simulation Data

Currently, METL-Global and METL-Local are treated as separate methods, but both approaches offer useful and potentially distinct information to downstream protein function prediction. Combining global stability information, local stability information, and local binding information could be a promising avenue for future research. One option to combine these models is through a multi-step fine-tuning scheme where a randomly initialized model is first trained with global Rosetta data like METL-Global, fine-tuned with local Rosetta data like METL-Local, and then fine-tuned even further on Rosetta docking simulations. An advantage of this multi-step approach is that it would allow for flexibility in dataset variants and pretraining tasks because each type of data could have its own training procedure. Otherwise, a combined loss function might be needed to encompass different potential variants and tasks. Future work can explore the contributions of different types of Rosetta pretraining data to downstream tasks as well as optimal ways to combine different types of Rosetta pretraining data. Combining global stability, local stability, and local binding simulations could offer stronger inductive biases that might be helpful for low-N protein engineering and extrapolation.

METL-Global as a Protein Language Model

METL-Global can be viewed as a biophysics-inspired protein language model, trained with molecular simulation data instead of evolutionary data. However, due to problems with overfitting to training set PDBs, METL-Global is unable to realize its full potential as a protein language model. Improving METL-Global to

generalize better to out-of-distribution PDBs would make the model more capable and generally applicable. Once improved, METL-Global could be evaluated on more tasks besides protein variant function prediction, such as contact prediction or structure prediction, similar to other protein language models. METL-Global is closely related to existing protein stability predictors (Capriotti et al., 2005; Folkman et al., 2016; Cao et al., 2019; Chen et al., 2020; Li et al., 2020; Wang et al., 2022b; Blaabjerg et al., 2023; Hummer et al., 2023; Zhou et al., 2023; Dieckhaus et al., 2023; Boyer et al., 2023; Sun et al., 2023), and it would be important to examine that set of related work closely to determine how METL-Global compares and whether there exists related work that has already solved some of the challenges encountered by METL-Global.

Revisiting the METL Architecture

One aspect of the METL architecture worth exploring is the global average pooling layer. Prior to the final fully connected layers of the network, the global average pooling layer takes the mean of the per-residue representations to create a single sequence-level representation. This operation helps METL-Global handle variable length sequences, but it is inherently lossy. The METL-Local architecture also contains the global average pooling layer, but because METL-Local sequences are all the same length, this pooling operation is not necessarily required. There is potential that METL-Local could benefit from maintaining separate per-residue encodings all the way through to the end of the network, preserving the very important residue-level signal. For METL-Global, there are also alternatives to

global average pooling worth exploring. Architectures based on BERT (Devlin et al., 2019) use a special input token known as the CLS (classification) token, placed at the beginning of the sequence. This token enables the network to learn an aggregated sequence-level representation, which is similar to a weighted average of per-token (in our case, per-residue) encodings, rather than a simple global mean. METL-Global could adopt this architectural feature, although it is not necessarily guaranteed to improve performance, it is worth exploring.

Integrating Multiple Signals

In this dissertation, I've discussed multiple types of data, including evolutionary, experimental, simulated (stability or docking), protein sequence, and protein structure. The results from Chapter 3 suggest that evolutionary and stability information may be complementary to some degree. In other words, these different types of data may contribute distinct but useful information to a potential model. A compelling area of future work is integrating these multiple sources of information. There are various potential methodologies for combining this information. One option would be to create an ensemble of distinct models and average their predictions to produce a final output. Another option might be to train a unified model using the various types of information. Regardless of the specific approach, the ultimate objective for protein fitness predictors is to output the most accurate predictions possible, and it stands to reason that integrating multiple distinct, complementary signals would be an effective strategy.

Standardized Benchmarks

The community would benefit from a standardized, comprehensive benchmark for protein variant function prediction focused on protein engineering. Although benchmarks such as TAPE (Rao et al., 2019) and ProteinGym (Notin et al., 2022) have laid some foundation, there is room for improvement. Ideally, a future benchmark should be suitable for both zero-shot and supervised methods, thus it should include train, validation, and test splits. It should contain a diversity of proteins and functions. Additionally, it should include tasks relevant to protein engineering like position, mutation, score, and regime extrapolation. A standardized benchmark will facilitate comparisons across different publications. One important consideration in developing a standardized benchmark is overfitting. It is possible for published methods to overfit to standardized benchmarks over time, as the same datasets and splits are used for method development. One idea to mitigate this risk is to implement an automated system that evaluates models on a hidden test set.

4.3 Reflections

My graduate studies at the University of Wisconsin-Madison began in 2016, marking the start of a significant and transformative chapter filled with research at forefront of computational protein engineering. It is impractical to recount every lesson and insight from the past seven years, but reflecting on select topics may provide value for the field. This closing section offers perspectives on select topics, both about the research itself and also the process of performing that research.

Nuances of Proteins

There has been significant progress and success in using machine learning to model both images and natural language. Building on this success, researchers have applied the same methodologies to proteins. Proteins are defined by sequences of amino acids. This makes them similar to text data because amino acids, represented by text characters, come together to form proteins, just like text characters come together to form meaningful words and sentences. However, that is about where the similarity ends.

There are distinctions between proteins, images, and natural language that are important to modeling and learning protein sequence-function relationships. Proteins are three-dimensional molecules, and their function is determined in part by their interactions with the environment and other molecules. One could argue that protein sequence defines protein structure, and thus modeling directly from sequence to function integrates or removes the need to explicitly consider structure. That argument may be technically valid, but similar logic dictates that fully connected networks are universal function approximators, and thus we do not need convolutional networks or transformers, whose inductive biases more closely model images and natural language. Practically, we need to consider the nature of images and natural language to achieve breakthrough performance in those domains. Likewise, protein structure is inherent to function, and it stands to reason that incorporating structure information could improve modeling of protein sequence-function relationships.

Moreover, there are differences between protein amino acid sequences and

natural language, even when considering proteins as text strings. There are 20 commonly used amino acids, and they have well-defined physicochemical properties. That's similar to the number of characters in the English alphabet, but substantially less than the number of words in the English language. I do not know enough about linguistics and the evolution of language to say whether or not alphabet characters have semantic meaning, but if they do, then certainly that meaning is not based on well-defined physical and chemical principles. Proteins, like language, evolved naturally, but are governed by different underlying structures and rules. There are almost certainly modeling implications, although perhaps they are not fully clear yet. For instance, with transformer-based neural networks applied to proteins, is it necessary to learn amino acid embeddings from the data? Or, can we represent amino acids by their physicochemical properties and restructure the network architecture to encode sequence context separately from the embedding?

An underexplored area of computational protein engineering concerns the potential of non-canonical amino acids. Non-canonical amino acids are amino acids that are not part of the standard 20 encoded by the genetics of most living organisms. Non-canonical amino acids can occur in certain organisms or be synthesized in laboratories, and there are potentially thousands of these non-canonical amino acids (Narancic et al., 2019; Zitti and Jones, 2023). Non-canonical amino acids have different physicochemical properties and thus have potential to affect protein structure and function in different ways than the standard 20. This presents both an opportunity and a challenge for protein engineering. Evolutionary data, naturally, only contains information about the standard 20 amino acids. Additionally, based on

my experience, most high-throughput experimental datasets only contain data with the standard 20. As a consequence, computational methods to model non-canonical amino acids may need to rely on low-throughput experiments and biophysics-based modeling, a topic I explored in this dissertation. Overall, considering these protein-specific nuances could be important to see the same breakthrough in proteins that we have seen in other fields.

Research in a Rapidly Advancing Field

The field of computational protein engineering has exploded in popularity and is advancing rapidly. The increased interest is driven by unrealized potential, and it has attracted researchers in academia, innovative startups, and large commercial organizations. This environment brings excitement and the potential to make a true impact, but it also brings unique challenges.

With explosive growth comes a large influx of relevant publications. It can be difficult for an individual researcher to keep up with ongoing research and its nuances and implications. Furthermore, with a growing number of different methods to predict protein variant fitness, it can be time-prohibitive to run all of them as baselines for comparative analysis. In our case, we planned to run EVE as a baseline in Chapter 3, but we did not accomplish that task until we were nearing the end of the research project. The EVE results were better than expected, changing the narrative and forcing us to explore the results in more detail. These types of disruptions may be inevitable in a dynamic and rapidly evolving field, and they require continual adaptation from researchers. The exciting aspect is that

every disruption and unexpected result is an opportunity for novel insights and advancement.

The commercial sector brings seemingly unlimited resources in terms of computational power and scientific expertise and talent, enabling companies to operate on a scale that is not easily available to graduate students or small academic teams. A straightforward example is training very large neural networks, which requires many costly, high-end GPUs. Even if a graduate student can access the required computational resources, they may be working alone or with a small group. Large corporations have access to teams of researchers and software engineers that allow them to operate faster and explore more potential avenues of research. This is not to say graduate students should give up hope. On the contrary, graduate students have the freedom to dive deep and focus on problems that might not immediately appeal to commercial entities. While graduate students may not be able to compete effectively on scale, there are opportunities for academic researchers to contribute valuable insights to the field by exploring novel ideas and methodological improvements.

Quality Science

Every scientist aspires to produce high-quality work. Over the last seven years of research, my perspective on what defines "quality" has deepened and matured. At its core, I believe performing quality science requires sincerity of intention, a trait I assume is inherent to most dedicated researchers.

We all strive to do work that is free from errors, deficiencies, and limitations.

However, even the most experienced researchers can make mistakes, whether they are minor oversights or fundamental misjudgements. Transparency and openness allow us to make meaningful contributions to the community, despite the inevitability of defects and limitations. By being candid about our methods, rationale, and known limitations, we allow others to evaluate and potentially improve upon our work. Being open reinforces the value of our research.

Beyond being transparent, I believe quality work requires researchers to present a complete picture with appropriate context. It is essential for researchers to understand the nuances of their own work and the broader implications in the field. Without this understanding, it is easy to accidentally mislead others. Achieving a nuanced understanding takes time. Throughout the course of my own work, I ran experiments and explored paths that never made it into a manuscript or this dissertation. However, that work was important to deepen my own understanding and ultimately uphold my responsibility to the community.

Reproducibility is recognized as a core tenet of science, and it demands more of researchers than is immediately evident, especially in the computational field. It goes beyond describing a method in detail. Quality research is also about actively enabling others to reproduce your work by providing the necessary code and tools. This includes taking the time to write clean code and documentation and publishing it in a way that is easy for others to access and utilize. Further, it requires researchers to be responsive to questions and maintain open communication with the community.

4.4 Conclusion

As I reflect on this dissertation and the years of research leading to it, I am drawn to the future that lies ahead. Protein engineering holds immense potential to make a real-world impact, and our accomplishments thus far are just the beginning. My sincere hope for the future is that the field continues with the current momentum, and that we are able to fully realize the potential of computational methods in protein engineering.

REFERENCES

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from `tensorflow.org`.

Adzhubei, Ivan A., Steffen Schmidt, Leonid Peshkin, Vasily E. Ramensky, Anna Gerasimova, Peer Bork, Alexey S. Kondrashov, and Shamil R. Sunyaev. 2010. A method and server for predicting damaging missense mutations. *Nature Methods* 7(4):248–249. DOI: `10.1038/nmeth0410-248`.

Aghazadeh, Amirali, Hunter Nisonoff, Orhan Ocal, David H. Brookes, Yijie Huang, O. Ozan Koyluoglu, Jennifer Listgarten, and Kannan Ramchandran. 2021. Epistatic Net allows the sparse spectral regularization of deep neural networks for inferring fitness functions. *Nature Communications* 12(1):5225. DOI: `10.1038/s41467-021-25371-3`.

Aghazadeh, Amirali, Hunter Nisonoff, Orhan Ocal, Yijie Huang, O. Ozan Koyluoglu, Jennifer Listgarten, and Kannan Ramchandran. 2020. Sparse Epistatic Regularization of Deep Neural Networks for Inferring Fitness Functions. *bioRxiv*. DOI: `10.1101/2020.11.24.396994`.

Ahmad, Walid, Elana Simon, Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2022. ChemBERTa-2: Towards Chemical Foundation Models. *arXiv:2209.01712*. DOI: `10.48550/arXiv.2209.01712`.

Alcántara, Andrés R., Pablo Domínguez de María, Jennifer A. Littlechild, Martin Schürmann, Roger A. Sheldon, and Roland Wohlgemuth. 2022. Biocatalysis as Key to Sustainable Industrial Chemistry. *ChemSusChem* 15(9):e202102709. DOI: 10.1002/cssc.202102709.

Alford, Rebecca F., Andrew Leaver-Fay, Jeliazko R. Jeliazkov, Matthew J. O'Meara, Frank P. DiMaio, Hahnbeom Park, Maxim V. Shapovalov, P. Douglas Renfrew, Vikram K. Mulligan, Kalli Kappel, Jason W. Labonte, Michael S. Pacella, Richard Bonneau, Philip Bradley, Roland L. Jr. Dunbrack, Rhiju Das, David Baker, Brian Kuhlman, Tanja Kortemme, and Jeffrey J. Gray. 2017. The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *Journal of Chemical Theory and Computation* 13(6):3031–3048. DOI: 10.1021/acs.jctc.7b00125.

Alley, Ethan C., Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. 2019. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods* 16(12):1315–1322. DOI: 10.1038/s41592-019-0598-1.

Alzubaidi, Laith, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. 2021. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data* 8(1):53. DOI: 10.1186/s40537-021-00444-8.

Ancona, Marco, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. *arXiv*. DOI: 10.48550/arXiv.1711.06104.

Angermueller, Christof, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy Colwell, and D Sculley. 2020. Population-based black-box optimization for biological sequence design. In *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *ICML'20*, 324–334. JMLR.org.

Asgari, Ehsaneddin, and Mohammad R. K. Mofrad. 2015. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLOS ONE* 10(11):e0141287. DOI: 10.1371/journal.pone.0141287.

Ashburner, Michael, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. 2000. Gene Ontology: Tool for the unification of biology. *Nature Genetics* 25(1):25–29. DOI: 10.1038/75556.

Barrett, Tanya, Stephen E. Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F. Kim, Maxim Tomashevsky, Kimberly A. Marshall, Katherine H. Phillippy, Patti M. Sherman, Michelle Holko, Andrey Yefanov, Hyeseung Lee, Naigong Zhang, Cynthia L. Robertson, Nadezhda Serova, Sean Davis, and Alexandra Soboleva. 2013. NCBI GEO: Archive for functional genomics data sets—update. *Nucleic Acids Research* 41(D1):D991–D995. DOI: 10.1093/nar/gks1193.

Bekker, Jessa, and Jesse Davis. 2020. Learning from positive and unlabeled data: A survey. *Machine Learning* 109(4):719–760. DOI: 10.1007/s10994-020-05877-5.

Bepler, Tristan, and Bonnie Berger. 2021. Learning the protein language: Evolution, structure, and function. *Cell Systems* 12(6):654–669.e3. DOI: 10.1016/j.cels.2021.05.017.

Berman, Helen M., John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. 2000. The Protein Data Bank. *Nucleic Acids Research* 28(1):235–242. DOI: 10.1093/nar/28.1.235.

Biswas, Surojit, Grigory Khimulya, Ethan C. Alley, Kevin M. Esvelt, and George M. Church. 2021. Low-N protein engineering with data-efficient deep learning. *Nature Methods* 18(4):389–396. DOI: 10.1038/s41592-021-01100-y.

Biswas, Surojit, Gleb Kuznetsov, Pierce J. Ogden, Nicholas J. Conway, Ryan P. Adams, and George M. Church. 2018. Toward machine-guided design of proteins. *bioRxiv*. DOI: 10.1101/337154.

Blaabjerg, Lasse M, Maher M Kassem, Lydia L Good, Nicolas Jonsson, Matteo Cagiada, Kristoffer E Johansson, Wouter Boomsma, Amelie Stein, and Kresten Lindorff-Larsen. 2023. Rapid protein stability prediction using deep learning representations. *eLife* 12:e82593. DOI: 10.7554/eLife.82593.

Boël, Grégory, Reka Letso, Helen Neely, W. Nicholson Price, Kam-Ho Wong, Min Su, Jon D. Luff, Mayank Valecha, John K. Everett, Thomas B. Acton, Rong Xiao, Gaetano T. Montelione, Daniel P. Aalberts, and John F. Hunt. 2016. Codon influence on protein expression in E. coli correlates with mRNA levels. *Nature* 529(7586): 358–363. DOI: 10.1038/nature16509.

Bordin, Nicola, Christian Dallago, Michael Heinzinger, Stephanie Kim, Maria Littmann, Clemens Rauer, Martin Steinegger, Burkhard Rost, and Christine Orengo. 2023. Novel machine learning approaches revolutionize protein knowledge. *Trends in Biochemical Sciences* 48(4):345–359. DOI: 10.1016/j.tibs.2022.11.001.

Boyer, Sebastien, Sam Money-Kyrle, and Oliver Bent. 2023. Predicting protein stability changes under multiple amino acid substitutions using equivariant graph neural networks. *arXiv:2305.19801* [*q-bio.BM*]. DOI: 10.48550/arXiv.2305.19801.

Boël, Grégory, Reka Letso, Helen Neely, W. Nicholson Price, Kam Ho Wong, Min Su, Jon D. Luff, Mayank Valecha, John K. Everett, Thomas B. Acton, Rong Xiao, Gaetano T. Montelione, Daniel P. Aalberts, and John F. Hunt. 2016. Codon influence on protein expression in E. coli correlates with mRNA levels. *Nature 2016 529:7586* 529:358–363. DOI: 10.1038/nature16509.

Brookes, David, Hahnbeom Park, and Jennifer Listgarten. 2019. Conditioning by adaptive sampling for robust design. In *Proceedings of the 36th International Conference on Machine Learning*, 773–782. PMLR.

Bryant, Drew H., Ali Bashir, Sam Sinai, Nina K. Jain, Pierce J. Ogden, Patrick F. Riley, George M. Church, Lucy J. Colwell, and Eric D. Kelsic. 2021. Deep diversification of an AAV capsid protein by machine learning. *Nature Biotechnology*. DOI: 10.1038/s41587-020-00793-4.

Cao, Huali, Jingxue Wang, Liping He, Yifei Qi, and John Z. Zhang. 2019. DeepDDG: Predicting the Stability Change of Protein Point Mutations Using Neural Networks. *Journal of Chemical Information and Modeling* 59(4):1508–1514. DOI: 10.1021/acs.jcim.8b00697.

Capriotti, Emidio, Piero Fariselli, and Rita Casadio. 2005. I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Research* 33(suppl_2):W306–W310. DOI: 10.1093/nar/gki375.

Carter, Paul J., and Arvind Rajpal. 2022. Designing antibodies as therapeutics. *Cell* 185(15):2789–2805. DOI: 10.1016/j.cell.2022.05.029.

Center for High Throughput Computing. 2006. Center for High Throughput Computing. DOI: 10.21231/GNT1-HW21.

Chandra, Abel, Laura Tünnermann, Tommy Löfstedt, and Regina Gratz. 2023. Transformer-based deep learning for predicting protein properties in the life sciences. *eLife* 12:e82819. DOI: 10.7554/eLife.82819.

Chen, Bo, Xingyi Cheng, Yangli-ao Geng, Shen Li, Xin Zeng, Boyan Wang, Jing Gong, Chiming Liu, Aohan Zeng, Yuxiao Dong, Jie Tang, and Le Song. 2023. xTrimoPGLM: Unified 100B-Scale Pre-trained Transformer for Deciphering the Language of Protein. *bioRxiv* 2023.07.05.547496. DOI: 10.1101/2023.07.05.547496.

Chen, Yuting, Haoyu Lu, Ning Zhang, Zefeng Zhu, Shuqin Wang, and Minghui Li. 2020. PremPS: Predicting the impact of missense mutations on protein stability. *PLOS Computational Biology* 16(12):e1008543. DOI: 10.1371/journal.pcbi.1008543.

Ching, Travers, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, Alexandr A. Kalinin, Brian T. Do, Gregory P. Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M. Hoffman, Wei Xie, Gail L. Rosen, Benjamin J. Lengerich, Johnny Israeli, Jack Lanchantin, Stephen Woloszynek, Anne E. Carpenter, Avanti Shrikumar, Jinbo Xu, Evan M. Cofer, Christopher A. Lavender, Srinivas C. Turaga, Amr M. Alexandari, Zhiyong Lu, David J. Harris, Dave DeCaprio, Yanjun Qi, Anshul Kundaje, Yifan Peng, Laura K. Wiley, Marwin H. S. Segler, Simina M. Boca, S. Joshua Swamidass, Austin Huang, Anthony Gitter, and Casey S. Greene. 2018. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface* 15(141):20170387. DOI: 10.1098/rsif.2017.0387.

Cranmer, Kyle, Johann Brehmer, and Gilles Louppe. 2020. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences* 117(48): 30055–30062. DOI: 10.1073/pnas.1912789117.

Dallago, Christian, Jody Mou, Kadina E. Johnston, Bruce J. Wittmann, Nicholas Bhattacharya, Samuel Goldman, Ali Madani, and Kevin K. Yang. 2022. FLIP: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv* 2021.11.09.467890. DOI: 10.1101/2021.11.09.467890.

Deckers, Jeroen, Tom Anbergen, Ayla M. Hokke, Anne de Dreu, David P. Schrijver, Koen de Bruin, Yohana C. Toner, Thijs J. Beldman, Jamie B. Spangler, Tom F. A. de Greef, Francesca Grisoni, Roy van der Meel, Leo A. B. Joosten, Maarten Merkx, Mihai G. Netea, and Willem J. M. Mulder. 2023. Engineering cytokine therapeutics. *Nature Reviews Bioengineering* 1–18. DOI: 10.1038/s44222-023-00030-y.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (*Long and Short Papers*), ed. Jill Burstein, Christy Doran, and Thamar Solorio, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics. DOI: 10.18653/v1/N19-1423.

Dieckhaus, Henry, Michael Brocidiacono, Nicholas Randolph, and Brian Kuhlman. 2023. Transfer learning to leverage larger datasets for improved prediction of protein stability changes. *bioRxiv* 2023.07.27.550881. DOI: 10.1101/2023.07.27.550881.

Ding, Wenze, Kenta Nakai, and Haipeng Gong. 2022. Protein design via deep learning. *Briefings in Bioinformatics* 23(3):bbac102. DOI: 10.1093/bib/bbac102.

Eastman, Peter, Pavan Kumar Behara, David L. Dotson, Raimondas Galvelis, John E. Herr, Josh T. Horton, Yuezhi Mao, John D. Chodera, Benjamin P. Pritchard, Yuanqing Wang, Gianni De Fabritiis, and Thomas E. Markland. 2023a. SPICE, A Dataset of Drug-like Molecules and Peptides for Training Machine Learning Potentials. *Scientific Data* 10(1):11. DOI: 10.1038/s41597-022-01882-6.

Eastman, Peter, Raimondas Galvelis, Raúl P. Peláez, Charlles R. A. Abreu, Stephen E. Farr, Emilio Gallicchio, Anton Gorenko, Michael M. Henry, Frank Hu, Jing Huang, Andreas Krämer, Julien Michel, Joshua A. Mitchell, Vijay S. Pande, João PGLM Rodrigues, Jaime Rodriguez-Guerra, Andrew C. Simmonett, Sukrit Singh, Jason Swails, Philip Turner, Yuanqing Wang, Ivy Zhang, John D. Chodera, Gianni De Fabritiis, and Thomas E. Markland. 2023b. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *arXiv:2310.03121*. DOI: 10.48550/arXiv.2310.03121.

Eddy, Sean R. 2011. Accelerated Profile HMM Searches. *PLOS Computational Biology* 7(10):e1002195. DOI: 10.1371/journal.pcbi.1002195.

Elnaggar, Ahmed, Hazem Essam, Wafaa Salah-Eldin, Walid Moustafa, Mohamed Elkerdawy, Charlotte Rochereau, and Burkhard Rost. 2023. Ankh: Optimized protein language model unlocks general-purpose modelling. *arXiv: 2301.06568* [*cs.LG*]. DOI: 10.48550/arXiv.2301.06568.

Elnaggar, Ahmed, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. 2022. ProtTrans: Toward

Understanding the Language of Life Through Self-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(10):7112–7127. DOI: 10.1109/TPAMI.2021.3095381.

Engelen, Stefan, Ladislas A. Trojan, Sophie Sacquin-Mora, Richard Lavery, and Alessandra Carbone. 2009. Joint Evolutionary Trees: A Large-Scale Method To Predict Protein Interfaces Based on Sequence Sampling. *PLOS Computational Biology* 5(1):e1000267. DOI: 10.1371/journal.pcbi.1000267.

Esposito, Daniel, Jochen Weile, Jay Shendure, Lea M. Starita, Anthony T. Papenfuss, Frederick P. Roth, Douglas M. Fowler, and Alan F. Rubin. 2019. MaveDB: An open-source platform to distribute and interpret data from multiplexed assays of variant effect. *Genome Biology* 20(1):223. DOI: 10.1186/s13059-019-1845-6.

Fannjiang, Clara, and Jennifer Listgarten. 2020. Autofocused oracles for model-based design. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 12945–12956. NIPS'20, Red Hook, NY, USA: Curran Associates Inc.

Faure, Andre J., Júlia Domingo, Jörn M. Schmiedel, Cristina Hidalgo-Carcedo, Guillaume Diss, and Ben Lehner. 2022. Mapping the energetic and allosteric landscapes of protein binding domains. *Nature* 604(7904):175–183. DOI: 10.1038/s41586-022-04586-4.

Faure, Andre J., Jörn M. Schmiedel, Pablo Baeza-Centurion, and Ben Lehner. 2020. DiMSum: An error model and pipeline for analyzing deep mutational scanning data and diagnosing common experimental pathologies. *Genome Biology* 21(1):207. DOI: 10.1186/s13059-020-02091-3.

Folkman, Lukas, Bela Stantic, Abdul Sattar, and Yaoqi Zhou. 2016. EASE-MM: Sequence-Based Prediction of Mutation-Induced Stability Changes with Feature-Based Multiple Models. *Journal of Molecular Biology* 428(6):1394–1405. DOI: 10.1016/j.jmb.2016.01.012.

Fout, Alex, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems 30*, ed. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 6530–6539. Curran Associates, Inc.

Fowler, Douglas M., and Stanley Fields. 2014. Deep mutational scanning: A new style of protein science. *Nature Methods* 11(8):801–807. DOI: 10.1038/nmeth.3027.

Fox, Richard J, S Christopher Davis, Emily C Mundorff, Lisa M Newman, Vesna Gavrilovic, Steven K Ma, Loleta M Chung, Charlene Ching, Sarena Tam, Sheela Muley, John Grate, John Gruber, John C Whitman, Roger A Sheldon, and Gjalt W Huisman. 2007. Improving catalytic function by ProSAR-driven enzyme evolution. *Nature Biotechnology* 25(3):338–344. DOI: 10.1038/nbt1286.

Frazer, Jonathan, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K. Min, Kelly Brock, Yarin Gal, and Debora S. Marks. 2021. Disease variant prediction with deep generative models of evolutionary data. *Nature* 599(7883):91–95. DOI: 10.1038/s41586-021-04043-8.

Gelman, Sam, Sarah A. Fahlberg, Pete Heinzelman, Philip A. Romero, and Anthony Gitter. 2021. Neural networks to learn protein sequence–function relationships from deep mutational scanning data. *Proceedings of the National Academy of Sciences* 118(48):e2104878118. DOI: 10.1073/pnas.2104878118.

Gerasimavicius, Lukas, Benjamin J. Livesey, and Joseph A. Marsh. 2023. Correspondence between functional scores from deep mutational scans and predicted effects on protein stability. *Protein Science* 32(7):e4688. DOI: 10.1002/pro.4688.

Gligorijevic, Vladimir, P. Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C. Taylor, Ian M. Fisk, Hera Vlamakis, Ramnik J. Xavier, Rob Knight, Kyunghyun Cho, and Richard Bonneau. 2020. Structure-Based Protein Function Prediction using Graph Convolutional Networks. *bioRxiv*. DOI: 10.1101/786236.

Gonzalez, Courtney E., and Marc Ostermeier. 2019. Pervasive Pairwise Intragenic Epistasis among Sequential Mutations in TEM-1 β-Lactamase. *Journal of Molecular Biology* 431(10):1981–1992. DOI: 10.1016/j.jmb.2019.03.020.

Gray, Vanessa E., Ronald J. Hause, Jens Luebeck, Jay Shendure, and Douglas M. Fowler. 2018. Quantitative Missense Variant Effect Prediction Using Large-Scale Mutagenesis Data. *Cell Systems* 6(1):116–124.e3. DOI: 10.1016/j.cels.2017.11.003.

Groth, Peter Mørch, Richard Michael, Jesper Salomon, Pengfei Tian, and Wouter Boomsma. 2023. FLOP: Tasks for Fitness Landscapes Of Protein wildtypes. *bioRxiv*. DOI: 10.1101/2023.06.21.545880.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, ed. Gaël Varoquaux, Travis Vaught, and Jarrod Millman, 11–15. Pasadena, CA USA.

Han, Kai, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. 2023. A Survey on Vision Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45(1):87–110. DOI: 10.1109/TPAMI.2022.3152247.

Harmalkar, Ameya, Roshan Rao, Yuxuan Richard Xie, Jonas Honer, Wibke Deisting, Jonas Anlahr, Anja Hoenig, Julia Czwikla, Eva Sienz-Widmann, Doris Rau, Austin J. Rice, Timothy P. Riley, Danqing Li, Hannah B. Catterall, Christine E. Tinberg, Jeffrey J. Gray, and Kathy Y. Wei. 2023. Toward generalizable prediction of antibody thermostability using machine learning on sequence and structure features. *mAbs* 15(1):2163584. DOI: 10.1080/19420862.2022.2163584.

Hawkins-Hooker, Alex, Florence Depardieu, Sebastien Baur, Guillaume Couairon, Arthur Chen, and David Bikard. 2020. Generating functional protein variants with variational autoencoders. *bioRxiv*. DOI: 10.1101/2020.04.07.029264.

Hecht, Maximilian, Yana Bromberg, and Burkhard Rost. 2015. Better prediction of functional effects for sequence variants. *BMC Genomics* 16(8):S1. DOI: 10.1186/1471-2164-16-S8-S1.

Hesslow, Daniel, Niccoló Zanichelli, Pascal Notin, Iacopo Poli, and Debora Marks. 2022. RITA: A Study on Scaling Up Generative Protein Sequence Models. *arXiv*. DOI: 10.48550/arXiv.2205.05789.

Høie, Magnus Haraldson, Matteo Cagiada, Anders Haagen Beck Frederiksen, Amelie Stein, and Kresten Lindorff-Larsen. 2022. Predicting and interpreting large-scale mutagenesis data using analyses of protein stability and conservation. *Cell Reports* 38(2). DOI: 10.1016/j.celrep.2021.110207.

Hollingsworth, Scott A., and Ron O. Dror. 2018. Molecular Dynamics Simulation for All. *Neuron* 99(6):1129–1143. DOI: 10.1016/j.neuron.2018.08.011.

Hopf, Thomas A, Anna G Green, Benjamin Schubert, Sophia Mersmann, Charlotta P I Schärfe, John B Ingraham, Agnes Toth-Petroczy, Kelly Brock, Adam J Riesselman, Perry Palmedo, Chan Kang, Robert Sheridan, Eli J Draizen, Christian Dallago, Chris Sander, and Debora S Marks. 2019. The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* 35(9):1582–1584. DOI: 10.1093/bioinformatics/bty862.

Hopf, Thomas A., John B. Ingraham, Frank J. Poelwijk, Charlotta P. I. Schärfe, Michael Springer, Chris Sander, and Debora S. Marks. 2017. Mutation effects predicted from sequence co-variation. *Nature Biotechnology* 35(2):128–135. DOI: 10.1038/nbt.3769.

Hsu, Chloe, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. 2022. Learning protein fitness models from evolutionary and assay-labeled data. *Nature Biotechnology* 40(7):1114–1122. DOI: 10.1038/s41587-021-01146-5.

Huang, Po-Ssu, Scott E. Boyken, and David Baker. 2016. The coming of age of de novo protein design. *Nature* 537(7620):320–327. DOI: 10.1038/nature19946.

Hummer, Alissa M., Constantin Schneider, Lewis Chinery, and Charlotte M. Deane. 2023. Investigating the Volume and Diversity of Data Needed for Generalizable Antibody-Antigen ΔΔG Prediction. *bioRxiv* 2023.05.17.541222. DOI: 10.1101/2023.05.17.541222.

Jemli, Sonia, Dorra Ayadi-Zouari, Hajer Ben Hlima, and Samir Bejar. 2016. Biocatalysts: Application and engineering for industrial purposes. *Critical Reviews in Biotechnology* 36(2):246–258. DOI: 10.3109/07388551.2014.950550.

Jha, Ramesh K., Tiziano Gaiotto, Andrew R.M. Bradbury, and Charlie E.M. Strauss. 2014. An improved Protein G with higher affinity for human/rabbit IgG Fc domains exploiting a computationally designed polar network. *Protein Engineering, Design and Selection* 27(4):127–134. DOI: 10.1093/protein/gzu005.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596(7873):583–589. DOI: 10.1038/s41586-021-03819-2.

Kawashima, Shuichi, Piotr Pokarowski, Maria Pokarowska, Andrzej Kolinski, Toshiaki Katayama, and Minoru Kanehisa. 2008. AAindex: Amino acid index database, progress report 2008. *Nucleic Acids Research* 36(Database issue):D202–205. DOI: 10.1093/nar/gkm998.

Kingma, Diederik P., and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. *arXiv*. DOI: 10.48550/arXiv.1412.6980.

Kosciolek, Tomasz, and David T. Jones. 2014. De Novo Structure Prediction of Globular Proteins Aided by Sequence Variation-Derived Contacts. *PLOS ONE* 9(3):e92197. DOI: 10.1371/journal.pone.0092197.

Kouba, Petr, Pavel Kohout, Faraneh Haddadi, Anton Bushuiev, Raman Samusevich, Jiri Sedlar, Jiri Damborsky, Tomas Pluskal, Josef Sivic, and Stanislav Mazurenko. 2023. Machine Learning-Guided Protein Engineering. *ACS Catalysis* 13(21):13863–13895. DOI: 10.1021/acscatal.3c02743.

Kryshtafovych, Andriy, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. 2019. Critical Assessment of Methods of Protein Structure Prediction (CASP) – Round XIII. *Proteins* 87(12):1011–1020. DOI: 10.1002/prot.25823.

Kuhlman, Brian, and Philip Bradley. 2019. Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology* 20(11):681–697. DOI: 10.1038/s41580-019-0163-x.

Kumar, Ananya, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution. *arXiv:2202.10054* [*cs*]. DOI: 10.48550/arXiv.2202.10054.

Laine, Elodie, Yasaman Karami, and Alessandra Carbone. 2019. GEMME: A Simple and Fast Global Epistatic Model Predicting Mutational Effects. *Molecular Biology and Evolution* 36(11):2604–2619. DOI: 10.1093/molbev/msz179.

Li, Bian, Yucheng T. Yang, John A. Capra, and Mark B. Gerstein. 2020. Predicting changes in protein thermodynamic stability upon point mutation with deep 3D convolutional neural networks. *PLOS Computational Biology* 16(11):e1008291. DOI: 10.1371/journal.pcbi.1008291.

Li, Mingchen, Liqi Kang, Yi Xiong, Yu Guang Wang, Guisheng Fan, Pan Tan, and Liang Hong. 2023. SESNet: Sequence-structure feature-integrated deep learning method for data-efficient protein engineering. *Journal of Cheminformatics* 15(1):12. DOI: 10.1186/s13321-023-00688-x.

Lin, Zeming, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* 379(6637):1123–1130. DOI: 10.1126/science.ade2574.

Linder, Johannes, and Georg Seelig. 2021. Fast activation maximization for molecular sequence design. *BMC Bioinformatics* 22(1):510. DOI: 10.1186/s12859-021-04437-5.

Lipsh-Sokolik, R., O. Khersonsky, S. P. Schröder, C. de Boer, S.-Y. Hoch, G. J. Davies, H. S. Overkleeft, and S. J. Fleishman. 2023. Combinatorial assembly and design of enzymes. *Science* 379(6628):195–201. DOI: 10.1126/science.ade9434.

Lipton, Zachary C., John Berkowitz, and Charles Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv*. DOI: 10.48550/arXiv.1506.00019.

Livesey, Benjamin J., and Joseph A. Marsh. 2022. Interpreting protein variant effects with computational predictors and deep mutational scanning. *Disease Models & Mechanisms* 15(6):dmm049510. DOI: 10.1242/dmm.049510.

Livesey, Benjamin J, and Joseph A Marsh. 2023. Updated benchmarking of variant effect predictors using deep mutational scanning. *Molecular Systems Biology* 19(8):e11474. DOI: 10.15252/msb.202211474.

Luo, Yunan, Guangde Jiang, Tianhao Yu, Yang Liu, Lam Vo, Hantian Ding, Yufeng Su, Wesley Wei Qian, Huimin Zhao, and Jian Peng. 2021. ECNet is an evolutionary context-integrated deep learning framework for protein engineering. *Nature Communications* 12(1):5743. DOI: 10.1038/s41467-021-25976-8.

Luo, Yunan, Lam Vo, Hantian Ding, Yufeng Su, Yang Liu, Wesley Wei Qian, Huimin Zhao, and Jian Peng. 2020. Evolutionary context-integrated deep sequence modeling for protein engineering. *bioRxiv*. DOI: 10.1101/2020.01.16.908509.

Madani, Ali, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. 2020. ProGen: Language Modeling for Protein Generation. *bioRxiv*. DOI: 10.1101/2020.03.07.982272.

Mater, Adam C., Mahakaran Sandhu, and Colin Jackson. 2020. The NK Landscape as a Versatile Benchmark for Machine Learning Driven Protein Engineering. *bioRxiv* 2020.09.30.319780. DOI: 10.1101/2020.09.30.319780.

McInnes, Leland, John Healy, and James Melville. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv*. DOI: 10.48550/arXiv.1802.03426.

Melamed, Daniel, David L. Young, Caitlin E. Gamble, Christina R. Miller, and Stanley Fields. 2013. Deep mutational scanning of an RRM domain of the Saccharomyces cerevisiae poly(A)-binding protein. *RNA* 19(11):1537–1551. DOI: 10.1261/rna.040709.113.

Mutalik, Vivek K, Joao C Guimaraes, Guillaume Cambray, Colin Lam, Marc Juul Christoffersen, Quynh-Anh Mai, Andrew B Tran, Morgan Paull, Jay D Keasling, Adam P Arkin, and Drew Endy. 2013. Precise and reliable gene expression via standard transcription and translation initiation elements. *Nature Methods* 10(4): 354–360. DOI: 10.1038/nmeth.2404.

Narancic, Tanja, Sarah A. Almahboub, and Kevin E. O'Connor. 2019. Unnatural amino acids: Production and biotechnological potential. *World Journal of Microbiology and Biotechnology* 35(4):67. DOI: 10.1007/s11274-019-2642-9.

Nedrud, David, Willow Coyote-Maestas, and Daniel Schmidt. 2021. A large-scale survey of pairwise epistasis reveals a mechanism for evolutionary expansion and specialization of PDZ domains. *Proteins: Structure, Function, and Bioinformatics* 89(8):899–914. DOI: 10.1002/prot.26067.

Nisonoff, Hunter, Yixin Wang, and Jennifer Listgarten. 2022. Augmenting Neural Networks with Priors on Function Values. *arXiv:2202.04798*. ArXiv:2202.04798 [cs, stat], DOI: 10.48550/arXiv.2202.04798.

Nordquist, Erik, Guohui Zhang, Shrishti Barethiya, Nathan Ji, Kelli M. White, Lu Han, Zhiguang Jia, Jingyi Shi, Jianmin Cui, and Jianhan Chen. 2023. Incorporating physics to overcome data scarcity in predictive modeling of protein function: a case study of BK channels. *bioRxiv*. DOI: 10.1101/2023.06.24.546384.

Notin, Pascal, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan Gomez, Debora S. Marks, and Yarin Gal. 2022. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. *arXiv*. DOI: 10.48550/arXiv.2205.13760.

Olson, C. Anders, Nicholas C. Wu, and Ren Sun. 2014. A Comprehensive Biophysical Description of Pairwise Epistasis throughout an Entire Protein Domain. *Current Biology* 24(22):2643–2651. DOI: 10.1016/j.cub.2014.09.072.

Omar, Sara Ibrahim, Chen Keasar, Ariel J. Ben-Sasson, and Eldad Haber. 2023. Protein Design Using Physics Informed Neural Networks. *Biomolecules* 13(3):457. DOI: 10.3390/biom13030457.

OSG. 2006. Open Science Pool. DOI: 10.21231/906P-4D78.

Otwinowski, Jakub, David M. McCandlish, and Joshua B. Plotkin. 2018. Inferring the shape of global epistasis. *Proceedings of the National Academy of Sciences* 115(32): E7550–E7558. DOI: 10.1073/pnas.1804015115.

Pan, Xingjie, and Tanja Kortemme. 2021. Recent advances in de novo protein design: Principles, methods, and applications. *Journal of Biological Chemistry* 296: 100558. DOI: 10.1016/j.jbc.2021.100558.

Paysan-Lafosse, Typhaine, Matthias Blum, Sara Chuguransky, Tiago Grego, Beatriz Lázaro Pinto, Gustavo A Salazar, Maxwell L Bileschi, Peer Bork, Alan Bridge, Lucy Colwell, Julian Gough, Daniel H Haft, Ivica Letunić, Aron Marchler-Bauer, Huaiyu Mi, Darren A Natale, Christine A Orengo, Arun P Pandurangan, Catherine Rivoire, Christian J A Sigrist, Ian Sillitoe, Narmada Thanki, Paul D Thomas, Silvio C E Tosatto, Cathy H Wu, and Alex Bateman. 2023. InterPro in 2022. *Nucleic Acids Research* 51(D1):D418–D427. DOI: 10.1093/nar/gkac993.

Pordes, Ruth, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery, Kent Blackburn, Torre Wenaus, Frank Würthwein, Ian Foster, Rob Gardner, Mike Wilde, Alan Blatecky, John McGee, and Rob Quick. 2007. The open science grid. *Journal of Physics: Conference Series* 78(1):012057. DOI: 10.1088/1742-6596/78/1/012057.

Ramírez-Palacios, Carlos, and Siewert J. Marrink. 2023. Super High-Throughput Screening of Enzyme Variants by Spectral Graph Convolutional Neural Networks. *Journal of Chemical Theory and Computation* 19(14):4668–4677. DOI: 10.1021/acs.jctc.2c01227.

Rao, Roshan, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S. Song. 2019. Evaluating protein transfer learning with TAPE. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 9689–9701. 869, Red Hook, NY, USA: Curran Associates Inc.

Rao, Roshan M., Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. 2021. MSA Transformer. In *Proceedings of the 38th International Conference on Machine Learning*, 8844–8856. PMLR.

Riesselman, Adam J., John B. Ingraham, and Debora S. Marks. 2018. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods* 15(10):816–822. DOI: 10.1038/s41592-018-0138-4.

Rives, Alexander, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. 2020. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*. DOI: 10.1101/622803.

Rives, Alexander, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. 2021. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences* 118(15):e2016239118. DOI: 10.1073/pnas.2016239118.

Romero, Philip A., and Frances H. Arnold. 2009. Exploring protein fitness landscapes by directed evolution. *Nature Reviews Molecular Cell Biology* 10(12):866–876. DOI: 10.1038/nrm2805.

Romero, Philip A., Andreas Krause, and Frances H. Arnold. 2013. Navigating the protein fitness landscape with Gaussian processes. *Proceedings of the National Academy of Sciences* 110(3):E193. DOI: 10.1073/pnas.1215251110.

Romero, Philip A., Tuan M. Tran, and Adam R. Abate. 2015. Dissecting enzyme function with microfluidic-based deep mutational scanning. *Proceedings of the National Academy of Sciences* 112(23):7159–7164. DOI: 10.1073/pnas.1422285112.

Rubin, Alan F., Hannah Gelman, Nathan Lucas, Sandra M. Bajjalieh, Anthony T. Papenfuss, Terence P. Speed, and Douglas M. Fowler. 2017. A statistical framework for analyzing deep mutational scanning data. *Genome Biology* 18(1):150. DOI: 10.1186/s13059-017-1272-5.

Saito, Yutaka, Misaki Oikawa, Hikaru Nakazawa, Teppei Niide, Tomoshi Kameda, Koji Tsuda, and Mitsuo Umetsu. 2018. Machine-Learning-Guided Mutagenesis for Directed Evolution of Fluorescent Proteins. *ACS Synthetic Biology* 7(9):2014–2022. DOI: 10.1021/acssynbio.8b00155.

Sanyal, Soumya, Ivan Anishchenko, Anirudh Dagar, David Baker, and Partha Talukdar. 2020. ProteinGCN: Protein model quality assessment using Graph Convolutional Networks. *bioRxiv*. DOI: 10.1101/2020.04.06.028266.

Sarkisyan, Karen S., Dmitry A. Bolotin, Margarita V. Meer, Dinara R. Usmanova, Alexander S. Mishin, George V. Sharonov, Dmitry N. Ivankov, Nina G. Bozhanova, Mikhail S. Baranov, Onuralp Soylemez, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor A. Kondrashov. 2016. Local fitness landscape of the green fluorescent protein. *Nature* 533(7603):397–401. DOI: 10.1038/nature17995.

Schymkowitz, Joost, Jesper Borg, Francois Stricher, Robby Nys, Frederic Rousseau, and Luis Serrano. 2005. The FoldX web server: An online force field. *Nucleic Acids Research* 33(suppl_2):W382–W388. DOI: 10.1093/nar/gki387.

Sfiligoi, Igor, Daniel C. Bradley, Burt Holzman, Parag Mhashilkar, Sanjay Padhi, and Frank Wurthwein. 2009. The Pilot Way to Grid Resources Using glideinWMS. In *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 2, 428–432. DOI: 10.1109/CSIE.2009.950.

Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 464–468. New Orleans, Louisiana: Association for Computational Linguistics. DOI: 10.18653/v1/N18-2074.

Simons, Kim T., Rich Bonneau, Ingo Ruczinski, and David Baker. 1999. Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins: Structure, Function, and Bioinformatics* 37(S3):171–176. DOI: 10.1002/(SICI)1097-0134(1999)37:3+<171::AID-PROT21>3.0.CO;2-Z.

Soneson, Charlotte, Alexandra M. Bendel, Guillaume Diss, and Michael B. Stadler. 2023. Mutscan—a flexible R package for efficient end-to-end analysis of multiplexed assays of variant effect data. *Genome Biology* 24(1):132. DOI: 10.1186/s13059-023-02967-0.

Song, Hyebin, Bennett J. Bremer, Emily C. Hinds, Garvesh Raskutti, and Philip A. Romero. 2021. Inferring Protein Sequence-Function Relationships with Large-Scale Positive-Unlabeled Learning. *Cell Systems* 12(1):92–101.e8. DOI: 10.1016/j.cels.2020.10.007.

Song, Yifan, Frank DiMaio, Ray Yu-Ruei Wang, David Kim, Chris Miles, TJ Brunette, James Thompson, and David Baker. 2013. High-Resolution Comparative Modeling with RosettaCM. *Structure* 21(10):1735–1742. DOI: 10.1016/j.str.2013.08.005.

Starita, Lea M., Jonathan N. Pruneda, Russell S. Lo, Douglas M. Fowler, Helen J. Kim, Joseph B. Hiatt, Jay Shendure, Peter S. Brzovic, Stanley Fields, and Rachel E. Klevit. 2013. Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. *Proceedings of the National Academy of Sciences* 110(14):E1263–E1272. DOI: 10.1073/pnas.1303309110.

Starr, Tyler N., and Joseph W. Thornton. 2016. Epistasis in protein evolution. *Protein Science* 25(7):1204–1218. DOI: 10.1002/pro.2897.

Strokach, Alexey, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, and Philip M. Kim. 2020. Fast and Flexible Protein Design Using Deep Graph Neural Networks. *Cell Systems* 11(4):402–411.e4. DOI: 10.1016/j.cels.2020.08.016.

Sun, Jinyuan, Tong Zhu, Yinglu Cui, and Bian Wu. 2023. Structure-based self-supervised learning enables ultrafast prediction of stability changes upon mutation at the protein universe scale. *bioRxiv* 2023.08.09.552725. DOI: 10.1101/2023.08.09.552725.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 3319–3328. ICML'17, Sydney, NSW, Australia: JMLR.org.

Suzek, Baris E., Yuqi Wang, Hongzhan Huang, Peter B. McGarvey, Cathy H. Wu, and the UniProt Consortium. 2015. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* 31(6):926–932. DOI: 10.1093/bioinformatics/btu739.

Tareen, Ammar, Mahdi Kooshkbaghi, Anna Posfai, William T. Ireland, David M. McCandlish, and Justin B. Kinney. 2022. MAVE-NN: Learning genotype-phenotype maps from multiplex assays of variant effect. *Genome Biology* 23(1):98. DOI: 10.1186/s13059-022-02661-7.

Tareen, Ammar, Anna Posfai, William T. Ireland, David M. McCandlish, and Justin B. Kinney. 2020. MAVE-NN: learning genotype-phenotype maps from multiplex assays of variant effect. *bioRxiv*. DOI: 10.1101/2020.07.14.201475.

Thain, Douglas, Todd Tannenbaum, and Miron Livny. 2005. Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience* 17(2-4):323–356. DOI: 10.1002/cpe.938.

The Gene Ontology Consortium, Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, David P Hill, Raymond Lee, Huaiyu Mi, Sierra Moxon, Christopher J Mungall, Anushya Muruganugan, Tremayne Mushayahama, Paul W Sternberg, Paul D Thomas, Kimberly Van Auken, Jolene Ramsey, Deborah A Siegele, Rex L Chisholm, Petra Fey, Maria Cristina Aspromonte, Maria Victoria Nugnes, Federica Quaglia, Silvio Tosatto, Michelle Giglio, Suvarna Nadendla, Giulia Antonazzo, Helen Attrill, Gil dos Santos, Steven Marygold, Victor Strelets, Christopher J Tabone, Jim Thurmond, Pinglei Zhou, Saadullah H Ahmed, Praoparn Asanitthong, Diana Luna Buitrago, Meltem N Erdol, Matthew C Gage, Mohamed Ali Kadhum, Kan Yan Chloe Li, Miao Long, Aleksandra Michalak, Angeline Pesala, Armalya Pritazahra, Shirin C C Saverimuttu, Renzhi Su, Kate E Thurlow, Ruth C Lovering, Colin Logie, Snezhana Oliferenko, Judith Blake, Karen Christie, Lori Corbani, Mary E Dolan, Harold J Drabkin, David P Hill, Li Ni, Dmitry Sitnikov, Cynthia Smith, Alayne Cuzick, James Seager, Laurel Cooper, Justin Elser, Pankaj Jaiswal, Parul Gupta, Pankaj Jaiswal, Sushma Naithani, Manuel Lera-Ramirez, Kim Rutherford, Valerie Wood, Jeffrey L De Pons, Melinda R Dwinell, G Thomas Hayman, Mary L Kaldunski, Anne E Kwitek, Stanley J F Laulederkind, Marek A Tutaj, Mahima Vedi, Shur-Jen Wang, Peter D'Eustachio, Lucila Aimo, Kristian Axelsen, Alan Bridge, Nevila Hyka-Nouspikel, Anne Morgat, Suzi A Aleksander, J Michael Cherry, Stacia R Engel, Kalpana Karra, Stuart R Miyasato, Robert S Nash, Marek S Skrzypek, Shuai Weng, Edith D Wong, Erika Bakker, Tanya Z Berardini, Leonore Reiser, Andrea Auchincloss, Kristian Axelsen, Ghislaine Argoud-Puy, Marie-Claude Blatter, Emmanuel Boutet, Lionel Breuza, Alan Bridge, Cristina Casals-Casas, Elisabeth Coudert, Anne Estreicher, Maria Livia Famiglietti, Marc Feuermann, Arnaud Gos, Nadine Gruaz-Gumowski, Chantal Hulo, Nevila Hyka-Nouspikel, Florence Jungo, Philippe Le Mercier,

Damien Lieberherr, Patrick Masson, Anne Morgat, Ivo Pedruzzi, Lucille Pourcel, Sylvain Poux, Catherine Rivoire, Shyamala Sundaram, Alex Bateman, Emily Bowler-Barnett, Hema Bye-A-Jee, Paul Denny, Alexandr Ignatchenko, Rizwan Ishtiaq, Antonia Lock, Yvonne Lussi, Michele Magrane, Maria J Martin, Sandra Orchard, Pedro Raposo, Elena Speretta, Nidhi Tyagi, Kate Warner, Rossana Zaru, Alexander D Diehl, Raymond Lee, Juancarlos Chan, Stavros Diamantakis, Daniela Raciti, Magdalena Zarowiecki, Malcolm Fisher, Christina James-Zorn, Virgilio Ponferrada, Aaron Zorn, Sridhar Ramachandran, Leyla Ruzicka, and Monte Westerfield. 2023. The Gene Ontology knowledgebase in 2023. *Genetics* 224(1):iyad031. DOI: 10.1093/genetics/iyad031.

The UniProt Consortium. 2023. UniProt: The Universal Protein Knowledgebase in 2023. *Nucleic Acids Research* 51(D1):D523–D531. DOI: 10.1093/nar/gkac1052.

Torrisi, Mirko, Gianluca Pollastri, and Quan Le. 2020. Deep learning methods in protein structure prediction. *Computational and Structural Biotechnology Journal* 18: 1301–1310. DOI: 10.1016/j.csbj.2019.12.011.

Varadi, Mihaly, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. 2022. AlphaFold Protein Structure Database: Massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research* 50(D1):D439–D444. DOI: 10.1093/nar/gkab1061.

Vaser, Robert, Swarnaseetha Adusumalli, Sim Ngak Leng, Mile Sikic, and Pauline C. Ng. 2016. SIFT missense predictions for genomes. *Nature Protocols* 11(1):1–9. DOI: 10.1038/nprot.2015.123.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. DOI: 10.48550/arXiv.1706.03762.

Voelz, Vincent A., Vijay S. Pande, and Gregory R. Bowman. 2023. Folding@home: Achievements from over 20 years of citizen science herald the exascale era. *Biophysical Journal* 122(14):2852–2863. DOI: 10.1016/j.bpj.2023.03.028.

Vu, Thi Thuy Duong, and Jaehee Jung. 2021. Protein function prediction with gene ontology: From traditional to deep learning models. *PeerJ* 9:e12019. DOI: 10.7717/peerj.12019.

Wang, Amy, Ava Soleimany, Alex X Lu, and Kevin Yang. 2022a. Learning from physics-based features improves protein property prediction. In *Machine Learning for Structural Biology Workshop at the 36th Conference on Neural Information Processing Systems*.

Wang, Bo, and Eric R. Gamazon. 2022. Modeling mutational effects on biochemical phenotypes using convolutional neural networks: Application to SARS-CoV-2. *iScience* 25(7). DOI: 10.1016/j.isci.2022.104500.

Wang, Connie Y., Paul M. Chang, Marie L. Ary, Benjamin D. Allen, Roberto A. Chica, Stephen L. Mayo, and Barry D. Olafson. 2018. ProtaBank: A repository for protein design and engineering data. *Protein Science : A Publication of the Protein Society* 27(6):1113–1124. DOI: 10.1002/pro.3406.

Wang, Shuyu, Hongzhou Tang, Yuliang Zhao, and Lei Zuo. 2022b. BayeStab: Predicting effects of mutations on protein stability with uncertainty quantification. *Protein Science* 31(11):e4467. DOI: 10.1002/pro.4467.

Wang, Zirui, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. 2019. Characterizing and Avoiding Negative Transfer. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11285–11294. DOI: 10.1109/CVPR.2019.01155.

Watanabe, Hideki, Chuya Yoshida, Ayako Ooishi, Yasuto Nakai, Momoko Ueda, Yutaka Isobe, and Shinya Honda. 2019. Histidine-Mediated Intramolecular Electrostatic Repulsion for Controlling pH-Dependent Protein–Protein Interaction. *ACS Chemical Biology* 14(12):2729–2736. DOI: 10.1021/acschembio.9b00652.

Watson, Joseph L., David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. 2023. De novo design of protein structure and function with RFdiffusion. *Nature* 1–12. DOI: 10.1038/s41586-023-06415-8.

Weinstein, Jonathan Yaacov, Carlos Martí-Gómez, Rosalie Lipsh-Sokolik, Shlomo Yakir Hoch, Demian Liebermann, Reinat Nevo, Haim Weissman, Ekaterina Petrovich-Kopitman, David Margulies, Dmitry Ivankov, David M. McCandlish, and Sarel J. Fleishman. 2023. Designed active-site library reveals thousands of functional GFP variants. *Nature Communications* 14(1):2890. DOI: 10.1038/s41467-023-38099-z.

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3(1):9. DOI: 10.1186/s40537-016-0043-6.

Wells, James A. 1990. Additivity of mutational effects in proteins. *Biochemistry* 29(37):8509–8517. DOI: 10.1021/bi00489a001.

Wittmann, Bruce J., Yisong Yue, and Frances H. Arnold. 2020. Machine Learning-Assisted Directed Evolution Navigates a Combinatorial Epistatic Fitness Landscape with Minimal Screening Burden. *bioRxiv*. DOI: 10.1101/2020.12.04.408955.

Wu, Zijun, and Saurabh Sinha. 2023. SPREd: A simulation-supervised neural network tool for gene regulatory network reconstruction. *bioRxiv* 2023.11.09.566399. DOI: 10.1101/2023.11.09.566399.

Wu, Zonghan, Pan, Shirui, Chen, Fengwen, Long, Guodong, Zhang, Chengqi, and Yu, Philip S. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 1–21. DOI: 10.1109/TNNLS.2020.2978386.

Xiong, Ruibin, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On Layer Normalization in the Transformer Architecture. *arXiv:2002.04745* [*cs, stat*]. DOI: 10.48550/arXiv.2002.04745.

Xu, Yuting, Deeptak Verma, Robert P. Sheridan, Andy Liaw, Junshui Ma, Nicholas M. Marshall, John McIntosh, Edward C. Sherer, Vladimir Svetnik, and Jennifer M. Johnston. 2020. Deep Dive into Machine Learning Models for Protein Engineering. *Journal of Chemical Information and Modeling* 60(6):2773–2790. DOI: 10.1021/acs.jcim.0c00073.

Yang, Kevin K., Nicolo Fusi, and Alex X. Lu. 2023. Convolutions are competitive with transformers for protein sequence pretraining. *bioRxiv*. DOI: 10.1101/2022.05.19.492714.

Yang, Kevin K., Zachary Wu, and Frances H. Arnold. 2019. Machine-learning-guided directed evolution for protein engineering. *Nature Methods* 16(8):687–694. DOI: 10.1038/s41592-019-0496-6.

Yang, Kevin K, Zachary Wu, Claire N Bedbrook, and Frances H Arnold. 2018. Learned protein embeddings for machine learning. *Bioinformatics* 34(15):2642–2648. DOI: 10.1093/bioinformatics/bty178.

Yu, Sungduk, Walter Hannah, Liran Peng, Jerry Lin, Mohamed Aziz Bhouri, Ritwik Gupta, Björn Lütjens, Justus Christopher Will, Gunnar Behrens, Julius Busecke, Nora Loose, Charles I. Stern, Tom Beucler, Bryce Harrop, Benjamin R. Hillman, Andrea Jenney, Savannah Ferretti, Nana Liu, Anima Anandkumar, Noah D. Brenowitz, Veronika Eyring, Nicholas Geneva, Pierre Gentine, Stephan Mandt, Jaideep Pathak, Akshay Subramaniam, Carl Vondrick, Rose Yu, Laure Zanna,

Tian Zheng, Ryan Abernathey, Fiaz Ahmed, David C. Bader, Pierre Baldi, Elizabeth Barnes, Christopher Bretherton, Peter Caldwell, Wayne Chuang, Yilun Han, Yu Huang, Fernando Iglesias-Suarez, Sanket Jantre, Karthik Kashinath, Marat Khairoutdinov, Thorsten Kurth, Nicholas Lutsko, Po-Lun Ma, Griffin Mooers, J. David Neelin, David Randall, Sara Shamekh, Mark A. Taylor, Nathan Urban, Janni Yuval, Guang Zhang, and Michael Pritchard. 2023. ClimSim: A large multi-scale dataset for hybrid physics-ML climate emulation. *arXiv:2306.08754*. DOI: 10.48550/arXiv.2306.08754.

Zhao, Yingwen, Jun Wang, Jian Chen, Xiangliang Zhang, Maozu Guo, and Guoxian Yu. 2020. A Literature Review of Gene Function Prediction by Modeling Gene Ontology. *Frontiers in Genetics* 11. DOI: 10.3389/fgene.2020.00400.

Zhou, Yunzhuo, Qisheng Pan, Douglas E V Pires, Carlos H M Rodrigues, and David B Ascher. 2023. DDMut: predicting effects of mutations on protein stability using deep learning. *Nucleic Acids Research* 51(W1):W122–W128. DOI: 10.1093/nar/gkad472.

Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A Comprehensive Survey on Transfer Learning. *arXiv*. DOI: 10.48550/arXiv.1911.02685.

Zitti, Athena, and Dafydd Jones. 2023. Expanding the genetic code: A non-natural amino acid story. *The Biochemist* 45(1):2–6. DOI: 10.1042/bio_2023_102.