Engineering TEV Protease Specificity: An Exploration of Machine Learning and High-Throughput Experimentation for Protein Design

by

Vikram Sundar

A.B. Mathematics, Harvard University, 2018
A.M. Physics, Harvard University, 2018
MPhil Chemistry, University of Cambridge, 2019

Submitted to the Computational and Systems Biology Program
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTATIONAL AND SYSTEMS BIOLOGY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

THESIS COMMITTEE

Thesis Supervisor

Kevin M. Esvelt
Associate Professor of Media Arts and Sciences
NEC Career Development Professor of Computer and Communications
Media Lab

Thesis Readers

Amy Keating
Jay Stein Professor of Biology
Professor of Biological Engineering
Department Head
Department of Biology

Jeff Gore
Professor of Physics
Department of Physics

Sergey Ovchinnikov
Assistant Professor of Biology
Department of Biology

Debora Marks
Professor
Department of Systems Biology
Harvard Medical School

Engineering TEV Protease Specificity: An Exploration of Machine Learning and High-Throughput Experimentation for Protein Design

by

Vikram Sundar

Submitted to the Computational and Systems Biology Program
on May 7, 2025 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTATIONAL AND SYSTEMS BIOLOGY

ABSTRACT

Engineering sequence-specific proteases would enable a wide variety of therapeutic applications in diseases ranging from cancer to Parkinson's disease. However, many previous experimental and physics-based attempts at protease engineering have failed to engineer specificity in cleaving alternative substrates, rendering them useless. In this thesis, we aim to engineer TEV (tobacco etch virus) protease, a highly sequence-specific protease, to cleave alternative substrates. We incorporate novel high-throughput assays and powerful machine learning (ML) methods for highly effective protein engineering. The first portion of this thesis focuses on generating fitness landscapes from high-throughput experiments. Most machine learning models do not account for experimental noise, harming model performance and changing model rankings in benchmarking studies. Here we develop FLIGHTED, a Bayesian method of accounting for uncertainty by generating probabilistic fitness landscapes from noisy high-throughput experiments. We demonstrate how FLIGHTED can improve model performance on two categories of experiments: single-step selection assays, such as phage display, and a novel high-throughput assay called DHARMA that ties activity to base editing. FLIGHTED can be used to generate robust, well-calibrated fitness landscapes, and when combined with DHARMA, our methods enable us to generate fitness landscapes of millions of variants. We then evaluate how to model protein fitness given a fitness dataset of millions of variants. Accounting for noise via FLIGHTED significantly improves model performance, especially of high-performing models. Data size, not model scale, is the most important factor in improving model performance. Furthermore, the choice of top model architecture matters more than the protein language model embedding. The best way to generate sufficient data scale is via error-prone PCR libraries; models trained on these landscapes achieve high accuracy. Using these methods, we successfully engineer both activity on an alternative substrate and specificity when compared to the wild-type. The ML-designed variants outperform anything found in the training set, demonstrating the value of machine learning even with experimental libraries of millions of variants. However, our results are limited to relatively close substrates. How best to improve model performance on distant substrates remains an open question.

Thesis supervisor: Kevin M. Esvelt
Title: Associate Professor of Media Arts and Sciences, and
NEC Career Development Professor
of Computer and Communications

Acknowledgments

This thesis would not have been possible without the contributions of many people. First, I'd like to thank my supervisor Kevin Esvelt and the Sculpting Evolution group for creating a collaborative and cross-disciplinary scientific environment. It is very rare for computational scientists to be able to collaborate so closely with experimentalists and use so much experimental data as a core component of a thesis; I would not have been able to even come up with the problems described here, let alone solve them, without Kevin's support. Possibly even more important than Kevin for my work was my experimental collaborator and friend Boqiang Tu, who did essentially all the experiments described in this thesis and came up with the idea behind DHARMA that made much of the thesis possible. Bo has been an outstanding colleague and collaborator in helping me solve issues and being willing to run endless experiments for the purpose of machine learning. I would also like to thank other past and present members of the lab - Emma Chory, Ceili Peng, Dana Gretton, Tom Fryer, Madeline Moore, Lenni Justen, Shay Toner, Marielle Russo, Rick Wierenga, and Jett Liu - for fascinating discussions, excellent feedback during presentations, and unwavering support throughout the PhD.

I have been honored to be financially supported throughout this PhD by the Hertz Foundation fellowship. The Hertz gave me flexibility in choosing an interdisciplinary, ambitious topic for my dissertation and access to a fantastic community of fellows pursuing equally interesting, ambitious work across the country. I would especially like to thank the many friends I made at summer workshops and at retreats over the years - Joseph Scherrer, Kettner Griswold, Nitya Mani, Hannah Lawrence, Constantine Tzouanas, Maxwell Wang, Katherine Xiang, Gita Abhiraman, Kartik Chandra, John Cherian, Alex Miller, Alexander Zlokapa, David Li, Shuvom Sadhuka, Nina Anikeeva, Liyam Chitayat, Vaibhav Mohanty, and undoubtedly many more whom I am forgetting - for countless engaging conversations about all aspects of science. I would also like to thank many of the mentors in the Hertz community, not only Megan Blewett (my assigned Hertz mentor), but also others including Ashvin Bashyam, Sam Rodriques, and Christopher Loose, for their career guidance throughout the course of my PhD. The Hertz community as a whole has been absolutely invaluable to me for giving me inspiration to be more ambitious and unconventional and exposing me to areas of science I would otherwise never have heard about.

Outside of my PhD, I have been blessed with a reasonable work-life balance filled with numerous engaging hobbies and countless social events. It is rare to be able to perform three hour-long solo piano recitals, participate fully in chamber music, and learn countless pieces at the highest levels of difficulty while pursuing a PhD, and for this I thank the Emerson/Harris program at MIT for giving graduate students the opportunity to also pursue a thorough music education. I would also like to thank my piano teacher Eileen Huang and all my chamber music coaches and partners over the years for refining my musical skill and giving me the opportunity to perform so many times, and I would like to thank everyone who attended my recitals for supporting this part of my life.

Finally, I'd like to thank the many other friends I've made for all the time and experiences we've had together and all the support they've given me throughout the PhD. I'd like to thank my fellow CSB students - especially Sam Goldman, Advait Athreya, Lindsey Guan, Allison Keys, Velina Kozareva, and Ivy Liu - for their help in navigating a particularly challenging start to the PhD during the craziness of COVID. I would also like to thank my friends from my year at Cambridge (especially the Churchill Scholars) - Aswini Krishnan, Anna Thomas, Jared Lichtman, Christopher Bambic, Merrick Pierson Smela, Chenchen Li, and Anton Graf - for many conversations, both scientific and non-scientific, a great deal of support and friendship beginning with the PhD application, and many gaming nights (especially our finally completed Gloomhaven campaign). I would like to thank my closest and best friend Melody Tong for her unwavering emotional support since undergrad through the hardest periods of my life and despite all the chaos that has affected us both. I certainly would not be where I am today without your constant advice and encouragement. Finally, I would like to thank my parents for supporting me in every way imaginable on this journey for so many years. Your focus on giving me a solid academic background, invaluable personal skills, and a clear moral code during my childhood has helped me in so many ways throughout my career.

To my parents for their eternal support, and to Artemis for his eternal companionship

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A protease is an enzyme that cleaves a protein substrate, generally for digestion or inactivation of the target [1]. Proteases are used in a wide variety of biological processes ranging from clotting [2] to apoptosis [3]; they have also been adapted to a number of industrial uses like detergents and food processing as well as 25 FDA-approved medications [4]. However, most proteases are fairly promiscuous [4]. Highly specific proteases that can be engineered to target particular substrates open up a wide range of additional therapeutic possibilities, including targeting biofilms [5], cleaving α-synuclein to treat Parkinson's disease [6], reducing metastasis [7], and augmenting the programmability of CAR-T therapies [8]. Proteases also open up a range of synthetic biology possibilities involving circuit engineering [9–12]. Unfortunately, nature has not provided us with a large number of substrate-specific proteases for any substrate; we will need to develop tools to engineer proteases with arbitrary substrate specificity.

The goal of this thesis is to engineer TEV (tobacco etch virus) protease, a substrate-specific protease, to specifically cut alternative substrates [13]. Engineering protease specificity has proven very difficult for a number of previous methods, ranging from structural engineering [13, 14] to directed or continuous evolution [15, 16]. We will take a data-driven approach combining modern advances in machine learning (ML) for protein engineering with unprecedented data generation capabilities enabled by modern synthetic biology. Our approach develops a number of tools that are useful for engineering TEV protease as well as for broader problems in the space of protein design and engineering. First, we begin with a historical overview of the field of protein engineering, from both computational and experimental perspectives, to explore the tools available to us for the problem of engineering TEV protease.

1.1 Physics-Based Protein Engineering

Proteins are nature's nanoscale machines, capable of performing a wide range of functions including metabolism, cleavage, transportation, and signaling. The ability to design and engineer a protein that can perform an arbitrary specified function has been a holy grail of synthetic and structural biology for years; however, this problem has proven challenging due to the complex dependency between a protein's sequence, structure, and function [17–21]. Over the past 75 years, scientists have pursued many different approaches to untangle these relationships, ranging from purely physics-based computational approaches to experimental evolution-based methods to more modern statistical models.

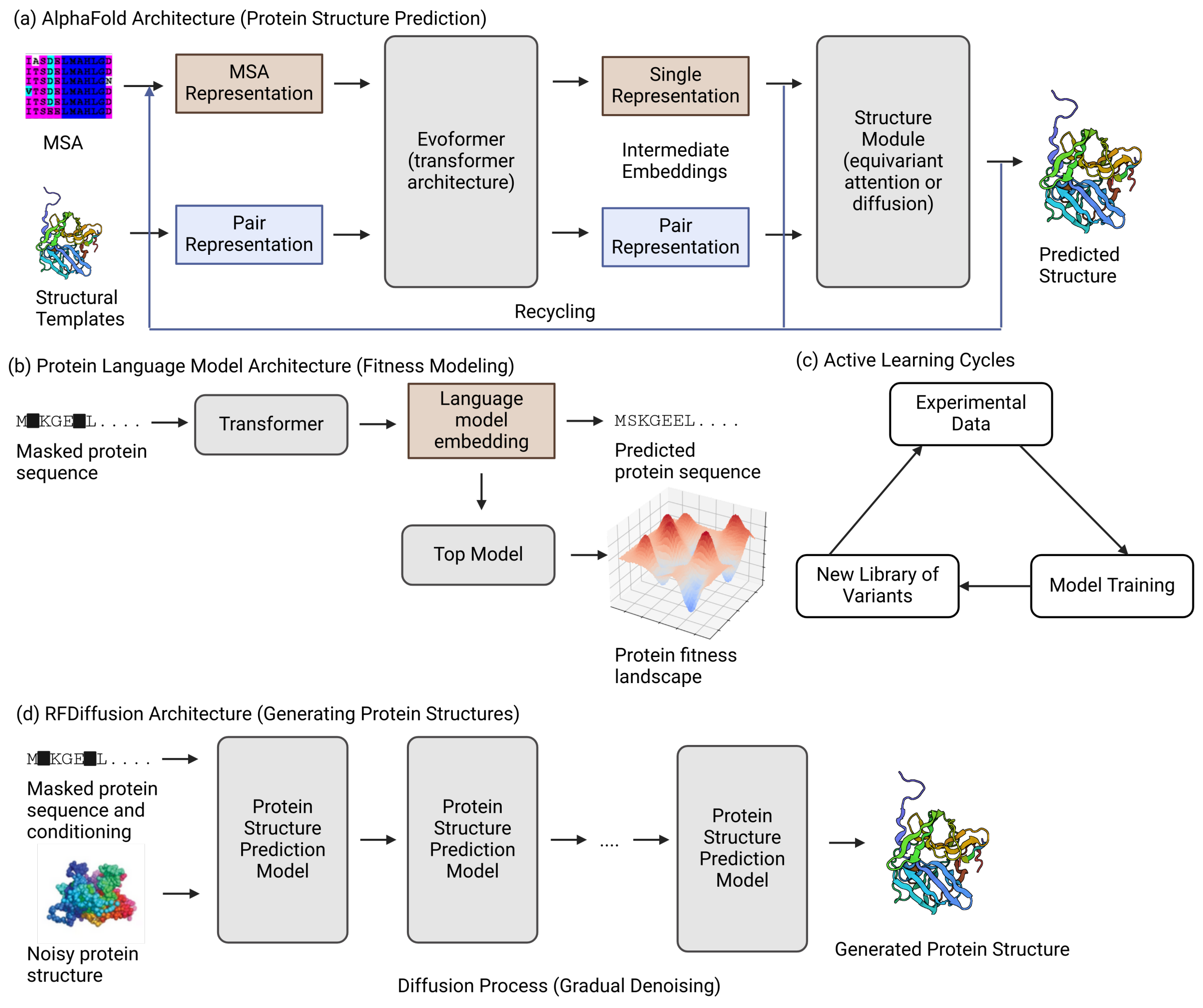
Most early attempts focused on designing proteins with a specified structure, since structures can be predicted or modeled via physicochemical principles and a protein's structure determines its interactions with the environment, which in turn should determine its function [17, 22]. One of the earliest successes by Regan and DeGrado [23] was α-4, a highly stable 4-helical bundle designed based on a simple parametric model of interactions between helices in proteins [23, 24]. Hecht et al. [25] extended the work of α-4 to design Felix by incorporating all amino acids and used helix-preference statistics along with an energy minimization algorithm for designing the structure. These early attempts at protein design focused on helical bundles since they were generally simple and stable to model; they also combined physics-inspired methods like energy functions and minimization with data-driven methods like helix statistics and parametrizations derived from protein structures.

Moving beyond helical bundles, Dahiyat and Mayo [26] redesigned a zinc finger domain, once again using an energy-based scoring function. This work also required understanding interactions associated with β-sheets. Similar methods allowed for the design of membrane proteins and other metal-binding proteins, as well as helical bundles with catalytic or inhibitory activity for protein-protein interactions [22]. However, these methods were still largely restricted to folds found in nature.

Top7 by Kuhlman et al. [27] was the first example of a protein designed with a fold not found in nature. It was designed using a fragment-based approach, where fragments found previously in the Protein Data Bank (PDB) were used to build up to a desired protein structure. This strategy was generalized in RosettaRemodel and used as a general blueprint for flexible backbone protein design [28]. Rosetta-based methods successfully designed a number of possible α-β folds and larger-scale protein assemblies like coiled-coil assemblies [22]. These strategies merge physics-based modeling like energy functions with fragment backbone statistics for backbone design, suggesting that more powerful machine learning tools could be very useful for protein design [29]. However, while these methods succeeded at designing specific backbone structures, they frequently required a large number of attempts for a single structure and did not point a clear path towards designing proteins with a specific function, like enzyme catalysis.

1.2 Machine Learning on Proteins

Machine learning always requires substantial amounts of data, and machine learning on proteins is no exception. Many physics-based protein design approaches described previously relied heavily on structural statistics, like those from the PDB [30] which now includes over 200,000 structures. However, databases of unannotated protein sequences like UniProt are much larger, including over 200,000,000 protein sequences due to the low cost of next-generation sequencing [31]. This dataset is massive, likely the largest biological dataset ever assembled, but generally lacks annotations or any descriptive context for a given protein sequence. The key to applying machine learning on protein data would be learning how to effectively leverage this dataset to learn useful information about proteins.



(a) AlphaFold Architecture (Protein Structure Prediction)
MSA
Structural Templates
MSA Representation
Pair Representation
Evoformer (transformer architecture)
Single Representation
Intermediate Embeddings
Pair Representation
Structure Module (equivariant attention or diffusion)
Predicted Structure
Recycling
(b) Protein Language Model Architecture (Fitness Modeling)
M■KGE■L. . . .
Masked protein sequence
Transformer
Language model embedding
MSKGEEL. . . .
Predicted protein sequence
Top Model
Protein fitness landscape
(c) Active Learning Cycles
Experimental Data
New Library of Variants
Model Training
(d) RFDiffusion Architecture (Generating Protein Structures)
M■KGE■L. . . .
Masked protein sequence and conditioning
Noisy protein structure
Protein Structure Prediction Model
Protein Structure Prediction Model
....
Protein Structure Prediction Model
Generated Protein Structure
Diffusion Process (Gradual Denoising)

Figure 1.1: Popular Machine Learning Architectures Used for Proteins. (a) Protein structure prediction models like AlphaFold 2 feed the multiple sequence alignment (MSA) and structural templates into a transformer architecture and then predict the structure using a structure module with equivariant attention or diffusion. (b) Protein language models are trained to predict masked tokens in a protein sequence and can be used with a task-specific top model to predict protein fitness. (c) Protein fitness models can be improved by means of active learning cycles, where models are iteratively trained on progressively more experimental data. (d) Diffusion models can be used to generate new protein structures by repeatedly applying a protein structure prediction model to an initial noisy structure. Part of (d) is from Ingraham et al. [32].

1.2.1 Structure Prediction and Design

Marks et al. [33] discovered that structural constraints can be inferred by closely examining sequence datasets. Specifically, co-evolutionary relationships in multiple sequence alignments (MSAs) generally correlate with neighboring residues in a protein structure, and by disentangling these co-evolutionary relationships, one can place constraints on possible structures and use those constraints to fold a given protein [33, 34]. This work, while not using modern machine learning techniques, established the first clear signal that datasets of protein sequences generated by evolutionary processes could be used to make inferences about protein structure and hopefully protein function.

With these results, it became clear that structure prediction would involve some combination of evolutionary constraints derived from sequence datasets and MSAs alongside information from datasets of protein structures to translate those evolutionary constraints into predicted structures. AlphaFold used modern machine learning architectures, first convolutional neural networks and then transformers, to synthesize protein sequence and structure datasets and learn these functions, achieving unprecedented accuracy on protein structure prediction (see Figure 1.1a) [35–37]. AlphaFold 2 was the first machine learning model that could arguably be considered to have solved the problem of single-chain protein structure prediction given a sequence and associated MSA; predictions were shown to be roughly as accurate as experiment [37]. AlphaFold 2 applies the transformer architecture to both the MSA and to a residue-by-residue pair representation of a protein to process protein sequences and structural templates into an intermediate embedding, and then applies an equivariant attention-based architecture to this intermediate embedding to generate the final structural prediction [37]. A recycling step reprocesses these results through the entire architecture to moderately improve performance [37].

Since the original publication of AlphaFold 2, many architectural changes have been made to extend its capabilities and improve performance. Competing methods have looked at extensions to multiple chains [38–40], removing the MSA requirement, and making the pipeline easier to run and more efficient [41, 42]. More recently, AlphaFold 3 and RosettaFold All-Atom have both tackled the much more general problem of biomolecular structure prediction for arbitrary molecules [43, 44] incorporating diffusion models into the structure prediction process.

Structure prediction was a very important problem but on its own does not help us design novel proteins. To do that, we need to be able to go from structure to sequence as well. Fortunately, the improved accuracy of AlphaFold makes this reverse problem much more tractable. Early attempts used graph-based models like message-passing neural networks to process the 3D structure and map it to a generated sequence [45–48]. This approach eventually yielded ProteinMPNN, a method with very high sequence recovery that also was validated *in vitro* for designing protein sequences for given backbones [49]. Another approach that showed success was hallucination: the use of a structure prediction model along with an optimization approach like gradient descent to yield a sequence for a given structure [50]. Like AlphaFold 2, these approaches have been extended and modified by many subsequent publications to focus on modalities like antibodies or to improve performance [51–55]. They have also been incorporated into many standard pipelines for use in validating more complicated machine learning models or for generating sequence designs for given desired structures in protein engineering projects.

1.2.2 Protein Language Models

The success of machine learning at protein structure prediction and structure-based sequence design is remarkable, but structure is at best a stepping stone to function, not the final end goal. To understand protein function more broadly, the field has developed protein language models, similar to large language models like GPT-4 in natural language processing [56–58]. These protein language models are trained on large corpora of sequence data with a masked language objective, i.e. predicting masked tokens that have been held out from the entire sequence during training (see Figure 1.1b). In theory, the ability to predict the identity of a masked token should correspond to a model's ability to understand protein syntax (like natural language syntax and meaning), which may eventually correspond to protein structure and function. Early protein language models like

UniRep [59] and TAPE (Tasks Assessing Protein Embeddings) [60] generated embeddings that corresponded to protein function and family and amino acid classifications. These embeddings could also be fine-tuned to predict specific protein functions given experimental datasets [61].

After these initial successes, protein language models have been trained at increasingly large scales [18, 62–64]. Arguably the most notable large protein language model was the ESM series (Evolutionary Scale Model) [65–67]. These large models have been released in multiple scales and generations and have easily usable embeddings that can be applied to a wide variety of protein-related tasks. The ESM series of models claimed good performance on a number of protein function prediction tasks, as well as structure-adjacent tasks like contact prediction and distance maps [65, 67]. However, experimental validation and practical application of these results was not included in the original ESM papers; we will discuss the performance of ESM models among other protein language models on specific tasks in subsequent sections.

Other protein language models developed at around the same time demonstrated similar capabilities and the ability to generate protein sequences, either unconditionally or conditioned on a given function [68–82]. Many of these papers also claim that increasing protein language model scale dramatically improves model performance at some or all of these tasks [65, 75, 76, 82]. More recent protein language models tend to include additional modalities beyond just protein sequence, like protein structure, text-based functional annotations, or biophysical simulation data [67, 83–93].

Protein language models have proven very popular due to their remarkable performance on a wide range of tasks and their relative simplicity, generally requiring only a single sequence or perhaps an MSA, as opposed to more complicated structural data. However, they do come with their weaknesses. Protein language models like the ESM series have not proven as capable at structure prediction as the AlphaFold series [94]; it seems that the additional architectural complexity of AlphaFold is essential for structure prediction. Determining exactly what information is contained in a protein language model embedding remains difficult. While protein language models are an incredibly useful tool for a number of problems, they are not the only method needed to solve problems of protein function prediction and design using machine learning.

1.2.3 Function Prediction and Design

1.2.3.1 Fitness Landscapes and Variant Effect Prediction

There are several related but distinct approaches to function prediction, depending on the amount of prior information and experimental data available for a given problem. One set of problems involves situations where a known protein has the desired function but requires optimization, or has a similar function that needs slight modification. This is the situation we find ourselves in with TEV protease, since we are trying to modify the function of the wild-type to specifically cut alternative substrates. Here, we are interested in exploring the region around a given wild-type protein. We may define the fitness of our protein to be the function we are trying to optimize and aim to predict a fitness landscape, i.e. a map between variants of the wild-type protein and fitnesses.

The literature often conflates many different machine learning benchmarks under the guise of protein function prediction. For our purposes, it is useful to distinguish between three categories of ML-related protein function prediction tasks:

1. Zero-shot variant effect prediction: predicting the effect of a mutation or variant without any experimental data about the protein in question.

2. Fine-tuned fitness prediction: doing the same with some experimental data and usually some sort of fine-tuning.

3. Global protein function prediction: predicting global functions like thermostability, functional classes, or annotations across all of protein space.

We are largely not interested in the last task, global protein function prediction, though it is still important and many literature benchmarks focus on it. We are primarily interested in the first two tasks that closely relate to our problem of local optimization of a known wild-type protein; we will start by examining approaches to the second: fine-tuned fitness prediction.

Given some experimental data, one logical approach is to take protein language model embeddings and fine-tune them or train a task-specific top model on them for the problem in question (see Figure 1.1b). This approach showed very early success with even small numbers of variants (64 or 96) on the early UniRep protein language models [61]. Similar successes were reported on engineering adeno-associated viral capsids and nucleases with substantially larger data sizes [95, 96]; other works have included automated self-driving laboratories [97], active learning loops [98], and lab-in-the-loop automated antibody design [99]. For this approach of using experimental data to train a protein language model-based fitness predictor, there are a number of questions that need to be addressed:

1. How is the experimental data to be generated, and how much will there be?
2. How many rounds of experimental data and ML model training are there to be?
3. What model architecture and protein language model should be chosen?
4. How should the initial dataset be chosen, and how should data for any subsequent round be chosen?

A number of works have tried to address these questions, but the results are often noisy and ambiguous. Wittmann, Yue, and Arnold [100] and Li et al. [101] found that having active variants in the initial dataset was particularly important. Frey et al. [99] recently found the use of biophysical priors and generative models for selecting new variants improved performance in antibody design. However, the field generally lacks a coherent, consistent set of results on how to make a number of these decisions, especially around model architectures and dataset selection; we will explore some of these questions in this thesis.

Given sufficient time and resources, one strategy to supercharge the potential of these models is to use cycles, known as design-build-test-learn cycles in the synthetic biology space or as active learning cycles in the machine learning space (see Figure 1.1c) [98, 102, 103]. In this approach, a model is trained on an initial set of data, designs are selected by the model to both optimize fitness and improve future modeling capabilities, and these designs are tested experimentally and fed back into the model for further training and refinement [97–99]. Such strategies can be optimized by automated laboratories on the experimental side [97] and by techniques such as Bayesian optimization to select designs from a given model [104–114]. These approaches have proven generally successful in a number of contexts and machine learning has been shown to improve overall performance; however, they do take considerably longer due to the requirement of running multiple experiments.

Another approach is to forgo experimental data entirely and rely on a protein language model's ability to perform zero-shot variant effect prediction; this is the first function prediction task listed above. This is generally done by computing the likelihood of a mutated sequence from the protein language model and comparing it to that of the wild-type sequence [66]. Protein language models have frequently claimed superlative performance at this task [65, 67]. However, comprehensive benchmarking often does not support these claims [74, 115]. Zero-shot variant effect prediction has been used as a starting point in protein design campaigns [99–101, 109] but on its own is likely not sufficient to design a new protein. One challenge for this approach is that there is currently no way to inform a protein language model of exactly what function one is trying to design; the model is simply asked to provide a likelihood of a given sequence with no context. Functional annotations currently being added to protein language models may help fix this problem [67, 93, 116].

1.2.3.2 Generative Models for Protein Structures

Given the success of protein structure prediction and structure-based sequence design, another major focus of the field has been on the development of generative models of protein structure. These can be either unconditional generative models that generate arbitrary protein structures with no specified function or conditional generative models that generate protein structures that belong to a particular class of protein or have other important attributes (i.e. binding a specific partner). Early attempts relied on hallucination of backbones as described previously [117–120]. However, diffusion models soon emerged as the most popular approach to generating new protein structures [121]; these models start from arbitrary noise (sometimes constrained in some fashion) and train a neural network to denoise step-by-step towards a protein structure (see Figure 1.1d). Diffusion models are easy to condition based on essentially anything, including a predictive neural network, a binding partner, or some other structural scaffold. Further, the denoising neural network is asked to predict a protein structure given some constraints and a noisy structure, so diffusion models can easily take advantage of highly accurate structure prediction models.

The most popular diffusion model today is RFDiffusion [122] which relies on the structure prediction method RoseTTAFold, similar to AlphaFold 2, and the diffusion model framework described above. It was demonstrated to generate oligomers, scaffold functional active sites around enzymes, and generate *de novo* binders [122]. A wide variety of other diffusion models have also been developed in recent years, covering other applications like adding sequences or working exclusively within sequence space, allowing conditioning based on other parameters like shape or functional descriptions, designing full protein structures beyond just backbones, designing other molecules beyond proteins, designing antibodies, and more [32, 123–131].

The most impressive application of these diffusion models has been the design of *de novo* protein binders, with AlphaProteo claiming hundreds of folds improvement in binding affinity with minimal experimental testing [128] and a number of other projects claiming results close behind using both diffusion- and hallucination-based methods [32, 132–137]. It is not yet clear which of these methods is the most successful, especially since consistent comparisons between them are difficult to find and AlphaProteo is currently closed-source; however, the rapid success and recent progress at designing *de novo* binders suggests that this problem is likely to be solved very soon. Enzyme design or redesign has proven substantially more challenging. Even when starting with a functional active site and focusing on redesigning the surrounding scaffold, the best attempts have yielded enzymes worse than those found in nature [138–144]. A purely *de* *novo* method of altering enzyme specificity or improving activity seems to be out of reach at the moment, though the rapid pace of advances in the field means that it may be possible within the next few years. For now, we still require experimental data, which means we need a method of collecting large quantities of such data for proteases; we now turn to examine experimental methods of measuring protein fitness to use alongside our machine learning methods.

1.3 Measuring Protein Fitness

All machine learning models are data-hungry, so we are most interested in high-throughput experimental methods capable of generating large quantities of data. We focus on a few broad categories of experimental methods that have been proven to generate this quantity of data relatively easily and are therefore suitable for being used in combination with machine learning.

1.3.1 Deep Mutational Scans and Selection-Based Approaches

Deep mutational scanning (DMS) refers to a broad collection of assays that measure some functional property of a library of proteins, i.e. a protein fitness landscape, in high throughput using mutagenesis and some sort of selection assay [145, 146]. These assays generally require two steps: some mutagenesis method to generate a large library of protein variants and some selection step or other method of fitness measurement to measure the actual fitness landscape [145–147]. This general paradigm is quite broad, so in this section, we will focus specifically on DMS assays that require selection steps, and look at other measurement methods in later sections. DMS assays in general have very wide applications, including antibody engineering [148–150], transcription factor functional mapping [151, 152], epitope mapping [153, 154], antibiotic resistance [155], viral spread during the COVID-19 pandemic [156–158], and H3N2 influenza spread [159, 160].

The first step in a DMS assay is the generation of the library of protein variants to measure. Many DMS assays focus primarily on single-mutant variants of a given protein and therefore only require economical means to generate a single-site saturation mutagenesis library [145,

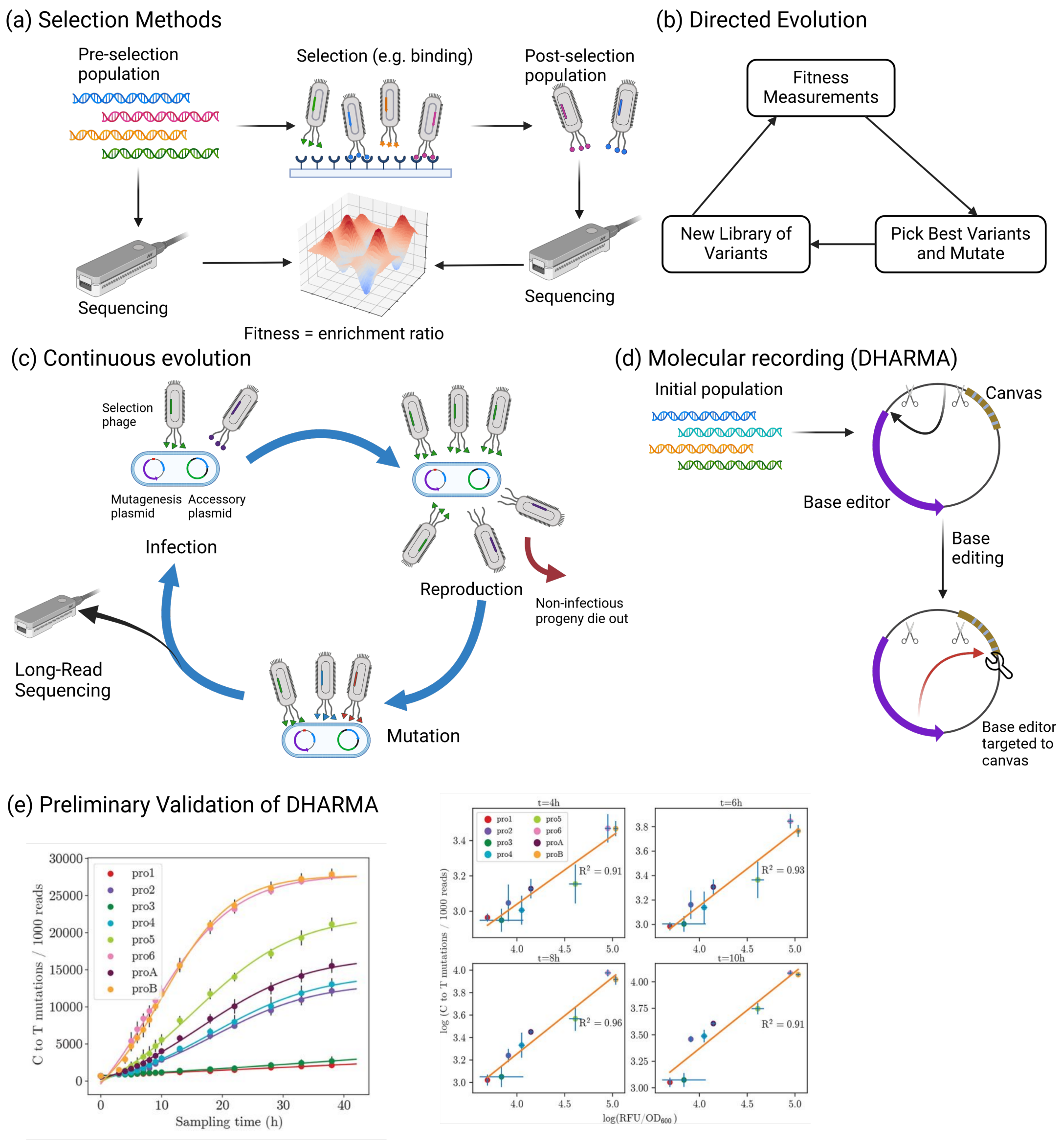


Figure 1.2: High-Throughput Experimental Methods for Measuring Protein Fitness. (a) Selection-based approaches like display assays measure fitness by a selection step, where variants that do not have the desired property like binding are washed away and the remaining variants are measured. Fitness is typically computed as an enrichment ratio of the number of post-selection to pre-selection variants. (b) Directed evolution involves repeatedly measuring fitnesses of libraries of variants, selecting the best variants, and then mutating them to generate new libraries. (c) Continuous evolution streamlines the directed evolution process by linking fitness to phage propagation and using the phage reproductive cycle. (d) Molecular recording approaches like DHARMA link fitness to transcription of a base editor and measure fitness via looking at the number of edits on the canvas sequence. (e) Preliminary validation of DHARMA using a small-scale promoter experiment on GFP against fluorescence measurements. (left) Kinetics of a DHARMA experiment as a function of time. (right) The number of edits measured by DHARMA correlates well with the fluorescence readout.

146]. For this purpose, oligonucleotide libraries can be generated for each codon by a number of standard methods and combined together via pooling [161–164]. For a more random approach that generates variants with higher mutation counts, we can use error-prone PCR (polymerase chain reaction), where mutations are randomly introduced via a low-fidelity DNA polymerase [165, 166]. This approach is potentially biased due to biases in the DNA polymerase used, but can be used to relatively cheaply generate large libraries of mutated variants [146].

The next step is the selection assay used to determine protein fitness. The simplest category of selection assays are assays where the protein function is essential for the survival of the organism, so one can measure fitness by measuring the growth and/or replication of the organism [167]. For example, antibiotic resistance and viral replication can be very naturally measured in this manner [155–160]. We measure the number of variants before and after selection by sequencing; fitness is then generally computed via the enrichment ratio, or the ratio of the number of reads after selection to the number of reads before selection for a given variant [146]. This method is very simple but inherently limited due to the requirement that fitness be directly linked to the growth of the organism in question.

The broadest category of selection steps are display technologies [168]. These generally measure fitness by generating a library with a protein variant displayed on some molecule that encodes the identity of the variant. This library is then passed through a selection step based on the desired fitness and as before, fitness is then computed using the enrichment ratio [169–171]. The original and simplest such method is phage display, originally developed by Smith [172] (see Figure 1.2a). In this method, a library of variants is inserted into gene III on the M13 bacteriophage; gene III encodes pIII (protein III), which forms the tip of the phage particle and conveniently tolerates large insertions [173–175]. The selection process is usually a binding selection, where the phage library is bound to a target and washed [175]. Depending on the system, strategies like polyvalent instead of monovalent display or other phage alternatives are also possible to improve selection efficiency and data quality [175]. These libraries can include from 10^{11} to 10^{12} different members at most [168, 176], enabling the collection of protein fitness landscape data at very large scales. Phage display on its own has already been extensively used for the design of therapeutic antibodies [177, 178], nanobodies, and anti-venom or anti-toxin compounds [178].

If further scale beyond phage display is desired, Roberts and Szostak [179] developed mRNA display, a method of covalently linking a protein with an mRNA molecule with the corresponding coding sequence. The linking process occurs via puromycin, an RNA base analogue that mimics tyrosine and thereby can be incorporated into both the mRNA molecule and the peptide [179]. This method is a fully *in vitro* selection and thereby is not limited by the transformation efficiency of the library, allowing for the preparation of libraries as large as 10^{14} members [168]. mRNA display molecules are also generally stable and compatible with PCR-based amplification and error-prone-PCR-based randomization [168]. Successful applications of mRNA display include macrocyclic peptide engineering and antibody development [180–182]. For greater stability, cDNA display assays can link proteins to cDNA molecules instead of potentially unstable mRNA molecules [183]; this method has been previously used to measure protein stability across all of protein space in massive scale [184].

While DMS assays have been used in a number of contexts for protein engineering without machine learning, they also generate a considerable amount of data which can be used to train machine learning models. Many popular protein fitness benchmarking datasets are generated by DMS assays using selection steps; for example, the largest dataset of protein fitness available today is ProteinGym, with over 217 DMS substitution assays covering 2,500,000 mutants and 66 DMS indel assays covering 300,000 mutants [115]. All of these assays include some sort of selection step, usually either a display assay or a viral replication assay. The other ProteinGym assays are a much smaller collection of clinical substitutions and indels that are not measured via DMS [115]. While other experimental methods of measuring protein fitness have been developed, the vast majority of machine learning scientists modeling protein fitness use the output of a DMS assay as training and benchmarking data for their method.

1.3.2 Directed Evolution

Directed evolution is one of the most popular experimental methods of enzyme engineering [185, 186]. The basic idea is to adapt the process of evolution to design a protein with a particular function: specifically, many rounds of mutation and artificial selection are used to generate new proteins (see Figure 1.2b) [185]. Unlike a DMS assay, a directed evolution campaign generally involves a much smaller number of variants per round but many more rounds instead of just one. The first reported directed evolution experiment by Chen and Arnold [187] evolved the protease subtilisin E to function in a 60% dimethylformamide environment. Since then, directed evolution has been used for a wide variety of applications, including engineering hydrolases [188], esterases [189], recombinases [190], fluorescent proteins [191], and more.

Just as with a DMS scan, the first step of a directed evolution campaign is the generation of a variant library, generally through totally random processes like error-prone PCR [186]. Other techniques like *in vivo* mutagenesis, CRISPR-based approaches, or recombination can also be used for generation of a library [186]. Once a library is generated, a screening method to measure fitness must be used; the simplest approach is a fluorescence-based screen or a closely related approach like fluorescence-activated cell sorting [186]. More complicated fitnesses can be measured via selection experiments using any of the display strategies described in the previous section [186]. After the fitnesses are measured, a new library can be generated around the previously identified high-performing variants, and the process can continue [186].

Directed evolution can be very naturally integrated with machine learning, since the process of directed evolution is similar to design-build-test-learn cycles or active learning loops previously described [192, 193]. This combined approach, known as MLDE (machine learning for directed evolution), has been extensively investigated by the Arnold group, beginning with evolutions on the GB1 protein and on enantioselective carbene insertion reactions [192]. More recent work has established the superiority of MLDE over traditional directed evolution approaches in both simulation and experiment and the importance of rationally selecting starting points with zero-shot variant effect predictors [100, 194]. Other opportunities for improvement include the use of uncertainty quantification via Gaussian processes or ensembles of neural networks with Bayesian optimization [109]. Even though the amount of data generated by a directed evolution campaign is frequently much smaller than a DMS scan, the ability of machine learning models to iteratively learn at every step of the process makes them quite valuable.

1.3.3 Continuous Evolution

One major downside of directed evolution is how time-consuming and labor-intensive multiple rounds can be. Continuous evolution seeks to perform directed evolution during the reproductive cycle of an organism, which requires somehow linking the desired fitness function with the ability of that organism to replicate. Early attempts focused on the evolution of bacteriophage RNA polymerase which is easily directly linked to bacteriophage propagation [195] and the evolution of catalytic RNA ligases via creation of a specialized *in vitro* reproductive cycle [196]. Esvelt, Carlson, and Liu [197] developed a more generally applicable system in phage-assisted continuous evolution (PACE), where the desired fitness function is linked to the reproductive cycle of the M13 bacteriophage (see Figure 1.2c). Specifically, a population of selection phage lacking pIII but possessing library members of the protein in question are introduced to a population of *E. coli.* This population of *E. coli* includes a mutagenesis plasmid to introduce mutations to the reproductive process and an accessory plasmid that contains pIII alongside a gene circuit which links transcription of pIII with the desired fitness function [197]. Functional variants result in increased phage propagation, hence allowing for selection, and mutation occurs due to the mutagenesis plasmid. This process efficiently evolved T7 RNA polymerase to recognize a novel promoter much more quickly than standard directed evolution [197].

It may seem like requiring a link between protein fitness and transcription of pIII constrains PACE to a small number of potential applications, like polymerases. However, a number of PACE circuits have been developed for a wide variety of applications [198], including proteases [16, 199, 200], DNA-binding proteins [201], aminoacyl-tRNA synthetases [202], biosensors [203], Cas9 variants [204–206], soluble proteins [207], dehydrogenase enzymes [208], base editors [209–211], protein-protein interactions [212], riboswitches [213], and prime editors [214]. PACE has also been adapted to include negative selections [215], varied selection stringencies for improved evolutionary success [216], and mutagenesis plasmids with higher mutation rates [198, 217]. For applications beyond *E. coli*, systems analogous to PACE have been developed: VEGAS in mammalian cells [218] and OrthoRep in yeast [219–221].

PACE can still be time-consuming and difficult for experimenters to run, so our lab (DeBenedictis et al. [222]) developed phage- and robotics-associated near-continuous evolution (PRANCE), a roboticized version of PACE. PRANCE runs PACE on a 96-well plate, so 96 simultaneous continuous evolutions can be run, and incorporates automated systems to add phage and bacteria appropriately. It also includes automated feedback control to monitor selection stringency in response to current fitness levels [222]. PRANCE's multiplexing enables the use of control trajectories in a continuous evolution experiment and allows for experiments into the importance of selection stringencies and the stochasticity of evolutionary trajectories [222]. More importantly for our purposes, it allows us to gather 96 times as much data from a single continuous evolution experiment. PRANCE has been used to evolve T7 RNA polymerase, aminoacyl-tRNA synthetases, and quadruplet tRNAs, all much more efficiently than PACE [222]; our lab has also worked on protease evolution with PRANCE (currently unpublished). One can gather sequencing data from either a PACE or a PRANCE experiment relatively easily, but the link between this and a fitness landscape to train a machine learning model on is not so obvious. We will explore this problem in Chapter 2 of this thesis.

1.3.4 Molecular Recording

Molecular recording involves storing information during an experiment in a biomolecule, usually DNA, to be sequenced later; the low cost of high-throughput sequencing makes this a very attractive method of storing large quantities of information [223, 224]. Molecular recording has generally been primarily applied to analysis of temporal biological processes in developmental biology or to gene circuit design and other synthetic biology applications; it has not yet been used substantially in protein fitness measurements until our work here [224]. The first examples of molecular recorders used recombinase-based reporters that turned on fluorescent proteins; these reporters were linked to transcription factors and used to track neuronal development [225, 226]. However, naïve application of these recombinases can only record a small number of signals (depending on how many fluorescent reporters are available); later advances extended these to recombinase state machines which could record multiple states and hold memory [227, 228]. This still does not allow for recording of analog data; the SCRIBE system (Synthetic Cellular Recorders Integrating Biological Events) enabled this by generating many single-stranded DNA molecules that triggered recombination in response to a transcriptional signal [229].

With the development of gene editing via CRISPR systems, much more precise molecular recorders became possible [230, 231]. CRISPR-Cas9 on its own generates double-strand breaks which can be used for lineage tracing as a molecular recorder [232]. More sophisticated applications use self-targeting RNA to guide the CRISPR enzyme to the gene encoding the guide RNA for recording, thereby accumulating the recording data at the same genetic locus [233, 234]. These technologies introduce essentially uncontrolled edits throughout a very large portion of the genome and risk off-target effects that may increase toxicity and damage data quality; base editing and prime editing offer more targeted and refined approaches to molecular recording [235–242]. The CAMERA (CRISPR-mediated analog multi-event recording apparatus) system allows engineered bacteria to record environmental stimuli like antibiotic, nutrient, and light exposure via base editing [243]; extensions have also allowed recording temporal data [244]. Prime editing, the most precise form of gene editing allowing for specific insertions and substitutions, has been used to develop very precise molecular recorders like the DNA typewriter for single cells [245].

Our lab has developed a method of molecular recording focused on measuring protein fitness activity, called DHARMA (Direct High-throughput Activity Recording and Measurement Assay) [246]. We link a desired protein fitness function to transcription of a base editor, specifically a CRISPR-Cas9-mediated cytosine deaminase (see Figure 1.2d). The base editor is targeted to a canvas sequence and randomly causes C⟶T edits; we expect higher fitness to correspond to higher base editor transcription and more C⟶T edits. The variant of the desired protein and the canvas sequence are proximally located on the plasmid and can therefore be sequenced together in the same read using long-read Nanopore sequencing; therefore, this method can be easily scaled to ultra-high-throughput measurements of around 10^6 variants at once [246]. Requiring a link between protein fitness and transcription may seem like a limitation, but we have already explored in the context of PACE how many desired fitnesses can be linked to transcription of a protein (in that case, pIII for phage propagation). PACE circuits can generally be adapted to DHARMA simply by replacing the gIII with a base editor. Preliminary data on a small-scale experiment with 8 different promoters in Figure 1.2e indicate that DHARMA can measure fitnesses; however, there are substantial challenges associated with base editing noise involved in using DHARMA effectively to measure protein fitnesses at scale. We will explore those challenges in Chapter 3 of this thesis.

1.4 Engineering Proteases

Now that we have discussed some of the computational and experimental options available for engineering proteins in general, let's focus specifically on proteases and on TEV protease in particular. Early studies focused on understanding the substrate specificity of TEV protease from a structural perspective [13, 247]. Dougherty et al. [247] identified H46, D89, and C159 as the catalytic triad (following UniProt numbering), i.e. the residues directly responsible for the proteolytic mechanism of cleavage, classifying TEV protease as a cysteine protease; see Figure 1.3. The consensus wild-type substrate sequence is ENLYFQS, with the cleavage site between the Q and the S residues [247, 248]. Structurally, TEV protease consists of 2 β-barrels, with the active site nestled in the interface between the two [13]. Substrate specificity of the protease is largely determined by the P1' and P1 to P6 residues, hence the substrate site description given above [13].

This is due to a number of interactions between the β-sheets of the TEV protease and the substrate itself [13, 247]. Phan et al. [13] identify a number of protease residues that directly interact with these substrate residues, noting that the P5 substrate residue (the N) is completely open and as a result does not constrain substrate specificity. The residues that have the strongest interactions and thereby the strongest constraints on substrate specificity are the P1, P3, and P6 residues [13]. This early work laid out some proposed directions for TEV protease engineering; structure-based *in silico* design succeeded in generating TEV protease variants with higher stability and solubility [14, 249]. Similarly, ProteinMPNN and other structure-to-sequence models have been used to generate TEV protease variants with considerably higher stability and therefore higher activity on the wild-type [250]. However, no reported results in these works indicate success of engineering TEV protease specificity *in silico* using purely structural methods.

A number of experimental approaches to TEV protease engineering have also been tried. Yi et al. [15] used a fluorescence-based selection assay with a large combinatorial library to engineer a TEV protease variant that cleaved a single-mutant substrate, a substrate with a single residue that differed from the wild-type substrate. Verhoeven, Altstadt, and Savinov [251] used directed evolution to similarly reach another single-mutant substrate and Renicke, Spadaccini, and Taxis [252] used a similar large combinatorial screen. More recently, Meister et al. [253] used a fluorescence-based selection system with a larger combinatorial screen to generate a protease variant that cleaved a triple-mutant substrate. Efficiently generating protease variants that cleaved much more distant sub-

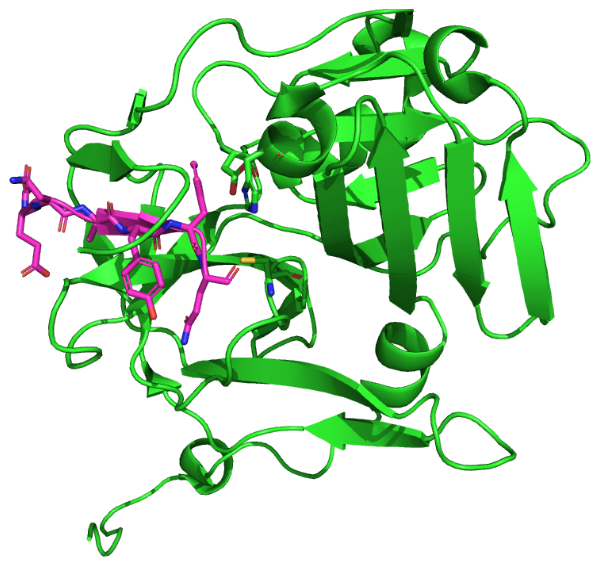


Figure 1.3: Structure of TEV Protease in Complex with Substrate, from PDB 1LVM [13]. The TEV protease is in green and the substrate is in pink. Active site residues (the catalytic triad) are highlighted as sticks along with the substrate side chains.

strates remained challenging until Packer, Rees, and Liu [16] used a number of PACE experiments to successfully generate a TEV protease variant that cleaved a substrate with every residue mutated. This required thousands of generations of PACE and a number of steps where a single residue on the substrate was cleaved at a time, since changing too many residues killed all activity in the protease [16]. Lu et al. [254] used the data from Packer, Rees, and Liu [16] to train a structure-based machine learning model (a graph convolutional network) and claimed to be able to engineer new proteases, but did not show any experimental data supporting these results.

In many past attempts at TEV protease engineering, specificity of the new proteases is not reported or reported to be generally poor; the resulting proteases are often substantially more promiscuous than the wild-type TEV protease, rendering them largely useless for further applications. A consistent, cost-efficient way to engineer TEV protease specificity remains challenging. In this thesis, we will explore the possibilities of combining machine learning and new high-throughput experimentation to tackle this difficult and important problem.

1.5 Outline of this Thesis

The first two chapters of this thesis focus on generating fitness landscapes from experimental data, specifically on understanding and incorporating experimental noise. Chapter 2 focuses on generating a fitness landscape from continuous evolution data, looking at potential applications towards a PRANCE experiment. We develop a latent-variable method that requires data-specific training and demonstrate its reliability and effectiveness on simulation data. We also examine the overall utility of using a PRANCE experiment as a data generation method for machine learning models and find that it is likely not as effective as other approaches.

Chapter 3 presents FLIGHTED, a more principled Bayesian approach based on variational inference to generating fitness landscapes from high-throughput experiments. We apply FLIGHTED to both single-step selection experiments, like the display experiments described previously, and to DHARMA, and demonstrate in both cases that FLIGHTED can effectively account for experimental noise and generate well-calibrated fitness landscapes. We evaluate FLIGHTED's ability to generate effective fitness landscapes under a wide variety of conditions and demonstrate its superiority over simpler alternatives and interpretability. Finally, we use FLIGHTED to better understand in what situations DHARMA or single-step selection experiments are more useful for generating fitness data, which motivates our choice of DHARMA for TEV protease landscape generation.

Chapter 4 benchmarks a number of standard protein fitness models on landscapes of increasingly larger size generated by FLIGHTED and DHARMA alongside one previously public landscape. We establish that FLIGHTED generally improves protein fitness model performance and identify the key modeling decision that matters most when modeling protein fitness: the top model architecture, or the task-specific model that learns our specific protein fitness landscape. We also demonstrate that protein language model scale is not particularly important when modeling protein fitness, but data scale is incredibly important, underscoring the importance of high-throughput experimentation. As data scale increases, we show that model performance becomes increasingly complicated to measure, with different metrics pointing to different models as optimal depending on our desired application. We find that error-prone PCR libraries with at least triple mutants are optimal for training machine learning models, and there appears to be no data plateau: more data is always better, though we do observe diminishing returns at higher scales.

In Chapter 5, with all of this preparation, we finally turn to the actual process of engineering TEV protease. We find that our models are able to effectively learn the fitness landscapes of substrates with a single mutant from the wild-type, but struggle substantially on substrates farther away from the wild-type. Our *in vitro* validation demonstrates that our models are able to engineer TEV protease variants with altered specificity for these single-mutant substrates, i.e. greater specificity than on the wild-type substrate, and that the machine learning model does overall improve performance, i.e. some of the engineered variants are more specific than anything found in the training set. Thus our results demonstrate that we can engineer TEV protease specificity and that both machine learning and high-throughput experimentation are critical to being able to do this. However, engineering TEV protease to specifically cleave substrates far away from the wild-type remains a substantial challenge, and greater modeling capability is needed to tackle this problem.

Chapter 2

Inferring Fitness Landscapes from Continuous Evolution Data

2.1 Introduction

Continuous evolution experiments like PACE or PRANCE offer the possibility of generating large quantities of high-throughput fitness data by sequencing evolutionary trajectories [197, 198, 222]. These experiments are also generally applicable to a wide range of protein fitnesses and protein engineering problems [16, 198–200]. However, an evolutionary trajectory is not generally a fitness landscape that can be fed directly into a machine learning model, like the types of protein fitness models previously discussed. We need an intermediate model to turn evolutionary trajectories into a true protein fitness landscape.

This problem has been most directly tackled by Shen, Zhao, and Liu [255], specifically focusing on the problem of inferring a protein fitness landscape from short-read sequencing of a PACE experiment at multiple regular timepoints. However, with the advent of cheap long-read sequencing [256–258], reconstructing long reads from short-read sequencing is no longer a necessary computational step. Shen, Zhao, and Liu [255] do validate their model performance on reconstructing long reads from short reads, but do not validate their model performance in reconstructing the actual fitness landscape. We found in reproducing and validating their results on simulations that a number of adjustments were necessary to ensure that the original models reliably reconstructed fitness landscapes. Beyond Shen, Zhao, and Liu [255], concurrently to our work here, related work was published on a very similar problem in predicting the fitness of SARS-CoV-2 variants from global surveillance sequencing trajectories [259–264]. We do not directly compare to the SARS-CoV-2 work since the exact problem statement and known data are somewhat different, but the models are very similar in design and spirit.

In this chapter, we present a model to infer fitness landscapes from long-read sequencing data generated by a continuous evolution experiment. Our model is inspired by Shen, Zhao, and Liu [255] but much more thoroughly validated; we show through simulation that our model can recapitulate a fitness landscape accurately under a wide variety of simulation conditions. We demonstrate how many sequencing reads and how frequently we need to sequence to generate accurate fitness landscapes, and estimate how many fitnesses we can measure in a single continuous evolution experiment. We also develop a negative data heuristic to generate further data from a PACE experiment by placing upper bounds on fitnesses of variants we do not see in a trajectory. Our results indicate that while our models are reliable, a continuous evolution experiment is likely not the best way to generate large fitness landscapes; we end up with far too little data given the sequencing costs.

2.2 Model Description

2.2.1 The Probabilistic Graphical Model

For this problem, we begin with a dataset consisting of small numbers (hundreds) of sequencing reads gathered at regular timepoints during a PACE or PRANCE experiment [197, 222] (see Figure 2.1a). We employ long-read Nanopore sequencing at each timepoint, so we should obtain a read of the whole protein and do not need to reconstruct it, but there may be sequencing noise [256–258].

For each variant $g = 1, \ldots, N_{\text{var}}$ in our sample, we assign a fitness f_g which we treat as a latent

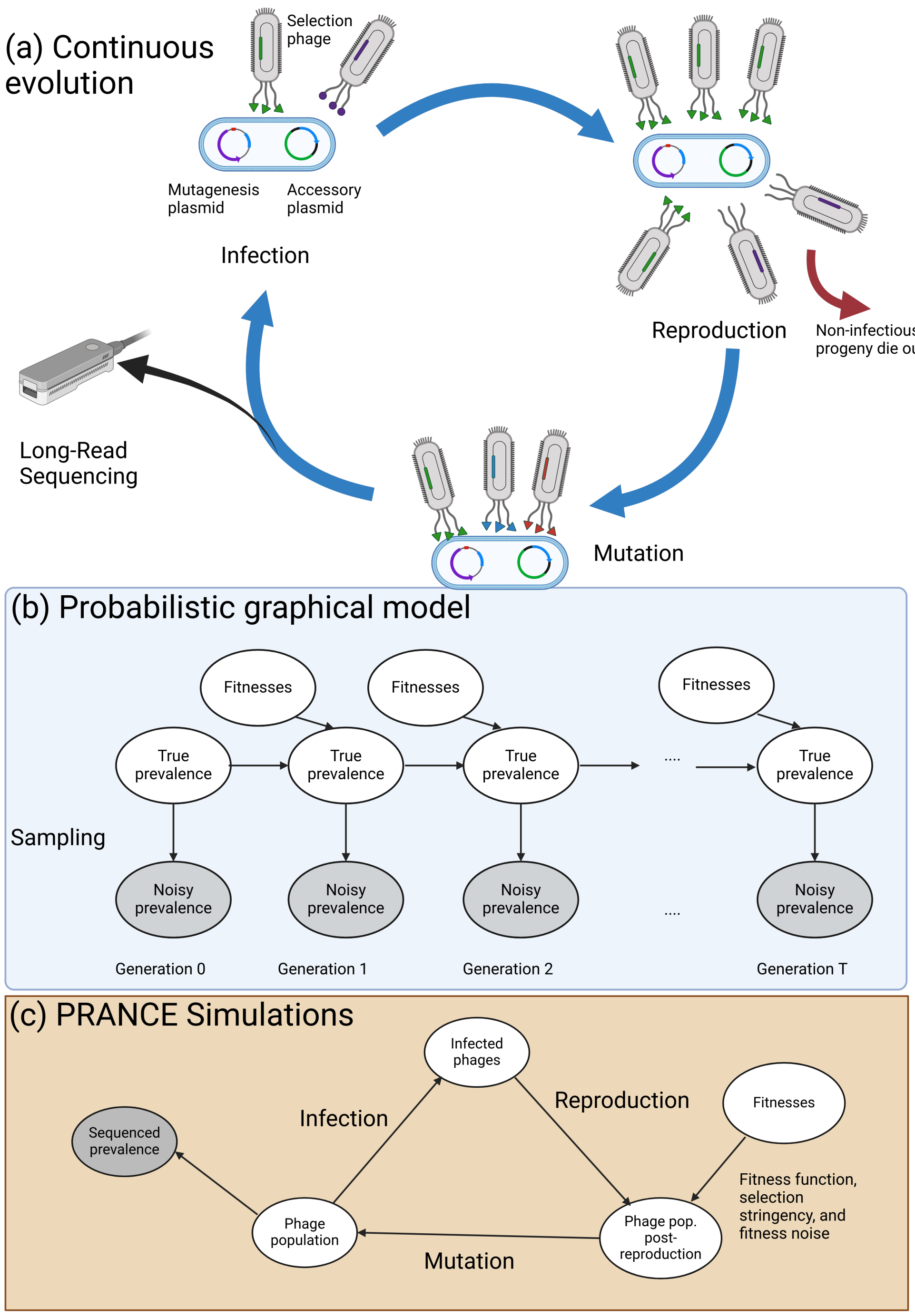


Figure 2.1: Machine Learning to Infer Fitnesses from Continuous Evolution. (a) PRANCE (phage- and robotic-assisted near-continuous evolution) links phage reproduction to the desired fitness of a given protein and then optimizes fitness through a cycle of infection, reproduction, and mutation. (b) A probabilistic graphical model of a continuous evolution process that represents PRANCE, to be optimized with machine learning. (c) The process used to generate PRANCE simulations for training and validation of the machine learning.

variable, a parameter to be directly optimized by the machine learning model. Our observed data consists of noisy relative prevalences $z_{g,t}$ at each timepoint t with N_{samples} drawn per timepoint. Our goal is to learn the true prevalences $x_{g,t}$ and the fitnesses f_g; see Figure 2.1b for the graphical model. By the definition of fitness, we know that approximately

$$x_{g,t} = \frac{e^{f_g} x_{g,t}}{\sum_{g'} e^{f_{g'}} x_{g',t}}.$$

We use e^{f_g} to represent the growth rate of a variant instead of just f_g to ensure that f_g can be any real number, which will simplify the optimization process. Note that we are not accounting for any mutations in this process, which seems entirely contradictory to the point of modeling a system like PRANCE. This is a reasonable assumption because the mutation rates in PRANCE are very low compared to the number of variants we observe; it will be many timesteps before we observe a variant that has emerged via mutation. Just like Shen, Zhao, and Liu [255], we found that omitting mutations makes the model considerably simpler and therefore easier to learn. To account for sequencing and sampling noise, we assume that samples are drawn according to the multinomial distribution, so

$$z_{g,t} \sim \text{Mult}\left(N_{\text{samples}}; x_{1,t}, \ldots, x_{N_{\text{var}},t}\right).$$

2.2.2 The Loss Function

We now need to compute a loss function to optimize in order to learn f_g and $x_{g,t}$. This loss function needs two parts: one to account for the errors in the true prevalence (i.e. the difference between the trajectory predicted by f_g and $x_{g,t}$) and one to account for the sampling noise (i.e. the difference between $x_{g,t}$ and $z_{g,t}$). Unfortunately because we assume mutations do not exist, there is no "correct" way to account for the second, so following [255] we use the KL divergence

$$\mathcal{L}_{\text{prevalence}} = D_{\text{KL}}\left(x_{g,t+1} \middle\| \frac{e^{f_g} x_{g,t}}{\sum_{g'} e^{f_{g'}} x_{g',t}}\right).$$

Hyperparameter	Value
Learning rate	10^{-2}
Number of epochs	1000
α (loss weight)	Optimized between $10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$
Prevalence cutoff	10^{-12}

Table 2.1: Hyperparameters used for inferring fitness from PRANCE.

To avoid NaN overflows, we set a minimum cutoff of 10^{-12} on all the prevalences; this cutoff must be greater than 0 and less than $\frac{1}{N_{\text{var}}}$.

The second part of the loss is simply the negative log likelihood of the multinomial distribution. This can be computed as

$$\mathcal{L}_{\text{sampling}} = -N_{\text{samples}} \sum_g z_{g,t} \log x_{g,t}$$

or equivalently using the cross-entropy loss in PyTorch. In order to combine the two losses, we need a weight hyperparameter α and can then write

$$\mathcal{L} = \mathcal{L}_{\text{prevalence}} + \alpha \mathcal{L}_{\text{sampling}}.$$

To initialize the parameters $x_{g,t}$ and f_g, we set $x_{g,t}$ to be the observed $z_{g,t}$ and f_g to be arbitrarily initialized $\sim \mathcal{N}(-1, 0.01)$. Given these initializations and the loss functions above, we can now use gradient descent via the Adam optimizer to optimize for $x_{g,t}$ and f_g. Optimization was run with a learning rate of 10^{-2} over 1000 epochs. For ease of implementation, we stored $\log x_{g,t}$ as the parameter to be optimized instead of $x_{g,t}$, so our parameter could be an arbitrary real number. Important hyperparameters can be found in Table 2.1.

We found it to be important to filter the data prior to feeding it to the model. Specifically, any pairs of timepoints $(t, t+1)$ where only one variant was present at both timepoints were eliminated. This improved overall model performance considerably and made training more computationally efficient.

This model must be separately trained on any trajectory that it aims to infer the fitness from; it has no ability to generalize from one trajectory. As a result, evaluating model performance requires comparing the predicted fitnesses f_g to true fitnesses that were used in generating this particular trajectory. Since we must know the true fitnesses, we must use simulations to accurately evaluate model performance.

For comparative purposes, we also trained a model that did not account for sequencing noise. This model assumed that the true prevalence at each timepoint was the observed prevalence $z_{g,t}$ and therefore entirely eliminated $\mathcal{L}_{\text{sampling}}$ from the loss function. Since the model was no longer dependent on inferring the complete prevalence trajectory, we fed the model individual data points consisting of pairs of timepoints $(t, t+1)$ as opposed to complete trajectories during training.

2.3 PRANCE Simulations

Prior work in this area has done very limited validation, since it has been primarily tied to past experimental results and therefore focused validation on re-identifying the optimal fitness peak [255]. We wanted to perform a much more thorough validation to better understand the impact of model parameters. As a result, we turned to simulations of continuous evolution experiments to generate large quantities of data needed for this more detailed validation.

In order to simulate a continuous evolution experiment, we need a combinatorial landscape fully evaluated across a small number of sites in a biomolecule. The most convenient landscape for this purpose is the published 4-site landscape on the GB1 protein which measures stability and binding to IgG. This landscape was previously computed via an mRNA display single-step selection assay [265] and has now become a standard benchmark of machine learning models [266]. It has also been previously used to simulate directed evolution and MLDE for very similar analysis [100], making it a natural choice for our work with continuous evolution. There are so many other parameters to vary in generating trajectories from the landscape that we did not consider it necessary to run simulations on more than one landscape.

Figure 2.1c outlines the overall simulation process of a PRANCE experiment. We simulate a number of wells of a PRANCE experiment with $N_{\text{phage}} = 10^{10}$ phage and $N_{\text{bacteria}} = 10^8$ bacteria in each well [222]. Our simulations begin with the entire well consisting of a single variant (selected randomly). Similar to a real-life PRANCE experiment, we use discrete time steps of $t = 0.5$ hours and assume that one cycle of infection, selection, and replication happens per time step. During each time step, we select N_{bacteria} phages to infect each bacterium under the Multinomial distribution, assuming that only one phage can infect a given bacterium.

We now compute the number of offspring produced by a phage of variant g. There are a number of variables we introduce here to explore different possibilities in the fitness landscape. The selection stringency $\alpha = 0.5, 1, 2$ is a parameter that we use to model varying stringencies in a PRANCE experiment, as one of the variables available to the experimenter to optimize the evolutionary process [222]. We also have a fitness transfer function F that converts real fitnesses f to nonnegative numbers of offspring. We explored three possibilities here: a logistic function

$$F(f) = \frac{1 + e^{f_0}}{1 + e^{f_0 - f}},$$

an exponential function

$$F(f) = e^f,$$

and a decay function

$$F(f) = \begin{cases} \frac{1+e^{f_0}}{1+e^{f_0-f}} & f < f_{\text{cutoff}} \\ \frac{(1+e^{f_0})\times\left(e^{-\beta(f-f_{\text{cutoff}})}\right)}{1+e^{f_0-f}} & f > f_{\text{cutoff}}. \end{cases}$$

The last decay function is designed to model the hook effect, an effect where higher fitness can result in reduced phage activity at the upper limit of a PRANCE assay due to interference in a three-body binding interaction [267]. In these functions, we set $f_0 = 0$, $f_{\text{cutoff}} = 1$, and $\beta = 0.2$.

The final key element of our offspring count computation is the inclusion of fitness noise, modeling any other source of noise in the PRANCE experiment; this noise is essential to recapitulate the stochasticity observed in real PRANCE experiments [222]. We use the log-normal distribution with an underlying normal standard deviation of $\sigma = 0, 3, 5$ to model fitness noise as

Simulation Parameter	Possible Values
Fitness transfer function F	Exponential, logistic, decay
Fitness noise σ	0, 3, 5
Mutation rate r	$10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$
Selection stringency α	0.5, 1, 2
Number of samples drawn at each timepoint N_{samples}	25, 50, 100, 200, ∞
Time interval between samples	0.5, 1, 2, 4, 12, 24 hours

Table 2.2: Possible parameters used for running PRANCE simulations for evaluating models.

the log-normal is an ubiquitous long-tailed distribution in biology [268]; in this case, the overall fitness is scaled by a factor of $\frac{1}{e^{\frac{\sigma^2}{2}}}$ to account for the shift in the mean of the log-normal distribution. Our final formula for the number of progeny released per phage is

$$N_{\text{progeny per phage}} = \frac{N_{\text{phage}}}{N_{\text{bacteria}}} F(\alpha f) \frac{z}{e^{\frac{\sigma^2}{2}}}, z \sim \text{Lognormal}(0, \sigma^2).$$

Finally, we need to simulate the mutation process. The mutation rate r varied over 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, and 10^{-10} bases per timestep, based on the minimum and maximum possible mutation rate in a PRANCE experiment [198]. For a given mutation rate r, mutations were simulated on the gene sequence corresponding to a given protein sequence, under the assumption that all possible gene sequences were equally likely for a given protein sequence. For computational efficiency, a transition matrix on the protein sequences was precomputed and used at each timestep, and the maximum possible number of mutations was capped at 1, 2, or 3 depending on the mutation rate.

In summary, there are a number of simulation parameters over which a combinatorial space can be explored (see Table 2.2): the fitness transfer function F, the fitness noise σ, the mutation rate r, and the selection stringency α. We ran 8 simulations for $T = 200$ hours over all possible values of these parameters, and also evaluated the effect of other properties like the number of samples drawn at each timepoint and the time between sampling. For every combination of parameters, we then trained an appropriate model and measured its performance. This performance was measured by the Pearson r between the true fitness and the predicted fitness from the model, for any variant present in at least 5% of at least one sample during the simulation. We found this 5% cutoff to be a reasonable minimum required prevalence for a variant's fitness to be accurately measured in a continuous evolution experiment.

2.4 Results

2.4.1 Model Robustness to Simulation Parameters

We examine the results of our model with varying loss weight α as a function of various simulation parameters, compared to the no-sequencing-noise baseline, in Figure 2.2. In general, our model is highly robust across a wide variety of simulation parameters, consistently achieving Pearson r values above 0.9 regardless of the simulation. In Figure 2.2a, our model is substantially more robust to high fitness noise than the baseline, even though our model is not designed to account for fitness noise. This shows that modeling accounting for one type of noise (sequencing noise) also has the added benefits of mitigating the effects of other types of noise (fitness noise). Our model performance shows no substantial dependence on either mutation rate or stringency in Figures 2.2b and 2.2c, and similarly does not depend strongly on the fitness transfer function (some data shown in Figure 2.4).

The parameters that show the strongest effect on model performance are the number of samples drawn and the time interval between samples, in Figures 2.2d and 2.2e, respectively. Fortunately, these are also the parameters we have the most control over in the experimental setup. In Figure 2.2d, we see that drawing 100 samples per timepoint is sufficient to get essentially perfect model accuracy. In Figure 2.2e, we see that the time interval between samples has a very strong effect on model performance, with 1 hour between samples being ideal. This is likely the largest practical limitation on using continuous evolution experiments with our machine learning method as a means of generating fitness landscapes.

Because fitness noise has such a large impact on model performance, in Figure 2.3 we isolate it and examine the impact of all other variables on model performance. There are no models with

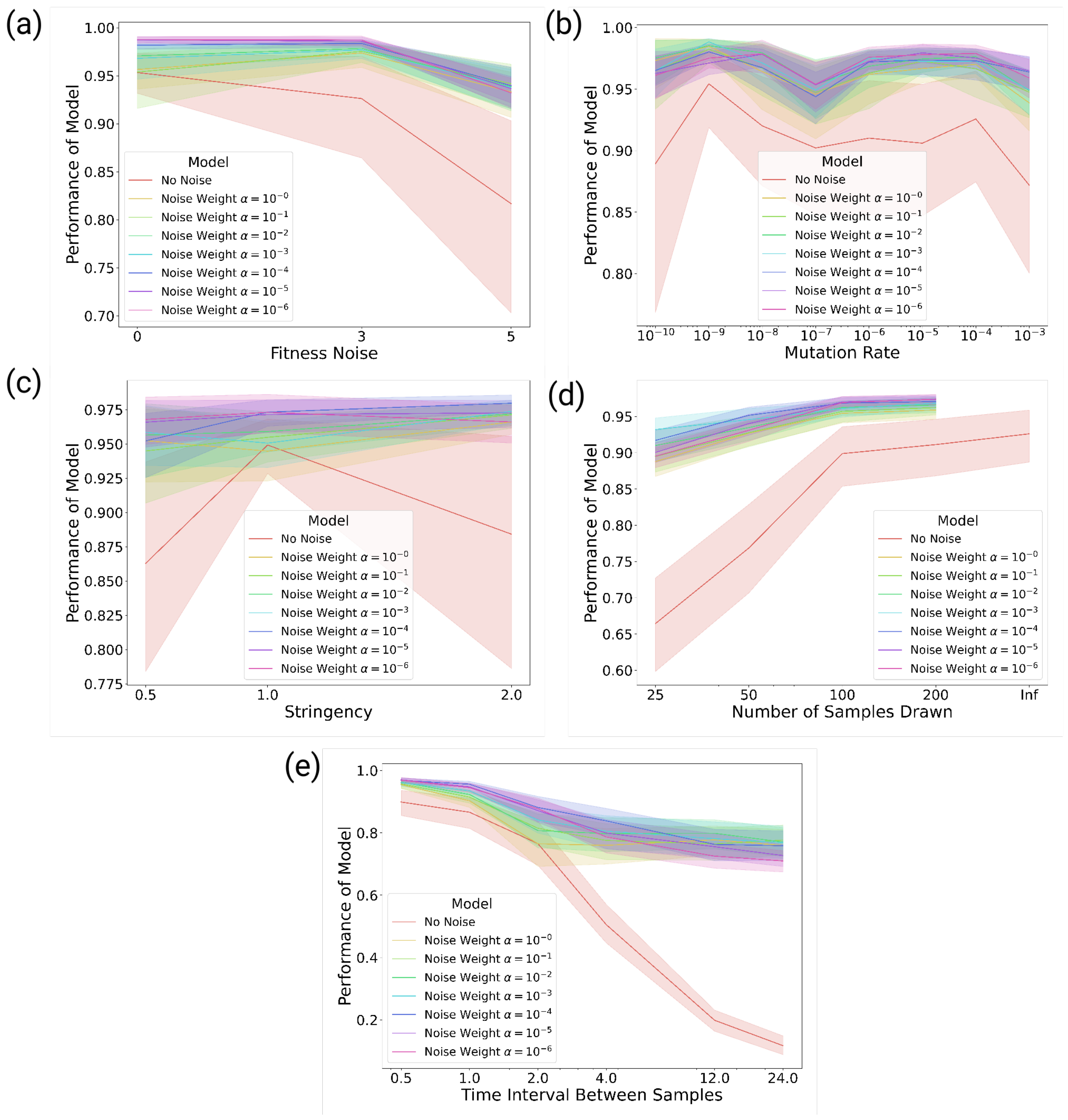


Figure 2.2: Performance of Fitness Inference from Continuous Evolution Method, as a function of various simulation parameters. (a) Fitness noise (noise associated with the fitness function in the simulation). Only simulations with 100 samples per timepoint, timepoints every 0.5 hours, and mutation rates at least 10^{-4} were included. (b) Mutation rate. Only simulations with 100 samples per timepoint and timepoints every 0.5 hours were included. (c) Stringency (difficulty of selection for a simulation). Only simulations with 100 samples per timepoint, timepoints every 0.5 hours, and mutation rates at least 10^{-4} were included. (d) Number of samples drawn. Only models with timepoints every 0.5 hours and mutation rates at least 10^{-4} were included. (e) Time interval between samples. Only models with 100 samples per timepoint and mutation rates at least 10^{-4} were included. Our model is generally robust to all simulation parameters, aside from requiring 100 samples per timepoint and 1 hour between samples.

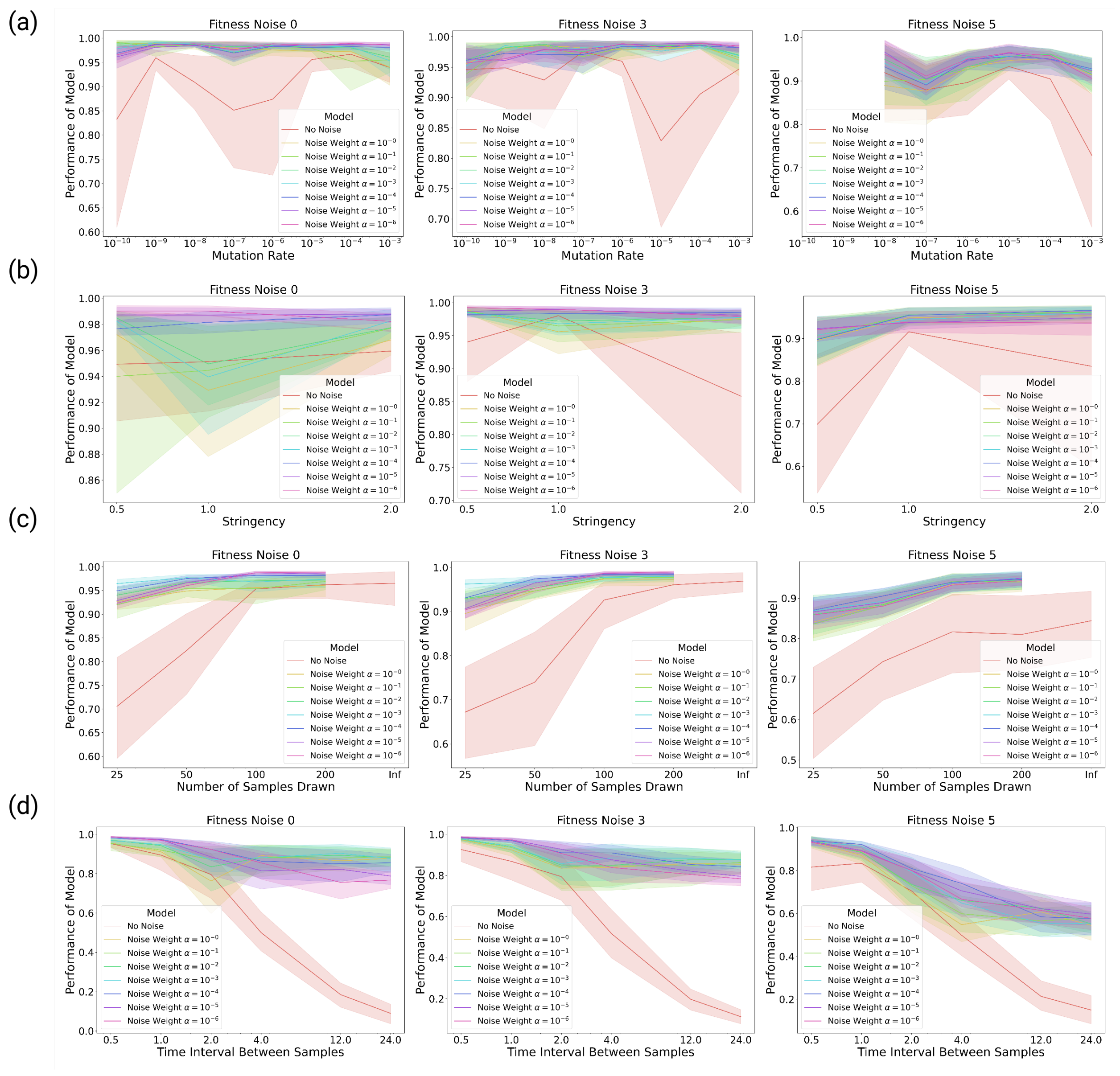


Figure 2.3: Impact of Fitness Noise on Fitness Inference Method Performance. Simulations shown are identical to those in Figure 2.2. (a) Mutation rate. (b) Stringency. (c) Number of samples drawn. (d) Time interval between samples. Our robustness conclusions do not seem to strongly depend on fitness noise.

mutation rate $< 10^{-8}$ and fitness noise 5 since simulations at these parameters generally failed to move away from the starting point. Many of the same general trends still hold, though we can see that the magnitude of model improvement over the baseline increases as fitness noise increases. This indicates that our fitness inference model is highly robust across a wide variety of simulation parameters, and our conclusions about the sampling time and number of samples drawn likely do not strongly depend on fitness noise.

2.4.2 Setting α and Experimental Constraints

We also need to identify the optimal value of the parameter α for use as a weight in the loss function. Based on model performance, we identified $\alpha = 10^{-4}$ as the optimal parameter. With this α, and selecting only simulations with 100 samples per timepoint, timepoints every 0.5 hours, and mutation rates $\geq 10^{-4}$, a t-test indicated our model performed better than the no-noise baseline with $p < 10^{-19}$.

As a final check, we examine the performance numbers with $\alpha = 10^{-4}$ and 100 samples per timepoint in Figure 2.4. Our model is robust with 0.5 and 1 hour between samples, but at 2 hours between samples we see a dramatic failure with the decay fitness transfer function, high stringency, and high fitness noise, and a few other poor performances with high stringency and exponential fitness transfer functions. This underscores the importance of sampling every hour, since more infrequent samples could lead to the model generating a nonsensical fitness landscape.

Prior work in this area has proposed a skew loss as a means of regularization towards relatively sparse inferred prevalences and provided some mathematical justification for this [255]. We tested the skew loss in our more thorough validation framework and found no data justifying its inclusion; the weight on the skew loss was simply too small to make any impact on overall model predictions.

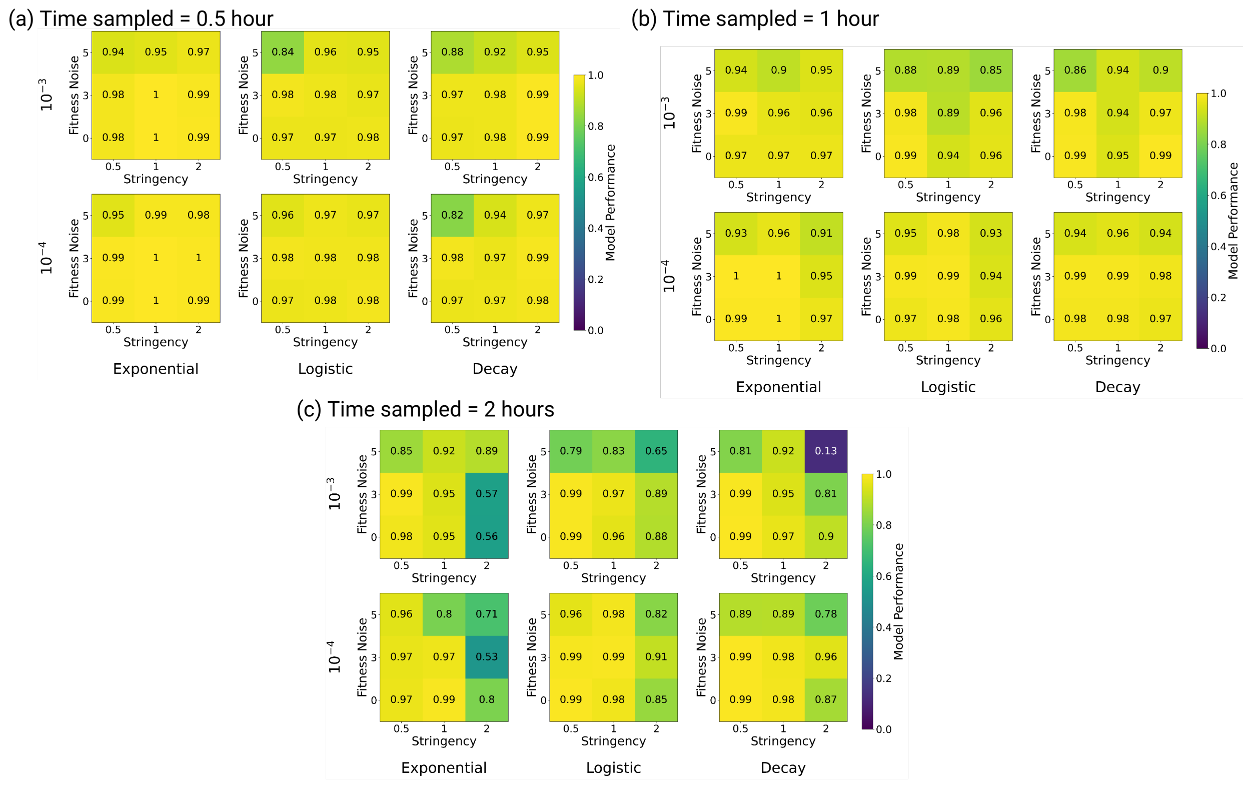


Figure 2.4: Robustness of Fitness Inference from PRANCE Method. This figure shows performance numbers with 100 samples per timepoint at (a) 0.5 hours between sampling, (b) 1 hour between sampling, and (c) 2 hours between sampling. Fitness functions are defined in the Methods section. Our results are generally robust, until the 2 hours between sampling point where the model shows clear failures on a few combinations of simulation parameters.

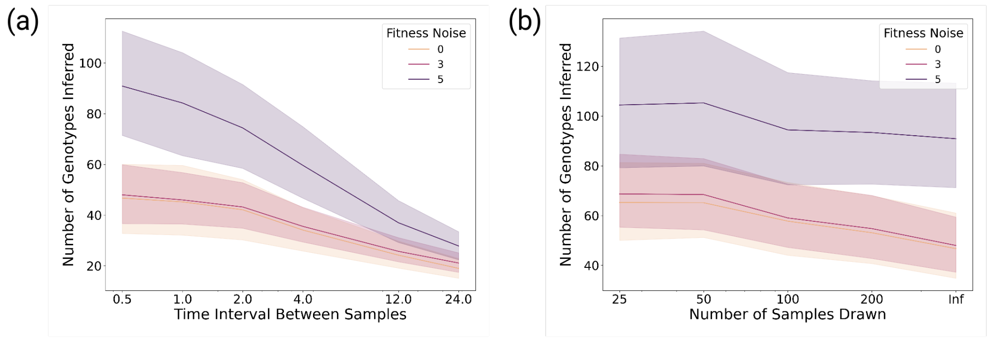


Figure 2.5: Number of Genotypes with Measured Fitness in a Continuous Evolution Experiment, as a function of (a) time between samples and (b) number of samples drawn. We can expect to see at best 100 genotypes in a single PRANCE trajectory, though this may increase with longer experiments and a complete protein landscape rather than a 4-site landscape. We use genotype and variant interchangeably.

2.4.3 Number of Genotypes Measured

For generating fitness landscapes, we would like to know how many genotypes can be measured in a single PRANCE experiment. In Figure 2.5, we see that at high fitness noise, we observe around 100 genotypes on the GB1 landscape, and at no or low fitness noise, we observe around 40 to 60 at best. We do expect these numbers to increase with a longer trajectory or with a complete protein landscape, rather than a 4-site landscape, since there is much more combinatorial space to explore. Further, increasing fitness noise during the experiment appears to be a potential way to increase the number of genotypes observed and generate a larger fitness landscape without severely harming model accuracy.

However, these numbers are obviously very small both compared to the expected training data size for a downstream machine learning model and to the region of the fitness landscape explored by a single PRANCE experiment. Unfortunately, the vast majority of the landscape explored by a continuous evolution experiment is never sequenced, since those variants are not fit and never grow enough to be sequenced. We now develop a heuristic to infer some information about the fitnesses of those variants.

2.5 Incorporating Negative Data

In a continuous evolution experiment with a sufficiently high mutation rate, we expect that all single mutants and many double mutants of predominant populations are likely present. However, many of these variants will not be measured by any sequencing process, because their fitness is too low and they never grow enough to be sequenced. This is negative data – information about the fitness of a variant that we can infer through the lack of data about it – and we want to develop a simple heuristic to capture this information.

2.5.1 Methods

Our heuristic is as follows. For any genotype g that is present beyond a cutoff (set to 5% of the population), we identify its peak $t_{\max}$ in our trajectory. We then identify all single mutants g' that are never observed in this trajectory; these mutants must be present, but are not sequenced. Consider the set $\mathcal{G}$ of all genotypes h that grow after time $t_{\max}$; they must be more fit than our missing variant g'. Therefore, we may place the upper bound $f_{g'} \leq \min_{h \in \mathcal{G}} f_h$. This argument of course assumes that no sources of noise are present at all, and is therefore only an approximate heuristic and not perfectly accurate. We will now evaluate its accuracy on our simulated PRANCE trajectories. Fitnesses here are inferred by the fitness inference model with $\alpha = 10^{-4}$, samples every 0.5 hours, and 100 samples per timepoint.

2.5.2 Heuristic Performance

Figure 2.6a shows the performance, i.e. proportion of predicted fitnesses that actually satisfy the predicted upper bound from the negative data heuristic. We routinely achieve accuracies greater than 90%, with the highest accuracy coming from the exponential fitness transfer function and from high stringency. Interestingly, fitness noise has a small but generally inconsistent effect on the performance of the negative data heuristic, suggesting that high fitness noise does not destroy the heuristic.

Next, we look at the number of genotypes inferred by the heuristic in Figure 2.6b. The number of genotypes inferred varies quite dramatically, with many more genotypes inferred at high fitness noise, high mutation rates, and high stringency. This makes sense: these simulation parameters ensure that more of the landscape is explored and provide more data. Our results demonstrate there are definite advantages to increasing the level of noise and mutation rate in a PRANCE experiment.

We now take a closer look at the performance of our negative data heuristic. Unsurprisingly, in Figure 2.6c, we see that fitness inference model performance and negative data heuristic

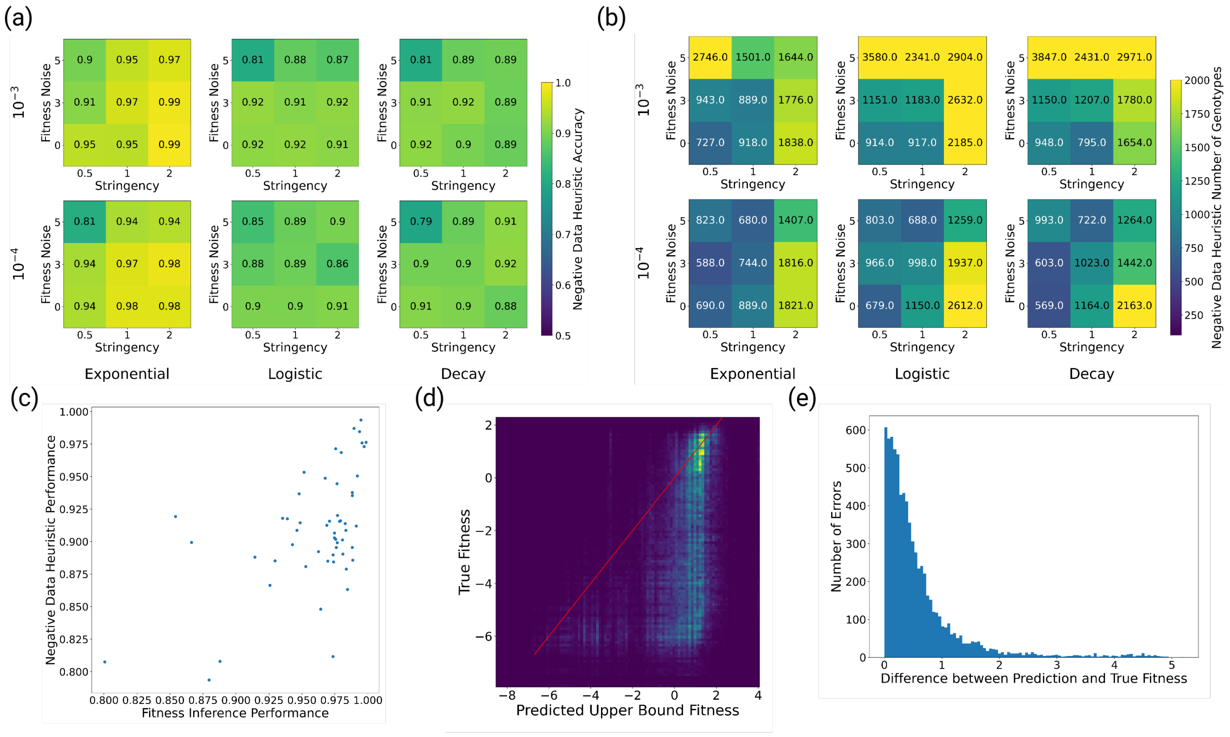


Figure 2.6: Performance of Negative Data Heuristic. (a) The negative data heuristic routinely achieves > 90% accuracy across a wide variety of simulation parameters. (b) More genotypes are inferred with higher fitness noise and higher mutation rates. (c) Better fitness inference predictions also improve the accuracy of the negative data heuristic. (d) The negative data heuristic produces some reasonably tight bounds on highly active variants and some generally useless bounds. (e) Histogram of erroneous bounds produced by the negative data heuristic, which appear to follow a logistic trend. This can be used in downstream machine learning modeling.

performance are reasonably correlated. Since our negative data heuristic relies on fitnesses provided by the fitness inference model, this is expected. In Figure 2.6d, we see that the heuristic is generally predicting relatively tight bounds on reasonably active variants, but sometimes predicts bounds that are so loose as to be borderline useless. Looking specifically at the errors, in Figure 2.6e, we see that the errors appear to form a logistic curve. This suggests the use of a downstream logistic loss function in any machine learning on the negative data produced from a PRANCE experiment, which we now explore the use of.

2.5.3 Modeling Protein Fitness

Given these results, we may now consider the problem of building a protein fitness model using both fitness predictions acquired from our previous model and negative data acquired from our heuristic. Our idea is to incorporate both into a single loss function, with an appropriately placed weight, and use a mean-squared-error for the fitnesses and a logistic loss for the negative data.

Specifically, suppose we have some positive data $x_{+,i}, y_{+,i}$ that correspond to actual fitnesses from our fitness inference model and some negative data $x_{-,j}, y_{-,j}$ that correspond to upper bounds from our negative data heuristic. Given a protein fitness model f, we may then write down the following loss function

$$\mathcal{L} = \sum_i (f(x_{+,i}) - y_{+,i})^2 + \beta \sum_j \left(\frac{1}{1 + e^{\min(f(x_{-,j}) - y_{-,j}, 0)}} - \frac{1}{2} \right).$$

The first term is a standard mean-squared-error loss function as typically used for regression problems. The second term over the negative data is a logistic function if $f(x_{-,j}) > y_{-,j}$, the assigned upper bound for the fitness of $x_{-,j}$, and 0 otherwise. We have a weight parameter β that needs to be fit via hyperparameter tuning; we expect substantially more negative data by around 2 or 3 orders of magnitude with substantially lower accuracy and information when compared to positive data, so it would be reasonable to set β to be around 0.01 or 0.001. This setup allows us to train any machine learning model to predict protein fitness on continuous evolution data using both positive and negative data from our fitness inference model and negative data heuristic.

Ultimately, the accuracy of our heuristic is promising despite its simplicity, but the number of genotypes predicted is still smaller than we would like for a truly high-throughput experiment. The landscape we used here is a 4-site landscape, and in a full protein landscape we expect substantially more single mutants and consequently more predictions from the negative data heuristic. It is possible we could see 10^4 or even 10^5 upper bound predictions from the heuristic in these situations. However, the value of an upper bound as opposed to an exact fitness value for a downstream machine learning model remains unclear.

2.6 Discussion

In this chapter, we have designed a fitness inference model for learning from continuous evolution experiments and thoroughly validated it with computational simulations under a wide variety of simulation parameters. Our results indicate that we require around 100 samples per timepoint and timepoints roughly every 1 hour to get reasonable fitness inference results, and using our methods we can expect high-accuracy fitness measurements of around 60 to 100 genotypes. We have also developed a negative data heuristic to generate upper bounds on fitnesses of variants we do not sequence, and this method can be expected to give us at least a few thousand additional upper bounds. Our methods have been thoroughly validated and therefore should be highly trustworthy for use with real-world experiments.

We now examine the overall utility of these methods from a machine learning and from an experimental perspective. From a machine learning perspective, there are a number of weaknesses in our method, even if it does produce good results. In particular, it needs to be trained on any PRANCE dataset separately, which increases computational cost and the chance that our predictions are not accurate for some unknown reason. It also assumes fundamentally that mutations do not occur in a PRANCE experiment, which is an absurd assumption that is nonetheless required to make the method viable. As a result, the method is a little unnatural and awkward. In the next chapter, we will introduce FLIGHTED, a fitness inference method from any high-throughput experiment that does not require individual training on each experimental dataset and avoids any such unnatural assumptions.

From an experimental perspective, the sequencing costs of this method are considerable: 100 samples per timepoint, hundreds of timepoints, and 96 wells in a PRANCE experiment add up to millions of reads per experiment. Given all of this pooled together, we expect to acquire under 100 fitnesses for specific genotypes with potentially more upper bounds of unknown value for machine learning. This is a low-throughput output for a fitness landscape with a high-throughput cost. So while PRANCE is an excellent method of exploring a fitness landscape and evolutionarily optimizing a protein, it may not be an ideal method for generating large quantities of data for machine learning. We now turn to other methods that are much more capable of generating high-throughput data for machine learning without the complexities of PRANCE.

Chapter 3

FLIGHTED: Inferring Fitness Landscapes from High-Throughput Experiments

The contents of this chapter were previously published in Sundar et al. [269], Sundar et al. [270], and Sundar et al. [271].

3.1 Introduction

Machine learning (ML) approaches have been remarkably successful at solving a variety of protein design problems [19, 64]. Many of these approaches involve training or fine-tuning protein fitness models on data generated by high-throughput experiments to optimize the desired protein function [61, 64, 95, 192]. Since performance improves with training on progressively larger datasets, and biodesign models are currently thought to be more limited by the available data than by the number of parameters [272], acquiring new datasets by measuring molecular activities in high throughput is essential for superior protein design. High-throughput protein fitness measurements are also broadly used for benchmarking and *in silico* retrospective model performance evaluations [65, 69, 73, 74, 266].

However, high-throughput laboratory experiments are inherently noisy. For example, single-step selection experiments such as phage display and other binding-based assays exhibit substantial noise in the measured enrichment ratio [273–276]; concretely, the highest-performing variants in a given single-step selection experiment show essentially 0 correlation between measured and true fitness. Despite warnings in the literature, most ML protein modeling efforts completely ignore any experimental noise present in large fitness datasets, which is tantamount to ignoring error bars. Using noisy datasets for training, benchmarking, and evaluating models can lead to inaccurate conclusions and subpar performance, as ML models can train on and potentially overfit to noise in a given dataset.

Past work on incorporating experimental noise into protein fitness modeling has primarily focused on binding-based deep mutational scanning assays. Some studies have attempted fully analytic solutions to this problem, which typically either limits the downstream modeling to linear regression [273] or forces assumptions about the experimental parameters [274]. Other related work has focused on mitigating sequencing noise, optimizing library design, and performing multiple rounds of selection, which we do not explicitly model here [275–277]. However, the field lacks a general method of accounting for experimental noise that can be easily applied to any experimental method and ML model. We hypothesize that a generalizable framework for accounting for experimental noise could universally improve the performance of any downstream modeling on protein fitness datasets.

In this chapter, we present FLIGHTED (Fitness Landscape Inference Generated by High-Throughput Experimental Data), a new approach based on Bayesian modeling to account for experimental error when generating fitness landscapes, or datasets that map a given protein sequence to fitness, from noisy high-throughput experimental data, i.e. from phage display. FLIGHTED serves as a general preprocessing step for using noisy high-throughput experimental data and is broadly applicable to any high-throughput experiment and any arbitrarily complex ML model. We apply FLIGHTED to single-step selection experiments and a new molecular recording assay (DHARMA) and show that in both cases it generates fitness landscapes with robust, calibrated errors.

3.2 Training a FLIGHTED Model

3.2.1 Accounting for Experimental Noise with FLIGHTED

The standard approach to machine learning for protein design takes a dataset with a single absolute fitness value corresponding to each individual protein sequence, ignoring experimental uncertainty, and uses it to generate a landscape with a single absolute fitness value for every nearby protein sequence. In contrast, FLIGHTED applies Bayesian modeling to generate probabilistic fitness landscapes in which the predicted fitness for each sequence is represented by a probability distribution (see Figure 3.1a). In practice, we approximate these probability distributions as normal distributions and produce means and variances for each protein sequence.

To translate the measured activities and experimental uncertainties of sequences in the training dataset into probability distributions describing the fitness of sequences, FLIGHTED explicitly models the known sources of experimental noise for each major type of experiment. As a result, a different version of FLIGHTED must be trained for each type of experiment. All subsequent datasets generated by experiments of that type, including in other laboratories, can use that version of FLIGHTED without further modification.

Training FLIGHTED requires:

1. knowledge of major sources of experimental noise for a given type of experiment
2. a noisy "calibration dataset" generated by running a high-throughput experiment of that type.

Training yields the FLIGHTED model, which uses our physical knowledge of noise to computationally generate a predicted distribution of experimental results from a noiseless fitness landscape, and the FLIGHTED guide, which predicts a probabilistic fitness landscape from noisy experimental results (see Figure 3.1b). The FLIGHTED guide and the FLIGHTED model are trained together to minimize the difference. As long as we can accurately describe the experimental noise in

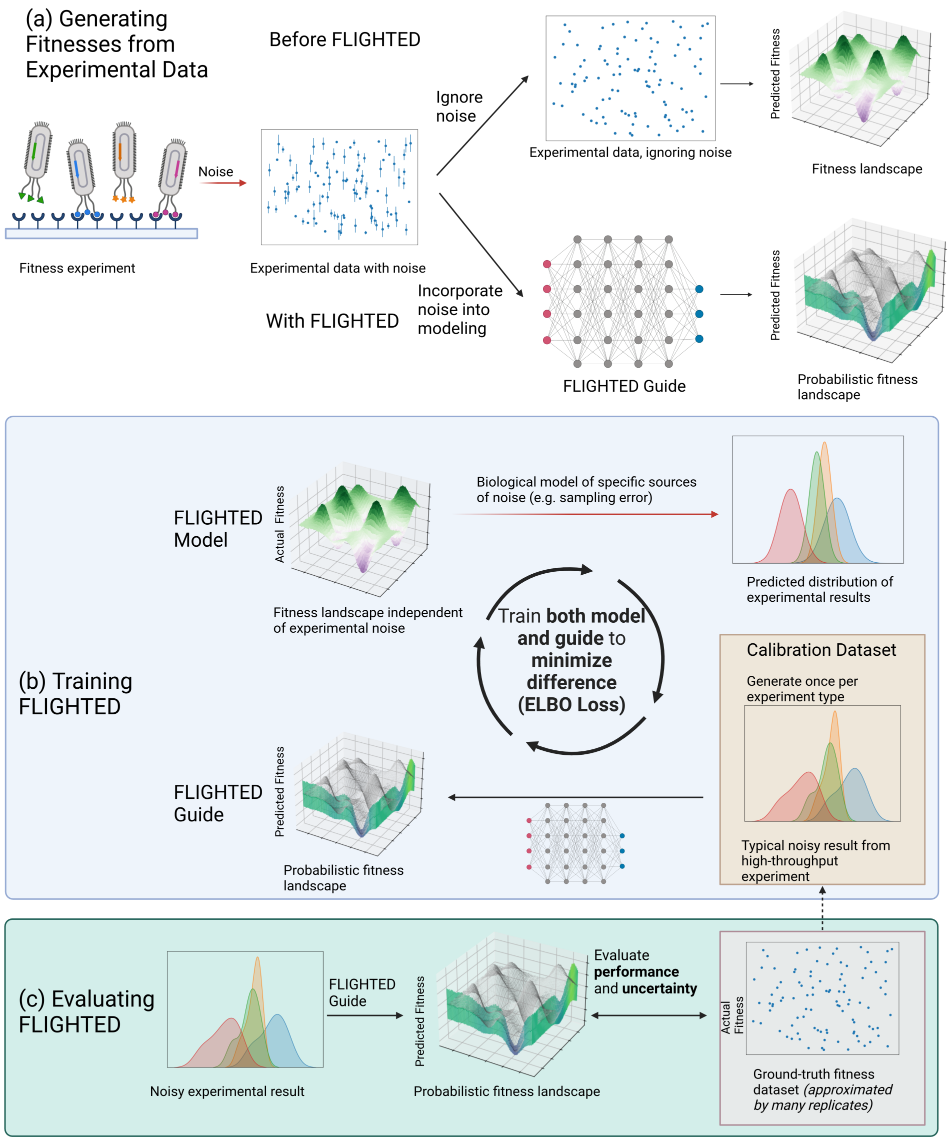

(a) Generating Fitnesses from Experimental Data
Before FLIGHTED
Noise
Fitness experiment
Experimental data with noise
Ignore noise
Experimental data, ignoring noise
Predicted Fitness
Fitness landscape
With FLIGHTED
Incorporate noise into modeling
FLIGHTED Guide
Predicted Fitness
Probabilistic fitness landscape
(b) Training FLIGHTED
FLIGHTED Model
Actual Fitness
Fitness landscape independent of experimental noise
Biological model of specific sources of noise (e.g. sampling error)
Predicted distribution of experimental results
Train both model and guide to minimize difference (ELBO Loss)
Calibration Dataset
Generate once per experiment type
Typical noisy result from high-throughput experiment
FLIGHTED Guide
Predicted Fitness
Probabilistic fitness landscape
(c) Evaluating FLIGHTED
Noisy experimental result
FLIGHTED Guide
Predicted Fitness
Probabilistic fitness landscape
Evaluate performance and uncertainty
Actual Fitness
Ground-truth fitness dataset (approximated by many replicates)

Figure 3.1: Summary of the FLIGHTED Approach to Incorporating Experimental Noise from High-Throughput Experiments. (a) A high-throughput experiment without FLIGHTED (fitness landscape inference generated by high-throughput experimental data) generates a readout that measures fitness and incorporates considerable noise. Before FLIGHTED, practitioners generally ignored this noise when generating a fitness landscape. With FLIGHTED, practitioners can generate a probabilistic fitness landscape that accounts for experimental noise in these high-throughput datasets. (b) To train FLIGHTED, the FLIGHTED model models a specific source of noise in a particular high-throughput experiment and generates a distribution of noisy experimental results from a fitness landscape. We then use the FLIGHTED guide (a neural network) to convert experimental readouts to fitness predictions with calibrated errors. We train both the model and the guide to minimize the difference, i.e. the ELBO (evidence lower bound) loss between the two distributions of experimental results. Training uses experimental results from a calibration dataset but does not require ground-truth fitness values. (c) FLIGHTED guide is evaluated using ground-truth fitness values from a calibration dataset for predictions and uncertainty. FLIGHTED guide can then be used to measure fitnesses from any high-throughput experiment of the same type, beyond the calibration dataset.

the calibration dataset, the FLIGHTED guide will generate a probabilistic fitness landscape that accurately accounts for the noise, which should theoretically improve accuracy over standard approaches that do not account for noise.

Evaluating FLIGHTED involves comparing the probabilistic fitness landscape generated by the FLIGHTED guide to the ground-truth fitness – which by definition is unknown, but can be approximated by averaging the results of a very large number of experimental replicates. Importantly, FLIGHTED is trained in a fully unsupervised fashion using only noisy experimental results from the calibration dataset; ground-truth fitness values are only used for evaluation. To ensure that the ground-truth fitness values are not used in training, the calibration dataset must be separated from any data used to approximate the ground-truth before training begins (see Figure 3.1b-c).

In this chapter, we train two versions of FLIGHTED: one for single-step selection experiments, and one for DHARMA, a molecular recording assay that links fitness to base editing mutations in a canvas [246]. These versions of FLIGHTED can be used by other researchers intending to train on datasets generated by single-step selection experiments or by DHARMA.

3.2.2 The Mathematical Framework for FLIGHTED

We now describe the mathematical framework that justifies FLIGHTED; see Figure 3.2b. Abstractly, a high-throughput experiment generates a library of protein variants x that map onto fitness values z defined by their molecular activity under a certain set of conditions, but the noisy experimental process imperfectly generates a noisy result y from the actual fitness (see Figure 3.1, top). For each experiment, we can write down a probabilistic graphical model for this noise generation process $z \to y$ using our biological knowledge. That is, we generate an mathematical model for each source of noise and use it to compute a distribution of possible experimental results y given the fitness z of a protein sequence. This constitutes the FLIGHTED model for a given experiment (see Figure 3.1b); any sources of noise not modeled in this process will not be accounted for by FLIGHTED. For example, in the context of single-step selection experiments, we model sampling noise, or the noise that occurs when sampling from a larger population to sequence specific variants. Other sources of noise in the experiment, like proteins that bind to plastic instead of the desired molecule, are not modeled and cannot be accounted for.

Because running inference on an arbitrary noise generation model to determine the fitness of each variant is computationally intractable, we use stochastic variational inference (SVI) to generate a FLIGHTED guide, or variational distribution, that predicts probabilistic fitness $\hat{z}$ with error from experimental results y [278]. Specifically, we treat the fitness z of each variant as a local latent variable. To avoid an arbitrarily large latent space, we use the amortization trick to learn a guide function that maps experimental results y to predicted fitness values $\hat{z}$ instead; this is the same trick that is used in the variational autoencoder (VAE) and the conditional variational autoencoder (CVAE) [279, 280]. The guide function approximates the posterior distribution of fitness as a normal distribution. We then train our model and guide simultaneously to minimize the ELBO loss between the guide-predicted fitness values $\hat{z}$ and the fitness landscape z, given noisy experimental results from the calibration dataset. Using SVI has two main advantages. First, model parameters can be used for any number of subsequent experimental measurements after being determined once on a calibration dataset; second, all models can have arbitrary complexity (including neural networks).

We can also view FLIGHTED as analogous to the CVAE, since both have the same graphical model (see Figure 3.2a) [280]. In the CVAE setup, we have an input x, an output y, and a latent variable z. We begin with a data-specific latent prior $p_\theta(z|x)$ and have a probabilistic graphical model $p_\phi(y|z,x)$ to generate the output. Our variational distribution or guide then predicts $q_\psi(\hat{z}|y,x)$. Within this framework, the ELBO loss may be written as

$$\mathcal{L}(x, y; \theta, \phi, \psi) = -D_{KL}(q_\psi(\hat{z}|x, y)||p_\theta(z|x)) + \frac{1}{L}\sum_{l=1}^{L} \log p_\phi(y|x, z^{(l)})$$

where L is the number of samples or particles used to approximate the loss, $z^{(l)}$ represents a given sample of the latent variable, and θ, ϕ, ψ represent learnable parameters. By analogy, here

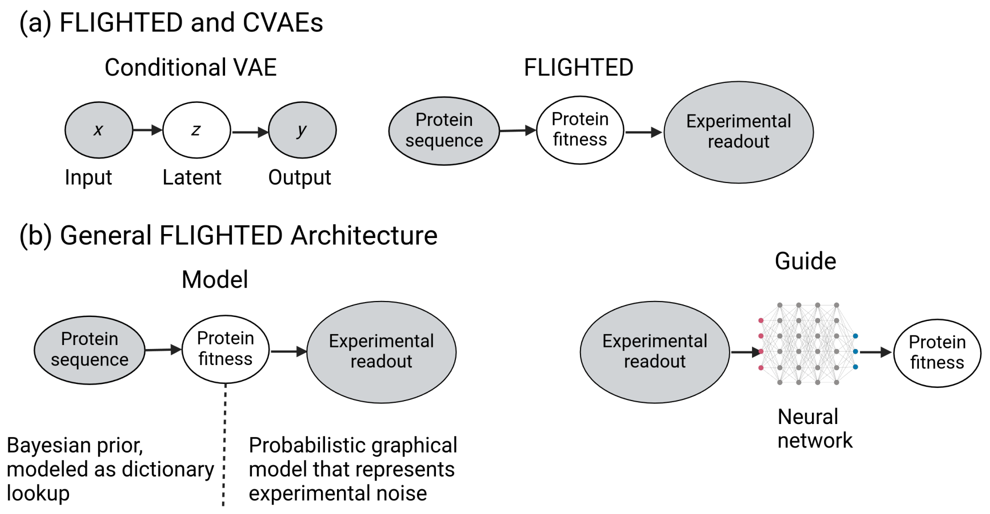


Figure 3.2: Architecture of a General FLIGHTED Model. (a) The probabilistic graphical models between FLIGHTED and the conditional variational autoencoder (CVAE) are analogous, motivating our approach, even though the applications of the two are completely different. (b) By analogy, the FLIGHTED model predicts the experimental readout from protein sequence via a sequence-to-fitness function (implemented as a dictionary lookup) and a probabilistic graphical model that models experimental noise. The FLIGHTED guide predicts protein fitness from the experimental readout using a neural network.

the sequences are the input, the experimental result is the output, and the fitness is the latent variable. FLIGHTED's setup is the same as the CVAE, except instead of using an arbitrary neural network to compute $p_\phi(y|z, x)$, we write down a probabilistic graphical model based on biological knowledge of the experimental process. The use of this probabilistic graphical model explains why learning from only experimental readouts – and not ground-truth fitness – is possible: our biological knowledge of the sources of experimental error constrains the space of possible models, whereas a neural network would not have any such constraints. This analogy only works because the graphical models are identical: the downstream applications of FLIGHTED and the CVAE are completely different.

To evaluate FLIGHTED's performance (see Figure 3.1c), we measure both performance and uncertainty prediction of the FLIGHTED guide's predicted fitness values and errors. The FLIGHTED guide can then be used to denoise other experimental results generated from the same high-throughput experimental assay, even those involving different molecular libraries and targets (see

Figure 3.1a).

The procedure we have outlined allows for the generation of trustworthy fitness measurements accounting for experimental error solely from using the FLIGHTED guide on a given experiment. While we demonstrate just two applications of FLIGHTED, it can be easily generalized to other assays; the only arbitrary step in the procedure described above is selection of neural network architecture for the FLIGHTED guide.

All models were implemented using the probabilistic programming package Pyro [281] alongside PyTorch [282]. Use of Pyro makes model development substantially easier due to rapid iteration of model architectures.

3.3 Single-Step Selection

We now turn to the application of FLIGHTED to single-step selection assays: experimental assays in which a library of variants is produced, a selection step occurs to separate more fit variants, and the selected population is measured; see Figure 3.3a [169, 175, 283]. Examples of single-step selection experiments include mRNA display, phage display, and many deep mutational scanning (DMS) studies. They comprise a substantial portion of the still-limited fitness landscapes used for machine learning [265, 266]. Fitness is usually measured as the enrichment ratio, or the ratio of variant sequences sampled post- to pre-selection [169]. However, the enrichment ratio is an inherently noisy measurement due to sampling noise, the noise associated with sampling the reads to be sequenced from a larger library [169]. Sampling noise is unavoidable in a single-step selection assay and the only source of noise we choose to model in FLIGHTED [169].

3.3.1 Methods

3.3.1.1 Dataset Generation

Single-step selection experiments were simulated on a landscape of $N_{\text{var}} = 20^4$ variants (representing a peptide of length 4), with the ith variant for $1 \leq i \leq N_{\text{var}}$ having a fitness or

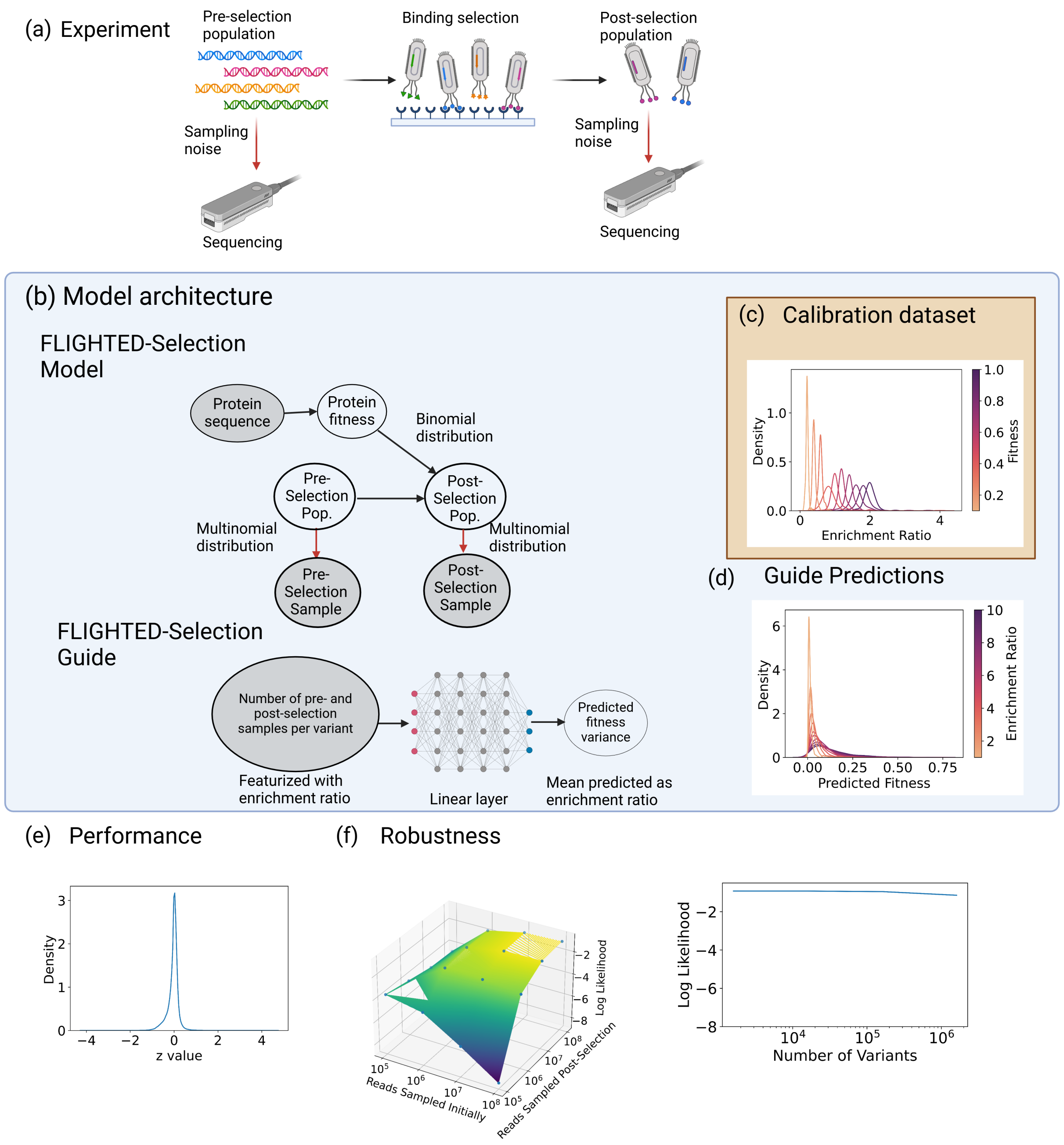


Figure 3.3: FLIGHTED-Selection Model Performance. (a) Single-step selection assays always include sampling noise from finite read depth when reads are sampled for sequencing. (b) The probabilistic graphical model used for FLIGHTED-Selection and model architecture for the guide (variational distribution or approximate posterior). (c) Simulations indicate that the enrichment ratio has considerable noise in a single-step selection experiment due to sampling. (d) Typical FLIGHTED-Selection predictions for a given enrichment ratio. (e) FLIGHTED-Selection model predictions are well-calibrated. (f) FLIGHTED-Selection model performance is robust to changes in number of reads selected pre- and post-selection if there are more reads than variants (20^4 here). Model performance is also robust to the number of variants.

selection probability p_i uniformly distributed between 0 and 1, i.e. $p_i \sim \mathcal{U}([0,1])$. The total initial population $N_{\text{init}} = 10^{11}$. Each variant i had an initial population drawn from the Dirichlet distribution $N_{\text{init},i} \sim N_{\text{init}}\text{Dir}(\alpha), \alpha = (1,\ldots,1)$ with length N_{var}. An initial pre-selection sample $N_{\text{sampled, init},i}$ was drawn according to a multinomial distribution $\text{Mult}\left(N_{\text{init}}; \frac{N_{\text{init},1}}{N_{\text{init}}}, \ldots, \frac{N_{\text{init},N_{\text{var}}}}{N_{\text{init}}}\right)$, with $N_{\text{sampled, init}} = 10^8$ samples being drawn.

In the selection step, the ith variant had a post-selection population drawn from the binomial distribution, i.e. $N_{\text{final},i} \sim \text{B}(N_{\text{init},i}; p_i)$ and $N_{\text{final}} = \sum_i N_{\text{final},i}$. The post-selection sample $N_{\text{sampled, final},i}$ was similarly drawn according to a multinomial distribution $\text{Mult}\left(N_{\text{final}}; \frac{N_{\text{final},1}}{N_{\text{final}}}, \ldots, \frac{N_{\text{final},N_{\text{var}}}}{N_{\text{final}}}\right)$ with $N_{\text{sampled, final}} = 10^8$ samples being drawn. We ran 100 simulations on a single landscape to generate training data and to assess the noise levels in the enrichment ratio as shown in Figure 3.3c.

The parameters of this simulation were selected to be similar to those used in generating the GB1 dataset [265]. Simulations with smaller numbers of reads drawn both pre- and post-selection were used to assess robustness. Simulations were also run with varying Dirichlet parameters and using a beta distribution for the selection probability instead of a uniform distribution; these did not significantly affect the noise found in the enrichment ratio. Specifically, we drew $p_i \sim \text{Beta}(1, b)$, and varied the scalar b between 1 and 100. This represents fitness values that peak at lower or higher values as opposed to being uniform between 0 and 1. We also drew the initial population $N_{\text{init},i} \sim N_{\text{init}}\text{Dir}((a,\ldots,a))$ with the scalar a varied between 0.1 and 10, allowing for biases in the initial population towards one variant.

3.3.1.2 The FLIGHTED-Selection Model

The probabilistic graphical model for FLIGHTED-Selection is shown in Figure 3.3b. Given a fitness landscape and a pre-selection population, we randomly sample and sequence pre-selection, select a post-selection population based on the provided fitness values, and randomly sample and sequence post-selection. The guide model as shown in Figure 3.3b uses the logit of the enrichment ratio as the mean and predicts the variance using linear regression on the number of pre- and post-selection samples. FLIGHTED-Selection is trained on the calibration dataset of simulations described previously. We now specify the mathematical details of these approaches.

The sequence-to-fitness function is simply a dictionary lookup, with each variant i mapped to a fitness mean m_i initially sampled from $\mathcal{N}(0,1)$ and variance σ_i sampled from Softplus$(\mathcal{N}(0,1))$, where the softplus function is used to ensure that the variance is positive. Both m_i and σ_i are learnable. The fitness z_i of each variant is sampled $\sim \mathcal{N}(m_i, \sigma_i)$ and the probability of binding $p_i = \sigma(z_i)$, where σ is the sigmoid function used to ensure the probability is between 0 and 1. To avoid issues near the boundaries, fitness values are clamped to be between 0.001 and 0.999. The initial population of each variant $N_{\text{init},i}$ is a local parameter in Pyro. We then follow the noise generation procedure described in the Data Generation Section to predict the distribution of experimental results. Note that while the noise generation procedure is mathematically identical, we are using it for different purposes; we previously used it for data generation, and are now using it for defining the probabilistic model. Robustness tests of FLIGHTED-Selection on changes to the data generation process ensure that FLIGHTED is not overly specialized to this single noise generation procedure.

The guide predicts the fitness mean for a given experiment as logit $\left(\frac{\text{ENR}}{100}\right)$, where ENR is the enrichment ratio of post-selection to pre-selection sampled reads:

$$\text{ENR} = \frac{N_{\text{sampled, final},i}/N_{\text{sampled, final}}}{N_{\text{sampled, init},i}/N_{\text{sampled, init}}}.$$

The scale factor of 100 ensures that the transformed enrichment ratio is between 0 and 1 before being fed into the logit function; it can be changed arbitrarily with no impact on model performance after retraining. The variance is predicted by a linear model with 2 inputs: $\frac{N_{\text{sampled, init},i}}{N_{\text{sampled, init}}}$ and $\frac{N_{\text{sampled, final},i}}{N_{\text{sampled, final}}}$. The predicted variance is also transformed by a softplus function F to ensure that it is positive.

Each data point for the FLIGHTED-Selection model consists of an experiment or simulation, not the outcome of a single variant within an experiment, because the number of reads of a given variant sampled pre- and post-selection depends on the population of other variants in the sample.

Hyperparameter	Value
Number of amino acids (alphabet length)	20
Sequence length for dataset	4
Learning rate	10^{-2}
Sequence-to-fitness function learning rate	10^{-1}
Batch size	10
Number of epochs	150
Test set batch size	1
Number of ELBO particles	10
Total population	10^{11}
Sampled population	10^{8}
Learning rate scheduler	Plateau scheduler
Plateau patience	4

Table 3.1: Hyperparameters used for FLIGHTED-Selection.

To combine predictions from the guide model for different experiments in a single batch, we simply multiply the relevant Gaussians and sample the fitness from the resulting combined Gaussian distribution.

We used a learning rate of 10^{-2}, a learning rate on the landscape model (the sequence-to-fitness function) of 10^{-1}, a batch size of 10, 150 epochs, 10 particles in the ELBO loss function in Pyro, and a plateau learning rate scheduler with a patience of 4 epochs. Other hyperparameters can be found in the released code and in Table 3.1. The model was trained on 80 of the 100 simulations, validated on another 10 (randomly split), and tested on the final 10.

3.3.1.3 Model Evaluation

Models were evaluated on the final 10 held-out simulations, as well as on a number of other simulations with different parameters. Since the model was simply predicting the fitness mean as a function of the enrichment ratio, we focused on evaluating model uncertainty via computing the log likelihood. To do this, the fitness means m_i and variances σ_i predicted by the model for each variant i were first converted back to probabilities p'_i using the sigmoid function σ with appropriate clamping. These predicted probabilities p'_i could be arbitrarily linearly related to the ground-truth probabilities p_i, so we ran a linear regression $p_i = ap'_i + b$ to relate them, filtering out any variants that were not observed pre-selection. Data leakage is not an issue here since we already know that the enrichment ratio is a linearly scaled version of the ground-truth probability.

Given this regression, we could then convert the true ground-truth probabilities p_i into sigmoid space by taking logit $\left(\frac{p_i - a}{b}\right)$, with clamping at $\epsilon = 0.001$ to avoid issues near boundaries. The z-score could then be computed by comparing this true transformed probability to the distribution $\mathcal{N}(m_i, \sigma_i)$ predicted by the model. These z-scores are plotted in Figure 3.3e and should approximate a normal distribution, so we could also compute the mean log likelihood across the dataset using them. We report the mean log likelihood here as our primary performance metric for the model.

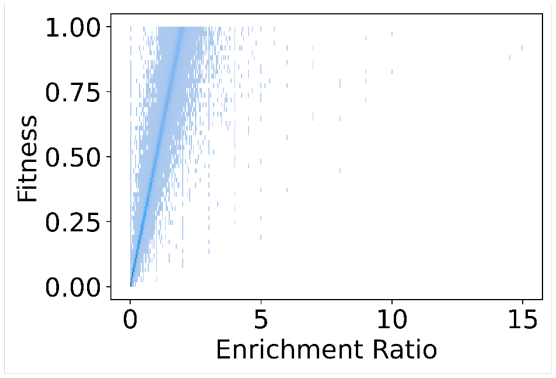
3.3.2 Results

3.3.2.1 Noise in Single-Step Selection

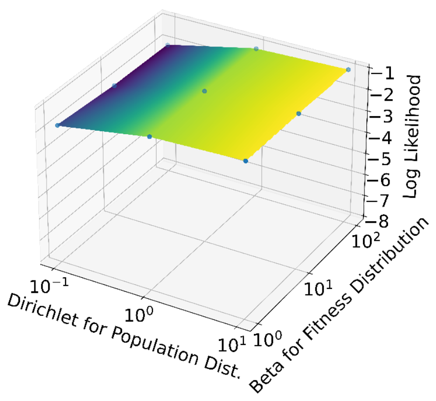
To understand the impacts of sampling noise and generate calibration datasets for downstream training, we ran simulations of a single-step selection assay on a randomly generated fitness landscape with 20^4 variants. We use simulations here as a substitute for real experimental data, since we need to know the ground-truth fitness to accurately evaluate FLIGHTED-Selection's performance; there is no easy way to measure the ground-truth fitness for any given real-world single-step selection experiment, and alternative measurement assays are generally not high-throughput and consequently fail to yield sufficient data. Figure 3.3c and 3.4a plot the probability distributions of the enrichment ratio for a given fitness, showing there is considerable noise and that this noise increases with increasing fitness.

To directly show the extent of the impact on downstream ML, we measure the Pearson r between fitness and enrichment ratio. Figure 3.4c demonstrates that the Pearson r for the roughly 1000 variants with highest enrichment ratio is roughly 0, indicating single-step selection experiments offer no useful information about variants with the highest fitness. Many papers [265] filter their data by using a minimum number of reads to reduce noise. We simulate this practice in Figure 3.4c and find no substantial change. This is particularly problematic for downstream ML

(a)



(b)



(c)

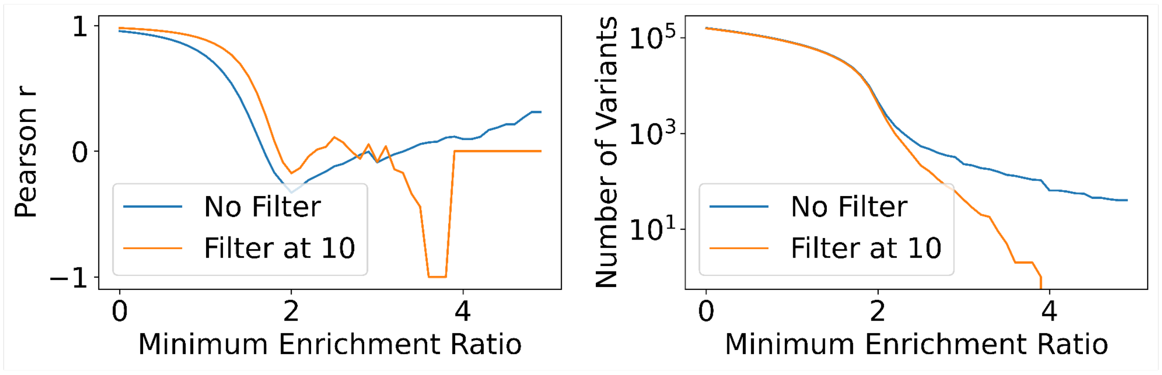


Figure 3.4: Noise in Single-Step Selection Simulations and Robustness of FLIGHTED-Selection Model Performance. (a) Another perspective on the noise, showing significant noise in the enrichment ratio. (b) The beta distribution is used to determine the distribution of fitness values and the Dirichlet distribution is used to determine the distribution of initial populations. We see that these parameters in general have minimal effect on FLIGHTED-Selection log likelihood, but with very low Dirichlet parameters there is a decrease in performance. (c) Left: the x-axis is the minimum enrichment ratio and the y-axis is the Pearson r between enrichment ratio and fitness for all data points above that minimum. The drastic drop-off in Pearson r as minimum enrichment ratio increases shows that high-activity variants essentially provide 0 information from just their enrichment ratio. Right: y-axis is the number of variants, showing that this is still a significant number of variants. We also tried filtering at a minimum of 10 reads, which improves results but still shows significant noise.

applications attempting to select variants with high fitness, since the most useful data points for this task are also the noisiest. Therefore, enrichment ratio is not a reliable measure of fitness.

3.3.2.2 Model Performance and Robustness

Typical model predictions are shown in Figure 3.3d for given enrichment ratios; for more active variants, single-step selection experiments produce less reliable fitness measurements and FLIGHTED-Selection assigns higher variance, as expected. To evaluate model performance, we focused on calibration, since the enrichment ratio is the optimal fitness value. We compute the z-value of the true fitness compared to the model prediction in Figure 3.3e; the resulting distribution looks very similar to a normal distribution, suggesting that FLIGHTED-Selection is well-calibrated. The mean log likelihood of our predictions is -0.97, very close to that expected for a normal distribution.

Next, we evaluated the robustness of FLIGHTED-Selection as a function of various parameters of the data generation process. In Figure 3.3f we showed the log likelihood as a function of number of reads sampled pre- and post-selection and demonstrated that the model was very reliable as long as the number of reads sampled was greater than the number of variants (at 20^4). Next, we fix the number of reads sampled pre- and post-selection and evaluate robustness as a function of number of variants. In Figure 3.3f, we see that the log likelihood of FLIGHTED-Selection does not substantially worsen as a function of the number of variants. We also experimented with varying different parameters of the random distributions in the simulation to see whether that affected FLIGHTED-Selection model performance. In Figure 3.4b, we see that these factors generally have very little effect on robustness. As a result, our use of simulated data to train FLIGHTED-Selection should not substantially affect real-world model performance.

3.3.2.3 Bootstrapping: A Simpler Alternative

One more traditional alternative to FLIGHTED-Selection is bootstrapping [284], a nonparametric approach where the dataset is repeatedly resampled to compute an error around a given estimator. Bootstrapping is not a replacement for FLIGHTED in general, since FLIGHTED can deal with more general cases where the fitness mean is not known. However, FLIGHTED-Selection is a simpler case where the fitness mean is known to be a function of the enrichment ratio, so bootstrapping

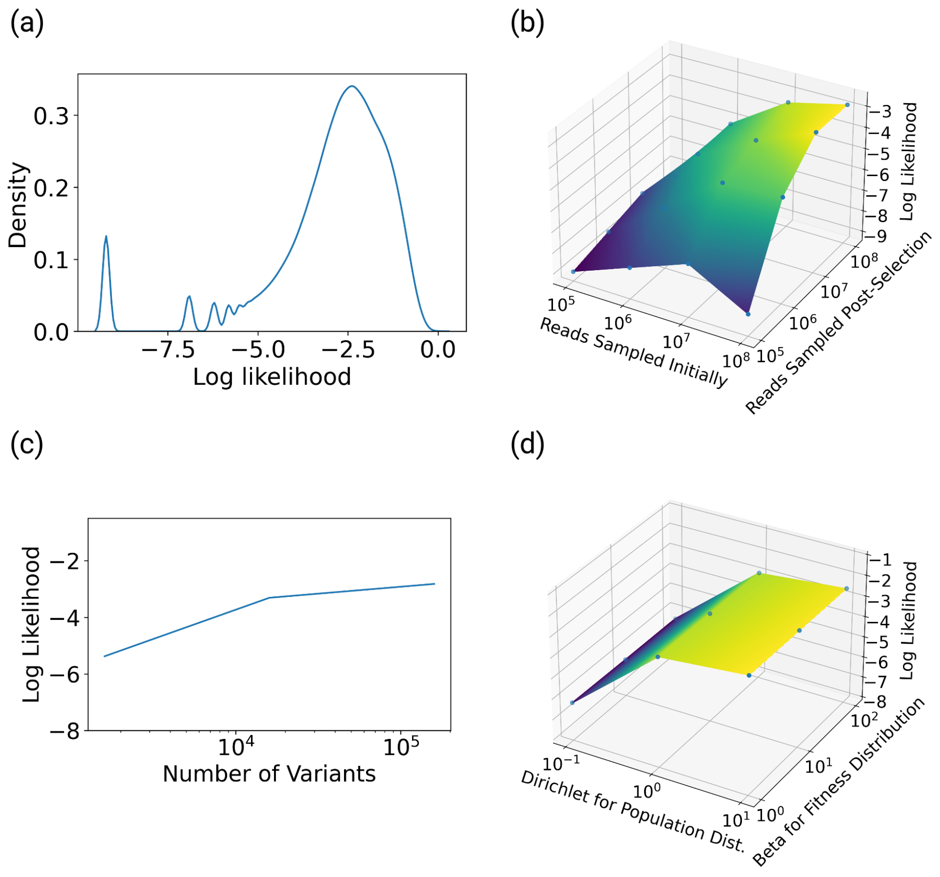


Figure 3.5: Bootstrapping as an Alternative to FLIGHTED-Selection. (a) Histogram of the log likelihood of each variant as predicted by bootstrapping. (b-d) Robustness of bootstrapping as a function of (b) number of reads selected pre- and post-selection, (c) number of variants, and (d) parameters used to simulate the population and fitness distribution. Bootstrapping yields poorer performance than FLIGHTED-Selection and is substantially less robust, especially to smaller sample sizes and to skewed population distributions.

is a potentially viable alternative. It has a few potential advantages over FLIGHTED-Selection: it is conceptually simpler, does not require training on a simulation-based dataset, and can be used to represent nonparametric distributions so it does not need to assume that the posterior distribution over fitnesses is normal.

Suppose we have a single-step selection experiment with known pre- and post-selection read counts $N_{\text{sampled, init},i}$ and $N_{\text{sampled, final},i}$. For each bootstrap sample, we compute

$$N'_{\text{sampled, init},i} \sim \text{Mult}\left(N_{\text{sampled, init}}; \frac{N_{\text{sampled, init},1}}{N_{\text{sampled, init}}}, \ldots, \frac{N_{\text{sampled, init},N_{\text{var}}}}{N_{\text{init}}}\right)$$

and similarly

$$N'_{\text{sampled, final},i} \sim \text{Mult}\left(N_{\text{sampled, final}}; \frac{N_{\text{sampled, final},1}}{N_{\text{sampled, final}}}, \ldots, \frac{N_{\text{sampled, final},N_{\text{var}}}}{N_{\text{final}}}\right).$$

The distribution of the resampled enrichment ratio

$$\frac{N'_{\text{sampled, final},i}/N_{\text{sampled, final}}}{N'_{\text{sampled, init},i}/N_{\text{sampled, init}}}$$

now represents a nonparametric distribution over fitnesses. We drew $N_{\text{bootstrap}} = 1000$ samples per experiment.

In order to evaluate the performance of bootstrapping, as before we must relate the enrichment ratio to the ground-truth selection probability, which we do by a linear regression. We then approximate the probability distribution function of the resampled enrichment ratio by binning: we used bins of width 0.01 between 0 and 1, bins of width 0.1 between 1 and 10, bins of width 1 between 10 and 100, bins of width 10 between 100 and 1000, and bins of width 100 between 1000 and 10000. We then approximate the log likelihood using this binned probability distribution. Unfortunately, there were many situations in which the true selection probability fell in a bin with 0 resampled enrichment ratios; in that case, we assigned a log likelihood of $\log \frac{1}{10N_{\text{bootstrap}}}$ rather than the proper log likelihood of $-\infty$. This represents an obvious weakness of bootstrapping: in situations where a variant is not observed post-selection, bootstrapping will always predict a fitness of 0 with total confidence, even though this is completely unrealistic. We observed around 5000 variants with this problem in simulation.

In Figure 3.5a, we plot the log likelihood of each variant as predicted by bootstrapping. The mean log likelihood is -2.9, substantially worse than that of FLIGHTED-Selection. There is a peak at around -9 due to variants with predicted 0, but even ignoring this peak, the overall performance of bootstrapping is substantially worse. The advantage of being able to model a nonparametric distribution does not make bootstrapping any better at predicting the distribution of fitnesses due to the relatively small sample sizes of a number of variants.

We also examine the robustness of bootstrapping as a function of the same parameters in the data generation process. In Figure 3.5b, we see that the number of reads sampled has a very clear and dramatic effect on the log likelihood measured by bootstrapping. Compared to FLIGHTED-Selection, bootstrapping is much worse at dealing with low read counts. Similarly, in Figure 3.5c we see that the number of variants actually has an impact on the performance of bootstrapping, and in Figure 3.5d the Dirichlet parameter used in generating the initial population distribution also worsens bootstrapping's performance. Specifically, simulations with spiky initial population distributions are much worse for bootstrapping overall. Based on these results, we see that bootstrapping is a substantially worse approach than FLIGHTED-Selection to the problem of understanding noise in single-step selection experiments; it both underperforms in ideal conditions and is much less robust to changes in simulation parameters.

3.4 DHARMA

We next turn to DHARMA (Direct High-throughput Activity Recording and Measurement Assay), a recently developed high-throughput protein fitness assay [246]. DHARMA measures fitness by linking protein fitness to transcription of a base editor; this base editor is targeted to a canvas where it randomly causes C$\longrightarrow$T edits, as shown in Figure 3.6a. Higher fitness leads to higher base editor transcription and more C$\longrightarrow$T edits.

DHARMA can be run in high-throughput on up to 10^6 variants at once, since the readout is measured by cheap long-read sequencing. It can measure any protein function that can be linked to base editor activity, which includes protease activity, gene editing, and protein and DNA binding [198]. However, base editor production and activity are inherently stochastic processes

[285] so DHARMA output is noisy; Figure 3.6c demonstrates that repeated DHARMA experiments on the same variant can result in differing C→T edit counts. Therefore accounting for noise via an ML model like FLIGHTED is essential for reliable fitness measurements.

3.4.1 Methods

3.4.1.1 T7 Polymerase Dataset Generation

Work in this section was performed by my experimental collaborator Boqiang Tu.

We engineered a biological circuit to associate the enzymatic characteristics of T7 RNA polymerase variants with mutations accumulating on a designated DNA sequence, known as the canvas. T7 RNA polymerase is an enzyme responsible for transcribing DNA into RNA, and we constructed a library of variants with different recognition and activity profiles.

The transcription of the base editor, an enzyme that can induce mutations, is controlled by a T3 promoter. This promoter is selectively recognized by a subset of the T7 RNA polymerase variants. To moderate the expression levels of the T7 polymerase library and prevent rapid saturation of the canvas—which would compromise the collection of meaningful activity data—we used both a weak constitutive promoter and a weak ribosomal binding site.

We incorporated this biological circuit into cells already containing plasmids that express the single guide RNA (sgRNA), through a technique called electroporation. The sgRNA serves to guide the base editor to the specific canvas sequence where mutations are to be introduced. After electroporation, the cells were grown continuously in a bioreactor. Following this growth period, the region of the T7 polymerase library and the canvas with mutations was selectively amplified as contiguous fragments of DNA. The amplified material was then subjected to long-read Nanopore sequencing for detailed analysis of the mutations and activities of different T7 RNA polymerase variants.

To facilitate data analysis, we implemented a data processing pipeline. Its core function is to identify, tabulate, and assign mutations present in each sequencing read to the corresponding

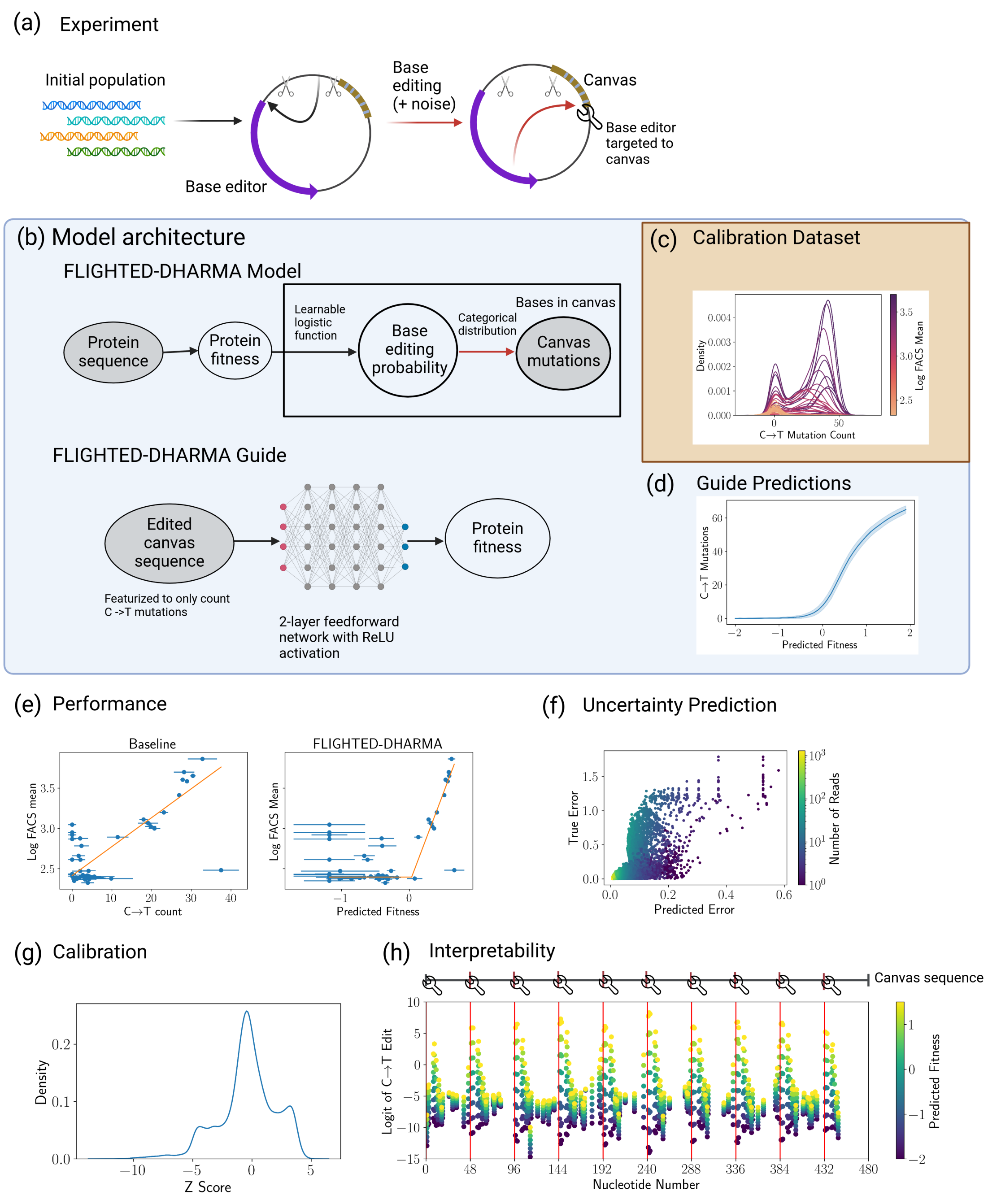


Figure 3.6: FLIGHTED-DHARMA Model Performance. (a) DHARMA links fitness to edit rates of a DNA segment (canvas) via transcriptional control of a base editor, an inherently noisy process. (b) The probabilistic graphical model for FLIGHTED-DHARMA and the architecture for the FLIGHTED-DHARMA guide. (c) Noise in DHARMA as a function of true fitness. (d) FLIGHTED-DHARMA's prediction of number of C⟶T mutations (cytosine-to-thymine transition introduced by the base editor) as a function of fitness. (e) FLIGHTED-DHARMA improves performance of fitness predictions when compared to a baseline model that predicts number of C⟶T mutations. (f) FLIGHTED-DHARMA errors correlated with true error on even small subsets of DHARMA reads. (g) FLIGHTED-DHARMA is reasonably well-calibrated. (h) Canvas nucleotides (nucleotides targeted by the guide RNA) most likely to be edited according to FLIGHTED-DHARMA are found at guide RNA binding sites, as expected.

library member. This assignment is based on internal barcodes represented as degenerate codons in the T7 RNA polymerase sequence. The pipeline accepts raw sequencing reads in standard genomic formats and performs length-based filtering and optional sequence trimming. An algorithm incorporating local sequence alignment is used for barcode recognition and classification during the demultiplexing of reads. Additionally, each read is aligned to the reference sequence of the canvas to identify the location of each C$\longrightarrow$T mutation, which is then stored in a matrix for downstream ML model training.

3.4.1.2 The FLIGHTED-DHARMA Model

The probabilistic graphical model for FLIGHTED-DHARMA is shown in Figure 3.6b. Given a fitness landscape, we compute a base editing probability for each base in the canvas as a logistic function of fitness and then sample to determine whether each base was mutated. This model treats all canvas bases as independent but allows each base to have a different learned edit propensity. The guide predicts both the mean and variance of fitness given a single DHARMA read using a two-layer feedforward neural network. We generated a calibration dataset by running a DHARMA experiment on a 3-site library of T7 polymerase. Concretely, in Figure 3.6d we show the predicted number of C$\longrightarrow$T edits with error given a particular fitness.

The sequence-to-fitness function is simply a dictionary lookup, with each variant i mapped to a fitness mean m_i initially sampled from $\mathcal{N}(0,1)$ and variance σ_i sampled from Softplus($\mathcal{N}(0,1)$), where the softplus function is used to ensure that the variance is positive. Both m_i and σ_i are learnable. The fitness z_i of each variant is sampled $\sim \mathcal{N}(m_i, \sigma_i)$ and fitness values are clamped at -2.

For each position j in the canvas, we set a learnable parameter r_j for a baseline rate of generic mutations (which accounts for most sequencing error from the long-read sequencing). For positions that are cytosines, we set a learnable parameter m_j for the fitness-dependent slope and b_j for the intercept. Then for all cytosines, given a fitness z of the variant, the logit of the C$\longrightarrow$T edit is set to $m_j z + b_j$ and the logit of any other mutation or deletion is r_j. For non-cytosine residues, we

Hyperparameter	Value
Learning rate	10^{-4}
Sequence-to-fitness function learning rate	10^{-2}
Batch size	2
Number of epochs	25
Test set batch size	10
Number of ELBO particles	1
Number of layers in guide	2
Guide hidden dimension	10
Guide activation	RELU
Learning rate scheduler	Plateau scheduler, on training loss
Plateau patience	1
Plateau learning rate factor	0.1

Table 3.2: Hyperparameters used for FLIGHTED-DHARMA.

simply have a logit of any mutation set to r_j. We sample from the one-hot categorical distribution for each residue independently to get the output DHARMA read, i.e. from $\mathrm{Cat}((1, m_j z + b_j, r_j))$ for cytosine residues and $\mathrm{Cat}((1, -\infty, r_j))$ for non-cytosine residues.

The guide predicts fitness mean and variance from a single DHARMA read. We used a feedforward network with 2 layers with a hidden dimension of 10 and a ReLU activation between the two layers. To simplify the learning problem, only cytosine residues were fed into the guide and each position was featurized with a one-hot encoding of two classes: a C$\rightarrow$T edit or a non-C$\rightarrow$T edit (including both no mutation and other mutations or deletions). Variances were transformed under the softplus function. Fitness values of variants with multiple reads can be predicted by multiplying the appropriate predicted Gaussians of the individual read.

The ELBO loss used 1 ELBO particle. The landscape model (sequence-to-fitness function) learning rate was 10^{-2}, the learning rate elsewhere was 10^{-4}, and the batch size was 2. We used a plateau learning rate scheduler with a patience of 1 epoch and a factor of 0.1, stepped based on the training loss (not validation loss). The model was trained for 25 epochs. Other hyperparameters are found in Table 3.2. 10% of the reads were held out as a validation set to pick the optimal epoch for the model.

3.4.1.3 Model Tuning and Evaluation

"Ground-truth" fitness measurements were produced by Fluorescence-Activated Cell Sorting (FACS) on a subset of 119 T7 variants, in which T7 polymerase drives the expression of GFP instead of base editor. Error was minimized in these FACS measurements by using a large number of cells per variant. To eliminate data leakage, FLIGHTED-DHARMA was not trained on any DHARMA read from any variant for which FACS was performed.

In order to assess model performance using the FACS dataset, we must convert fitness values predicted by FLIGHTED-DHARMA to the FACS readout; since fitness is scaled arbitrarily, these two fitness values can be related by any nondecreasing function. We used a validation set to fit this function to a piecewise linear function as shown in Figure 3.7a. We did not use Spearman correlation so we could measure true error (as opposed to rank-order error), and consequently we could account for situations where the function relating the FACS data to predicted fitness was not increasing (as happened here). For this, we split out 50% of the FACS data to use as a fitness regression/validation set and evaluated the mean of the logarithm of all FACS samples. Each model was given the option of fitting using either a piecewise linear function or a linear regression. The linear regression was fit normally. The piecewise linear function corresponded to the functional form

$$f(x) = \begin{cases} a & x \leq x_c \\ m(x - x_c) + a & x > x_c. \end{cases}$$

This ensured that for low fitness values, the predicted fluorescence was constant, as would be expected for background fluorescence dominating the FACS measurement on low-activity variants. We then used orthogonal distance regression, as implemented in `scipy`, to fit the piecewise linear function accounting for errors in both predicted fitness and the FACS data [286, 287]. Between the linear regression and piecewise linear fit, the function with the lower MSE on the validation data was selected.

This validation set MSE was also used for selecting hyperparameters via a grid search, leaving the remaining 50% of the FACS data as a test set used solely for evaluating model performance as shown in Figure 3.6e. Hyperparameters tuned included the batch size, learning rate, learning rate scheduler step, and number of layers in the guide model. We also evaluated the optimal number of hours to grow the cells in the bioreactor, as an example of an experimental parameter that can be tuned using FLIGHTED.

Model performance was measured with the 50% held-out test set of the FACS data, as described above, using the fitness values predicted by the guide model in FLIGHTED-DHARMA and the fitness-to-fluorescence function fit on the validation set. The results of that performance evaluation are shown in Figure 3.6e.

We then evaluated calibration of the model. To increase the number of data points (since we only had 119 variants measured through FACS) and to evaluate model performance with very small numbers of reads, we subsampled subsets of reads for each variant. Specifically, for all test set variants, we sampled 10 subsets each of sizes ranging from 1 to the maximum possible number of reads. We then predicted the fitness of each variant using the given subset of reads with the guide model. We computed the true fitness using the FACS data, eliminating any data points where the true fitness was below the baseline of the piecewise linear function. We then computed the z-score by comparing this predicted and true fitness. This generates the plots shown in Figure 3.6f and 3.6g.

3.4.2 Results

We now assess performance and uncertainty of the trained FLIGHTED-DHARMA model. FLIGHTED-DHARMA was compared to a baseline consisting of the mean and variance of C$\rightarrow$T mutation count. We found that FLIGHTED-DHARMA's mean-squared-error (MSE) was 0.72, an improvement over the baseline MSE of 0.78. Most of the improvement is observed on the portion of the fitness landscape where both DHARMA and FACS are measuring fitness.

We also examined uncertainty by evaluating the calibration of our predicted variances. First, we directly examine calibration on the test data, using all reads available for each datapoint in the

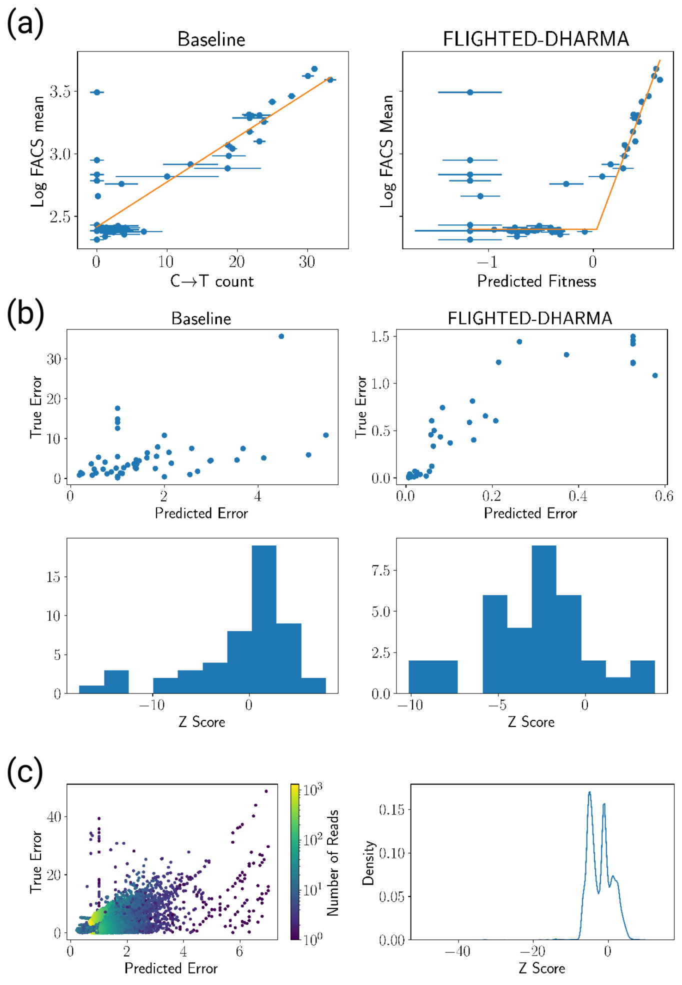


Figure 3.7: FLIGHTED-DHARMA Model Performance. (a) Accuracy on Validation set for (left) Baseline and (right) FLIGHTED-DHARMA. This shows the fit between the fitness values predicted by each model and the log FACS mean, as done by either a piecewise linear function or a linear function. (b) Calibration of FLIGHTED-DHARMA as Compared to Baseline. The baseline model's true errors are not related to the predicted error, while FLIGHTED-DHARMA has higher true error when predicted error is higher. As such, FLIGHTED-DHARMA is considerably better calibrated with many fewer z-score outliers. (c) Calibration of Baseline C⟶T Model. The baseline model's calibration is considerably worse compared to FLIGHTED-DHARMA's in Figures 3.6f and g.

test set in Figure 3.7b. FLIGHTED-DHARMA's predicted errors increase as true error increases. The log likelihood of the baseline model on this dataset was −19.8 while the log likelihood of FLIGHTED-DHARMA was −10.0. In Figure 3.6d, we selected random subsets of DHARMA reads and measured the true and predicted error of FLIGHTED-DHARMA. Random subsampling tests our ability to predict error given even very small subsets of DHARMA sequencing reads. The mean log likelihood here was −3.93, suggesting that FLIGHTED-DHARMA is slightly overconfident but reasonably well-calibrated. Baseline model performance was substantially worse with a mean log likelihood of −8.00 (see Figure 3.7c). Since our calibration tests included small subsets of DHARMA reads, we can be confident that FLIGHTED-DHARMA will produce reasonable errors even when given few DHARMA reads. This aids experimentalists by informing them whether a given number of DHARMA reads is sufficient for a fitness measurement.

Finally, since FLIGHTED uses a biological model of DHARMA, we can examine the parameters of that biological model directly. In Figure 3.6g, we compute the logit of the C⟶T edit probability at a given canvas residue as a function of fitness. The probability of a C⟶T edit increases and is more fitness-dependent every 48 residues. The base editor is targeted at these locations, so these results are expected. This is an example of how FLIGHTED can yield an interpretable model.

3.5 Discussion

FLIGHTED has consistently improved model performance by generating fitness landscapes with robust, calibrated errors for two very different high-throughput experiments. Since it is based on Bayesian modeling and variational inference, it can be easily extended to other high-throughput experiments or more complicated noise generation models of single-step selection by taking into account effects such as biased library design or sequencing error. We have demonstrated that FLIGHTED models can be used reliably under a wide range of experimental conditions and offer a modest performance improvement in fitness prediction. These results are improvements over baselines like bootstrapping for single-step selection experiments and the C⟶T count for DHARMA experiments. FLIGHTED results can also be used to combine fitness readouts from different experiments and optimize experimental design by maximizing signal-to-noise.

FLIGHTED-DHARMA improves the use of DHARMA as a data generation method for machine learning. The noisy nature of biological circuits presents challenges in effectively using DHARMA to quantify activities. We provide calibrated error estimates that inform practitioners when they have enough DHARMA reads to make a reliable fitness estimate. DHARMA can be used to cheaply generate large datasets of up to 10^6 variants using Nanopore sequencing alone. Comparing FLIGHTED predictions for DHARMA and single-step selection, we see that DHARMA may be more accurate for distinguishing between highly active variants while selection experiments may be more accurate at identifying the fitness values of comparatively inactive variants.

Chapter 4

Benchmarking Protein Fitness Models on Large Landscapes

Some of the contents of this chapter were previously published in Sundar et al. [269], Sundar et al. [271], Sundar et al. [270], and Tu, Sundar, and Esvelt [246].

4.1 Introduction

Modeling protein fitness landscapes is essential for applications in protein design and drug discovery. In particular, modeling the fitness landscape locally in the neighborhood of a wild-type protein helps us optimize or slightly modify the function of that specific protein, like that of TEV protease. As discussed previously, perhaps the most popular approach to this problem is to use embeddings from a protein language model which are then fed into task-specific models, also known as top models [61, 95, 99–101, 288]. Within this space, there are a vast number of choices to be made: protein language model choice and scale, top model architecture, experimental data size, library design, and more. We aim to tackle these questions in this chapter, focusing primarily on modeling a single landscape due to our experimental limitations.

We begin by focusing on 4-site FLIGHTED-calibrated landscapes: a 4-site landscape for the GB1 protein that measures stability and binding to IgG via an mRNA display single-step selection assay

[265] and a 4-site landscape on TEV protease that measures protease activity on the wild-type substrate using a DHARMA assay. The first landscape has frequently been used to benchmark ML models in the field [266]; we release the second landscape publicly as a source for the ML community. We begin by benchmarking a number of standard ML protein fitness models on these landscapes to answer the questions listed above. Our results show that using FLIGHTED improves model performance generally, especially for models that perform well, and changes relative rankings. Further, experimental data size is the most limiting factor when determining the performance of protein fitness models. On the modeling side, we find that convolutional neural networks are the best top model given large data scales and that protein language model size does not particularly affect model performance.

We then turn to larger landscapes on TEV protease, continuing to focus on the wild-type substrate; we generate an 8-site combinatorial landscape and an error-prone PCR landscape including millions of variants. Here, we use the model architectures previously identified as optimal and focus on optimizing experimental dataset design. We find that error-prone PCR datasets with at least triple mutants are optimal, and there is no data plateau: performance continues to increase with increasing data scale, though we do observe diminishing returns. We also find that different metrics of model performance can tell different stories about which models are optimal; it is very important to pick metrics that closely align with the problems we want to solve.

4.2 Protein Fitness Models

The models we chose to benchmark are outlined in Figure 4.2a. Specifically, given a sequence, we generate an embedding from a protein language model or from a one-hot baseline [66, 75]. We then feed this embedding into a task-specific top model – either a linear layer, an augmented model [289], a feedforward neural network (FNN), a convolutional neural network (CNN), or a fine-tuned FNN or CNN – which is trained on this particular landscape. This strategy of blending a protein language model with a task-specific top model has repeatedly proven successful [66, 75, 115, 192, 266]. While these models are certainly not novel, to our knowledge this is the first systematic exploration of the use of CNNs as a top model. See Table 4.1 for a list of hyperparameters.

The linear regression model (labeled Linear) takes one-hot embeddings of the full sequence and runs them through 1 linear layer. It uses an Adam optimizer with a learning rate of 10^{-2}, a batch size of 256, and a weight decay of 1. The CNN model (labeled CNN) takes one-hot embeddings of the full sequence and has 1 convolutional layer with 1024 channels and filter size 5, and same padding. It then has a 1D batch normalization layer and a ReLU activation, followed by an embedding neural network consisting of a linear layer to 2048 dimensions and a ReLU activation. Then there is a max-pooling layer over residues, a dropout layer with probability 0.2, and a final linear layer for the output. The CNN is trained with a batch size of 256 and an Adam optimizer with a learning rate of 10^{-3} and weight decay of 0 for the convolutional layer, a learning rate of $5 * 10^{-5}$ and weight decay of 0.05 for the embedding layer, and a learning rate of $5 * 10^{-6}$ and weight decay of 0.05 for the output layer. Unlike the original FLIP paper, we did not use early stopping and trained all models for a full 500 epochs, using validation set performance to select the optimal model.

The models labeled TAPE, ESM-1b, ESM-1v, ESM-2, ProtT5, and CARP all use mean embeddings across the entire sequence that are fed into a feedforward neural network to compute the output [60, 65, 66, 75, 290]. The output feedforward neural network has 2 layers with a hidden dimension equal to the embedding dimension of the protein language model and ReLU activation. All models are trained with an Adam optimizer with batch size 256 and learning rate 10^{-3}. The TAPE model used was the transformer, the ESM-1v model used was version 1, the largest ESM-2 model used was the 3 billion parameter version (due to memory issues with the larger model), the ProtT5 model was the suggested ProtT5-XL-UniRef50 model, and the CARP model was the largest 640 million parameter version. For Figure 4.2d, we ran smaller versions of CARP and the ESM-2 models, as specified by their parameter number, with the same architectures as described above [75, 291].

Linear

Hyperparameter	Value
Learning rate	10^{-2}
Batch size	256
Weight decay	1
Number of epochs	500

FNNs

Hyperparameter	Value
Learning rate	10^{-3}
Hidden dimension	Embedding dimension
Activation function	ReLU
Batch size	256
Weight decay	1
Number of epochs	500

CNNs

Hyperparameter	Value
Convolutional layer learning rate	10^{-3}
Embedding layer learning rate	$5 * 10^{-5}$
Output layer learning rate	$5 * 10^{-6}$
Convolutional layer weight decay	0
Embedding layer weight decay	0.05
Output layer weight decay	0.05
Filter size	5
Number of channels	Embedding dimension
Intermediate dimension	2*embedding dimension
Activation function	ReLU
Batch size	256
Number of epochs	500

Table 4.1: Hyperparameters used for benchmarking models on fitness datasets.

The models labeled TAPE (CNN), ESM-1b (CNN), ESM-1v (CNN), ESM-2 (CNN), ProtT5 (CNN), and CARP (CNN) use residue-level embeddings that are fed into a CNN, similar to the baseline CNN described above. The intermediate dimension output by the embedding neural network is set to be twice the dimension of the output of the protein language model. All other parameters remained the same.

The models labeled ESM-1v (Augmented), ESM-2 (Augmented), CARP (Augmented), and EVMutation (Augmented) used the zero-shot variant effect prediction from the model in question, combined with a one-hot encoding of the entire sequence that was fed into a linear layer [66, 75, 291, 292]. These were trained with the same parameters as the baseline linear model. The ESM models used the masked marginal approach proposed in [66] to compute zero-shot variant effect prediction. EVMutation used the default parameters [292].

Fine-tuned models are trained as described above, but we also fine-tuned the underlying ESM model that generated the embeddings with a learning rate of 10^{-6}. Due to compute and cost limitations, fine-tuning was only done for the ESM models with 8 and 35 million parameters for the TEV dataset and 8, 35, and 150 million parameters for the GB1 dataset.

We observed high run-to-run variance in the performance of many models. As a result, all models were run in triplicate and we have reported both the mean and standard deviation of model performance.

4.3 4-Site Landscapes

4.3.1 The GB1 Dataset

To compute the corrected GB1 landscape with FLIGHTED-Selection, we took the guide and ran inference on the released GB1 landscape data, which provided both pre- and post-selection read counts in addition to the enrichment ratio [265]. We omitted all data points that were omitted in the original study due to not being observed. We then followed the published data splits provided by FLIP [266] to generate datasets both with and without FLIGHTED-Selection for the

GB1 problem; here, one-vs-rest refers to just single mutants in the training set, two-vs-rest refers to just double mutants in the training set, three-vs-rest refers to just triple mutants in the training set, and low-vs-high refers to low-activity variants in the training set. Models trained on datasets with corrections from FLIGHTED-Selection are trained with an MSE loss weighted by the inverse variance, i.e.

$$\mathcal{L} = \sum_{\text{dataset}} \frac{1}{\sigma^2}(y - \hat{y})^2$$

where y is the true value, $\hat{y}$ is the prediction, and σ^2 is the variance. Weighted MSE is used to account for the variance, assuming that the likelihood of the data is the same as that of a normal distribution with the provided variance. Performance results are regular MSEs for the datasets without corrections and weighted MSEs for datasets with corrections. Weighted Spearman ρ measurements, used for comparing results with and without FLIGHTED-selection, were made with 100 Monte Carlo replicates [293].

4.3.2 The 4-Site TEV Dataset

Work in this section was performed by my experimental collaborator Boqiang Tu.

High-throughput quantification of TEV protease fitness using DHARMA was made possible by a biological circuit that couples the activity of TEV protease to base editor transcription driven by a T7 promoter. As part of the circuit, the T7 RNA polymerase, normally repressed by its natural inhibitor T7 lysozyme, becomes functional after the inhibitor is disabled by active TEV protease variants, thus allowing the transcription of base editor to proceed, which in turn introduces mutations to the canvas. We engineered a variant of T7 lysozyme in which the TEV protease substrate sequence is inserted in the middle of the coding sequence. This modified T7 lysozyme, expressed on a separate plasmid under the control of a medium constitutive promoter, together with the T7 lysozyme tethered to the T7 RNAP, provides an enhanced dynamic range of base editing coupled to the activity of TEV protease variants.

The library of TEV protease used in this study was obtained by performing site-saturation mutagenesis on 4 amino acid residues (T146, D148, H167, and S170) in the TEV protease S1 pocket, which is known to interact with P1 residue on the substrate and determine substrate specificity. Briefly, NNK degenerate primers were used to introduce mutations at the targeted residues in a PCR reaction. The pool of amplicons was then cloned into a Golden Gate vector comprising the rest of the TEV protease expression cassette, the sgRNA and the canvas sequence. Commercial electrocompetent cells were then transformed with the cloning reaction, selected with appropriate antibiotics on agar plate overnight and subjected to DNA extraction. The resulting library was then sequenced to assess for bias and coverage. After quality control, the TEV protease library was transformed into electrocompetent cells that already express the base editor, the T7 RNAP and the engineered T7 lysozyme. The transformants were then selected and grown continuously in a bioreactor. Multiple time points were taken during this growth period to find the optimal incubation time. The region of the plasmid containing TEV protease library and the canvas sequence was selectively amplified as contiguous fragments of DNA. To minimize the amplicon size, nucleotides not of interest were removed via self-circularization and re-amplification. The final amplified material was then subjected to long-read nanopore sequencing to simultaneously retrieve both variant identity and the mutations on the canvas sequence for each individual library member. Sequencing data processing and analysis was performed as described in the previous chapter.

The TEV landscape was generated by processing the raw TEV data with FLIGHTED-DHARMA as trained on the T7 dataset. Validation of these fitnesses as compared to an *in vitro* fluorescence assay can be found in Figure 4.1a. This *in vitro* assay included a fluorophore (5-FAM) linked to a quencher (QXL 520) via a peptide containing the wild-type TEV protease substrate. Selected TEV protease variants were inserted into a plasmid via Golden Gate cloning and transformed into electrocompetent cells. After cell growth and lysing, the desired protein was purified via column chromatography with amylose resin and used in the fluorescence assay. Given cleavage by TEV protease or an active variant, we expect the rate of fluorescence increase to be directly proportional to enzyme activity. We used enzyme concentrations of 50 nM, 33.33 nM and 22.22 nM with substrate concentrations of 500 nM and performed triplicate measurements. Fluorescence was recorded every 10 s for 30 min and reaction rates were obtained from a linear regression.

For the ML analysis, we split the dataset by mutation distance from the wild-type to create a one-vs-rest, two-vs-rest, and three-vs-rest data split as we did for the GB1 data. We then split out a random 10% of the training data to use as validation data. Models trained on the TEV dataset are trained with a weighted mean-squared-error (MSE) loss, weighted by the inverse variance, as above. Performance results are also weighted MSEs. In Figure 4.3a, we use a separate test set consisting of only quadruple mutants (i.e. the test set in the three-vs-rest split) for all models. Basic statistics about the TEV landscape may be found in Figures 4.1b - f. For Figure 4.2d, some models were trained on the C$\longrightarrow$T mean and variance instead as a baseline; training used the same inverse-variance weighted MSE loss as described above. Only baseline, FNNs, and CNNs were run with the C$\longrightarrow$T mean comparison to save computation time.

4.3.3 Results

See Tables B.1, B.2, B.3, and B.4 for raw GB1 model performance and Tables B.5 and B.6 for raw TEV model performance.

4.3.3.1 Impact of FLIGHTED

We first examine the impact of using FLIGHTED on models trained on the GB1 dataset in Figure 4.2b. The ladder plot (Figure 4.2b, left) shows that FLIGHTED significantly affects benchmarking outcomes. This is particularly important because of how frequently ML practitioners use single-step selection datasets to benchmark their models; without a method like FLIGHTED, the results of this benchmarking are unreliable.

Further, we also observe a general performance improvement upon using FLIGHTED data for the training set (see Figure 4.2b, right). Here performance is measured using both the Spearman ρ and the weighted Spearman ρ so results with and without FLIGHTED can be directly compared; the weighted Spearman ρ accounts for variance on the test set [293]. 37 of the 45 models on the

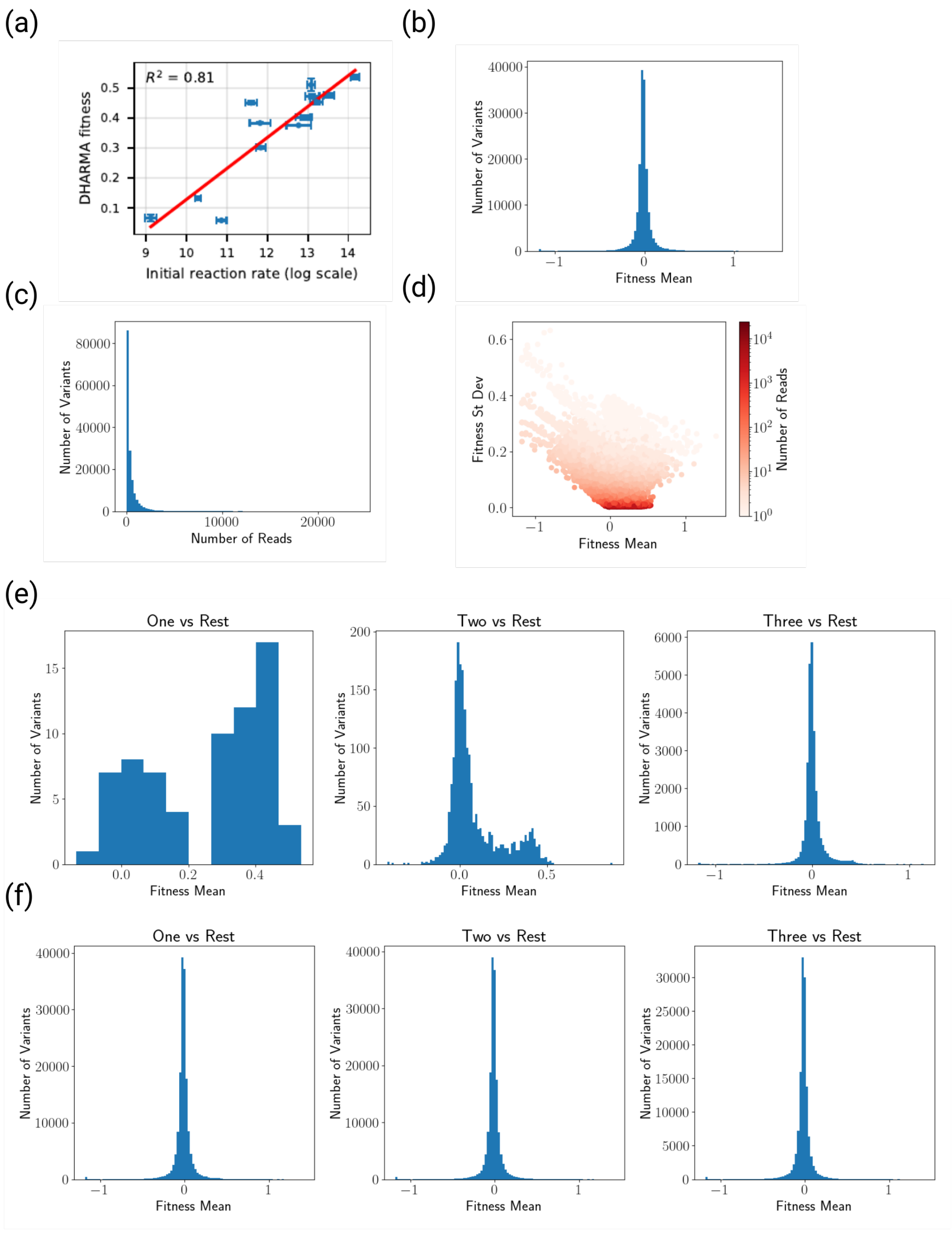


Figure 4.1: 4-Site TEV Landscape Statistics. (a) TEV protease fitnesses measured by DHARMA (direct high-throughput activity recording and measurement assay) and FLIGHTED compared to an *in vitro* fluorescence assay. (b) Distribution of fitness values in the TEV dataset. (c) Distribution of read counts (sequencing depth per variant) in the TEV dataset. (d) Fitness means, variances, and read counts in the TEV dataset. (e) Distribution of fitness values in the training sets for various splits of the TEV dataset. (f) Distribution of fitness values in the test sets for various splits of the TEV dataset.

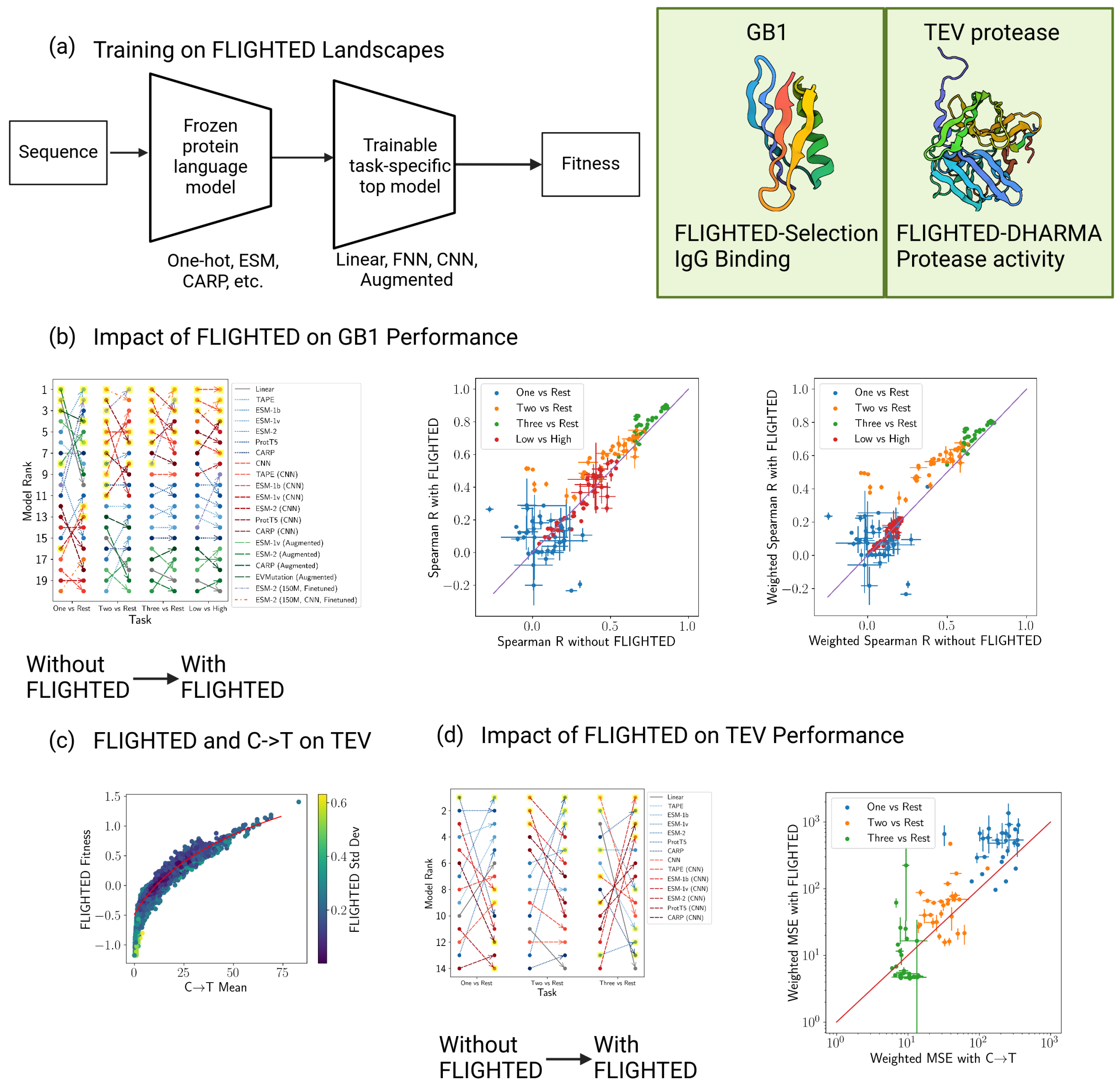


Figure 4.2: Impact of FLIGHTED on Protein Fitness Model Performance. (a) The typical model architecture we tested and the two landscapes: GB1, based on FLIGHTED-Selection [265], and TEV protease, based on FLIGHTED-DHARMA. (b) (left) FLIGHTED changes model ranking on the GB1 dataset, indicating the importance of accounting for noise when benchmarking ML models. FLIGHTED also modestly improves model performance on the GB1 dataset, as measured by weighted Spearman ρ (middle) or Spearman ρ (right) to the test set. (c) Fitness means predicted by C$\rightarrow$T count and by FLIGHTED correspond reasonably well, meaning a comparison between the two is somewhat appropriate. (d) (left) FLIGHTED changes model ranking on the TEV dataset, indicating the importance of accounting for noise when benchmarking ML models. (right) FLIGHTED substantially improves the performance of only the best models on the largest TEV datasets and often worsens the performance of worse models, suggesting that FLIGHTED is beneficial but only given sufficient data and sufficiently expressive models.

two-vs-rest split and 41 of the 45 models on the three-vs-rest split exhibited statistically significant improvements. We saw an average improvement of 0.15 on the two-vs-rest split and 0.05 on the three-vs-rest split. Results on the other two splits were not as consistent. We observed the most dramatic improvement with the augmented models on the two-vs-rest split and the most dramatic worsening of performance with the CARP/FNN models on the one-vs-rest split. The augmented models likely improved the most since they generally performed the most poorly (see the next section); it is unclear why CARP/FNN models suffered specifically when other protein language models were not nearly as strongly affected.

We now examine the impact of FLIGHTED on models trained on the TEV dataset. Evaluating this is more difficult, because FLIGHTED changes the fitnesses as compared to the C$\longrightarrow$T count baseline we have previously been using; we have already established in Figure 3.6e that this change improves performance. However, in Figure 4.2c we observe that the C$\longrightarrow$T mean and FLIGHTED fitnesses are still well-correlated; their relationship can be described by an exponential and has a Pearson r value of 0.95. We used this regression relationship to compare the predictions of models trained on C$\longrightarrow$T count with models trained on the FLIGHTED fitness.

In Figure 4.2d, we observe the results of this comparison. Here, weighted MSE is measured on the test set using the FLIGHTED variance as the weight, since the C$\longrightarrow$T variance is highly unreliable. As before, the ladder plot (Figure 4.2d, left) shows that FLIGHTED significantly affects benchmarking outcomes.

The performance results are shown in Figure 4.2d, right. The most important models in this figure are the high-performing CNNs (see the next section) on the three-vs-rest dataset, where FLIGHTED does indeed improve model performance and often by a substantial margin. However, for most other models on smaller datasets, FLIGHTED worsens model performance compared to the C$\longrightarrow$T baseline. Specifically, we observed an improvement in MSE of 3.82 for CNNs on the three-vs-rest dataset, an improvement of 1.85 for FNNs on the three-vs-rest dataset, and an improvement of 2.17 for FNNs on the two-vs-rest dataset, all substantial given how low the MSEs are for these models and datasets. On the other hand, essentially every model worsened with

FLIGHTED on the one-vs-rest dataset by an average of 55.2 and CNNs on the two-vs-rest dataset worsened by an average of 24.5. In other words, given an inadequate or insufficiently expressive model or an insufficiently large dataset, FLIGHTED can actually substantially worsen model performance, despite the FLIGHTED fitnesses being more accurate and better calibrated.

Overall, these results indicate that using FLIGHTED will generally improve model performance, though this is not guaranteed. We require sufficiently expressive models and sufficiently large datasets to unlock the full potential of improvement from FLIGHTED; otherwise we may see substantially worse results instead. Fortunately, for the rest of this thesis we will be focusing on large datasets and sufficiently expressive models where we expect FLIGHTED to improve overall performance.

4.3.3.2 Data Scale and Modeling Choices

Next, we examine the impact of various other parameters on model performance. In Figure 4.3a we look at the importance of data size on the TEV dataset, controlling the test set to include only quadruple mutants. As training data increases from single to double to triple mutants, model error decreases exponentially. This dramatic improvement in model accuracy indicates an increasing ability to generalize and highlights the importance of high-throughput experimental methods: with more data, our models become substantially more powerful.

We then examined which model architecture choices mattered most for performance. We found in Figure 4.3b that the top model was the most impactful, with all models equally unable to learn from single mutants, and CNNs performing better on larger datasets, especially three-vs-rest. Fine-tuning generally improves performance, especially for FNNs on small datasets, but is substantially more computationally expensive. Since CNNs performed well as a top model and are relatively computationally efficient compared to fine-tuning, we recommend practitioners use CNNs as a top model when they have large enough protein fitness datasets.

More surprisingly, increasing the number of parameters in the protein language model did not have a large impact on performance for most models. In Figure 4.3c, we ran published variants

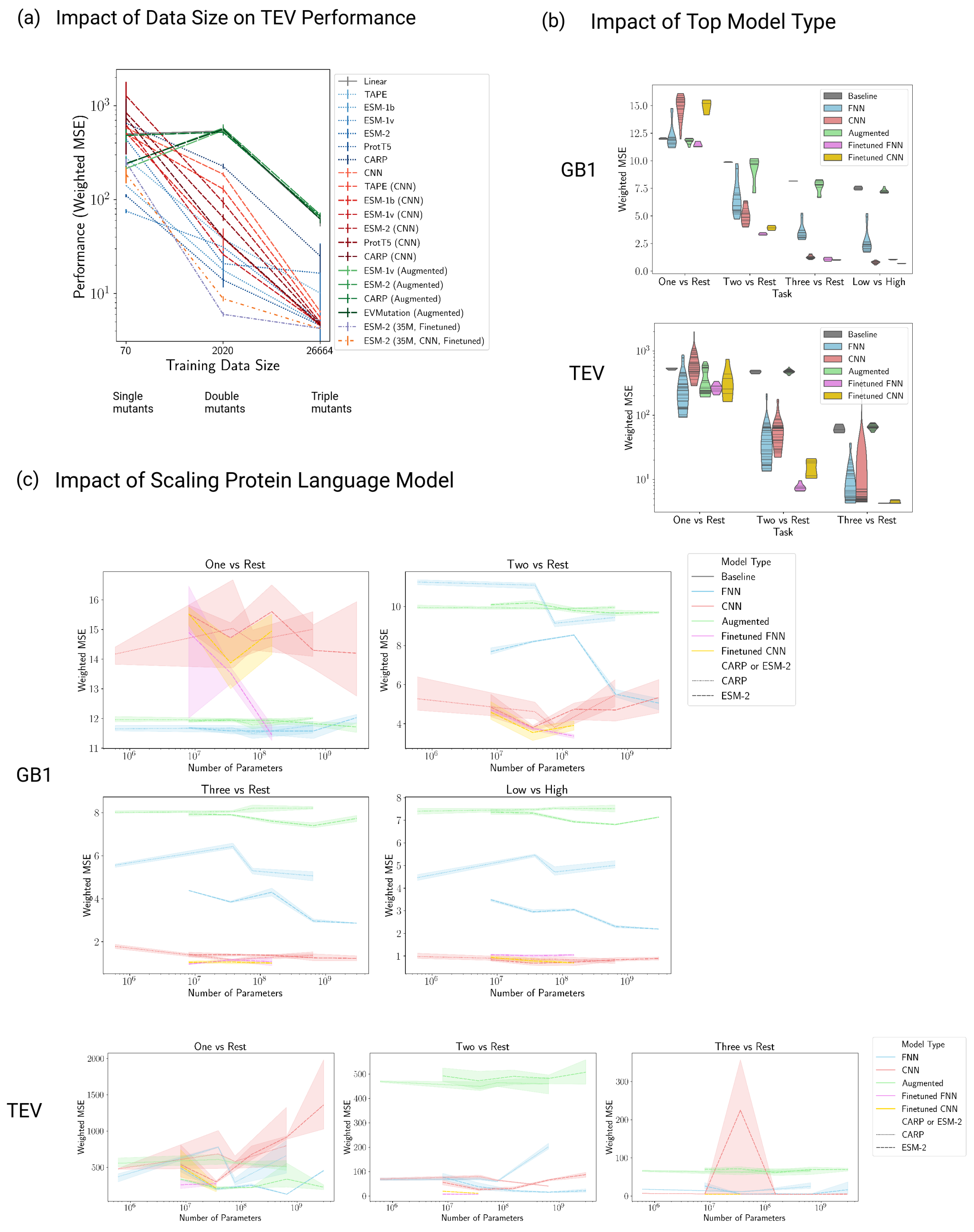


Figure 4.3: Benchmarking Protein Fitness Models on FLIGHTED Landscapes. (a) Exponential increases in data on the TEV dataset result in exponential increases in accuracy. (b) Convolutional neural networks (CNNs) and fine-tuned models are the highest-performing top models across both the GB1 and TEV datasets. (c) Scaling up protein language models does not substantially improve model performance.

of CARP and ESM-2 with smaller numbers of parameters [75, 291]. We found no substantial or consistent performance improvement, across models ranging from millions to billions of parameters, aside from fine-tuning FNNs on small datasets. Our results suggest that scaling up protein language models will not improve performance in predicting protein fitness given current data limitations. This finding has recently been corroborated in Li et al. [272]. It is worth noting that increasing the scale of the language model within a family like the ESM family generally also changes the dataset on which the language model is trained (usually based on the number of proteins per cluster sampled). TEV protease has a relatively shallow MSA, so it is possible that this increased training data did not help performance on TEV protease but would help performance on other proteins with deeper MSAs. It is worth disentangling the relationship between protein language model parameter size and the dataset the protein language model is trained on, but those investigations require training new models and would be substantially more computationally expensive. Based on our results, we selected ESM-2 embeddings (somewhat arbitrarily out of all the protein language model embeddings) and a CNN top model for further investigation.

4.4 Larger Landscapes and Dataset Design

We now turn to larger landscapes than the 4-site combinatorial landscape, focused specifically on TEV protease acting on the wild-type substrate. Analyzing model performance on different datasets will give us more insight into how to design our training data for TEV protease engineering. We will focus primarily on the CNN trained on ESM-2 8-million-parameter embeddings in this section, based on our model benchmarking results obtained previously, and compare to linear regression on one-hot encodings as a baseline. We ran a short toy experiment on the 8-site combinatorial dataset to confirm that model scale did not substantially improve performance, just like we saw on the 4-site landscapes.

4.4.1 The 8-Site TEV Dataset

4.4.1.1 Dataset Design

We first began by generating an 8-site TEV protease landscape, with 5 amino acids at each site. We first selected the sites through a literature search for previous studies that have explored TEV protease substrate specificity and engineering [13, 16]. The selected sites were I138, S153, N171, N176, N177, V209, W211, and M218; these are sites known to be involved in substrate specificity but not in the active site for TEV protease, so our variant proteases would likely maintain some activity.

We then used the ESM-2 650-million parameter model to guide selection of the amino acids. Specifically, we generated zero-shot predictions of fitness from the model and identified the 5 amino acids with the highest probability of occurring at each position. We used these 5 amino acids to generate the complete landscape. This strategy is similar to other training set design strategies that have been proposed; the goal is to prioritize amino acids that are more likely to leave the protein relatively functional [100]. The 8-site TEV dataset was generated by a similar procedure as the 4-site TEV dataset described above; generation of the experimental data was performed by my experimental collaborator Boqiang Tu.

4.4.1.2 Model Evaluation

With the generated datasets, we assigned fitnesses using the FLIGHTED-DHARMA guide as before. We then held out all variants 8 mutations away from the wild-type as a test set, and generated a number of training sets for the models. We generated one set of training sets including variants that were at most 1, 2, 3, 4, 5, 6, and 7 mutations away from the wild-type, and split out a random 10% of the training data to use as validation data. We also generated training datasets with a randomly selected 1%, 3%, 10%, and 30% of the 7-mutant dataset, and split out a validation dataset similarly. Models are trained with a weighted MSE loss as before. Basic statistics about this landscape may be found in Figures 4.4a-d.

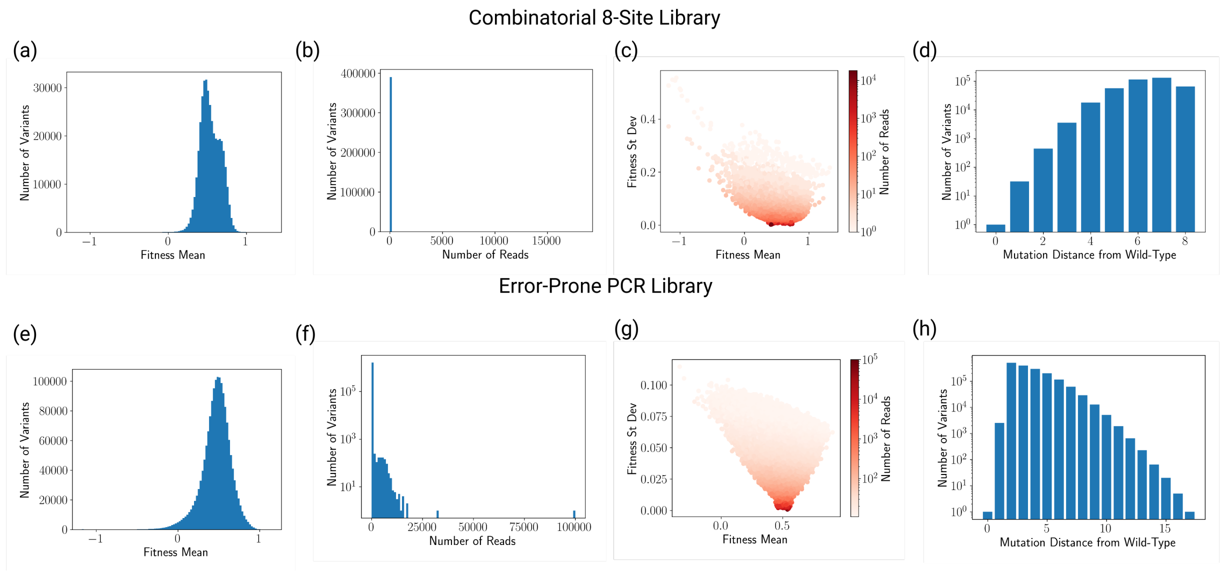


Figure 4.4: 8-Site and Error-Prone PCR TEV Landscape Statistics on the Wild-Type Substrate. (a-d) Statistics about the 8-site combinatorial landscape, including (a) fitness mean histogram, (b) read count histogram, (c) fitness mean and standard deviation, and (d) mutation distance histogram. (e-h) Statistics about the error-prone PCR landscape, including (e) fitness mean histogram, (f) read count histogram, (g) fitness mean and standard deviation, and (h) mutation distance histogram.

We evaluate models using 3 primary metrics. Inverse-variance weighted MSE is implemented as before and has until now been used as the primary performance metric. Active weighted MSE is an inverse-variance weighted MSE that looks only at the top 5% of variants in the dataset; this is a measure of our model's ability to predict the fitness of active variants. The mean fitness of top 100 variants, or mean top 100, is the true fitness of the test set variants predicted to be in the top 100 by the model; this is used as a proxy for performance in a protein design setting. We also explored the maximum fitness of the top 100 variants; we found that while it is closer to the desired objective of a protein design setting, it is considerably more random and therefore not used. The literature has also used the normalized discounted cumulative gain (NDCG) score to avoid picking a cutoff of exactly 100 [194], but we found all our models performed extremely well on this metric so it was not useful.

4.4.2 The Error-Prone PCR Dataset

4.4.2.1 Experimental Data Generation

As an alternative strategy to a designed combinatorial landscape, we also explored the use of error-prone PCR to generate a large number of mutations. Generation of the experimental data was performed by my experimental collaborator Boqiang Tu. Mutations were generated via a low-fidelity Taq polymerase aiming for around 6 to 9 nucleotide substitutions per kilobase. The same procedure was used to transform the TEV protease library into the DHARMA assay as discussed previously. Analysis of a subset of this library indicated a broad range of mutation distance from the wild-type from 1 to 7 mutations, as subsequently confirmed on the whole library in Figure 4.4h. The mutations were also located throughout the entire TEV protease sequence, as desired. Given these quality controls, the DHARMA assay was performed as previously described.

4.4.2.2 Fitness Variance for High Read Counts

Error-prone PCR experiments by their nature often produce variants with extremely high read counts, greater than 10^5 or 10^6. In this regime, we expect that experimental error not associated with base editing dominates error associated with base editing and accounted for by FLIGHTED, so FLIGHTED variance estimates may be substantial underestimates, by multiple orders of magnitude. FLIGHTED is also not tested for calibration on high read count variants due to limitations in the calibration experiment. As a result, variants with extremely high read counts may have unrealistically low fitness variances produced by FLIGHTED and models trained on such datasets may place correspondingly high weights on those data points. This potentially affects all the landscapes we have discussed thus far, but is most impactful on error-prone PCR datasets, where the variants with high read count are the wild-type variant and single mutants of the wild-type. Training a protein fitness model on this dataset without any adjustment would lead to a model that essentially only learned the single-mutant landscape and none of the other data.

We adjust for this by setting a minimum fitness variance cutoff across the entire dataset.

After hyperparameter tuning (detailed below) we found that a fitness variance cutoff of 10^{-3} was appropriate. Since we unfortunately only discovered this effect after already training models on all the datasets, we only retrained the error-prone PCR datasets with this minimum cutoff due to computational expense and report those results here. The combinatorial datasets also do have some variants with very high read counts, but those read counts are not as extreme and those variants are farther away from the wild-type and have greater diversity of fitnesses. As a result, we think the impact of a fitness variance cutoff on the models trained on the combinatorial datasets should be relatively minimal.

4.4.2.3 Model Evaluation

Like with the 8-site combinatorial dataset, we assigned fitnesses and generated training datasets with the same procedure. Due to the lack of internal barcodes or other means of mitigating Nanopore sequencing noise, we eliminated all variants with 1 or 2 reads, assuming those were sequencing errors, not the base editing noise modeled by FLIGHTED. The only difference was that the test set consisted of all variants at least 8 mutations away from the wild-type, since the error-prone PCR dataset included variants with more than 8 mutations. Basic statistics about this landscape may be found in Figures 4.4e-h. The same metrics were used here as in the 8-site landscape. Due to batch effects in DHARMA experiments (coming from codon optimization and other cell-related effects), the results of models on the 4-site, 8-site combinatorial, and error-prone PCR datasets should not be directly compared.

4.4.3 Results

See Tables B.7, B.8, and B.9 for raw performance results on the 8-site combinatorial TEV landscape and Table B.10, B.11, B.12, and B.13 for raw performance results on the error-prone PCR TEV landscape.

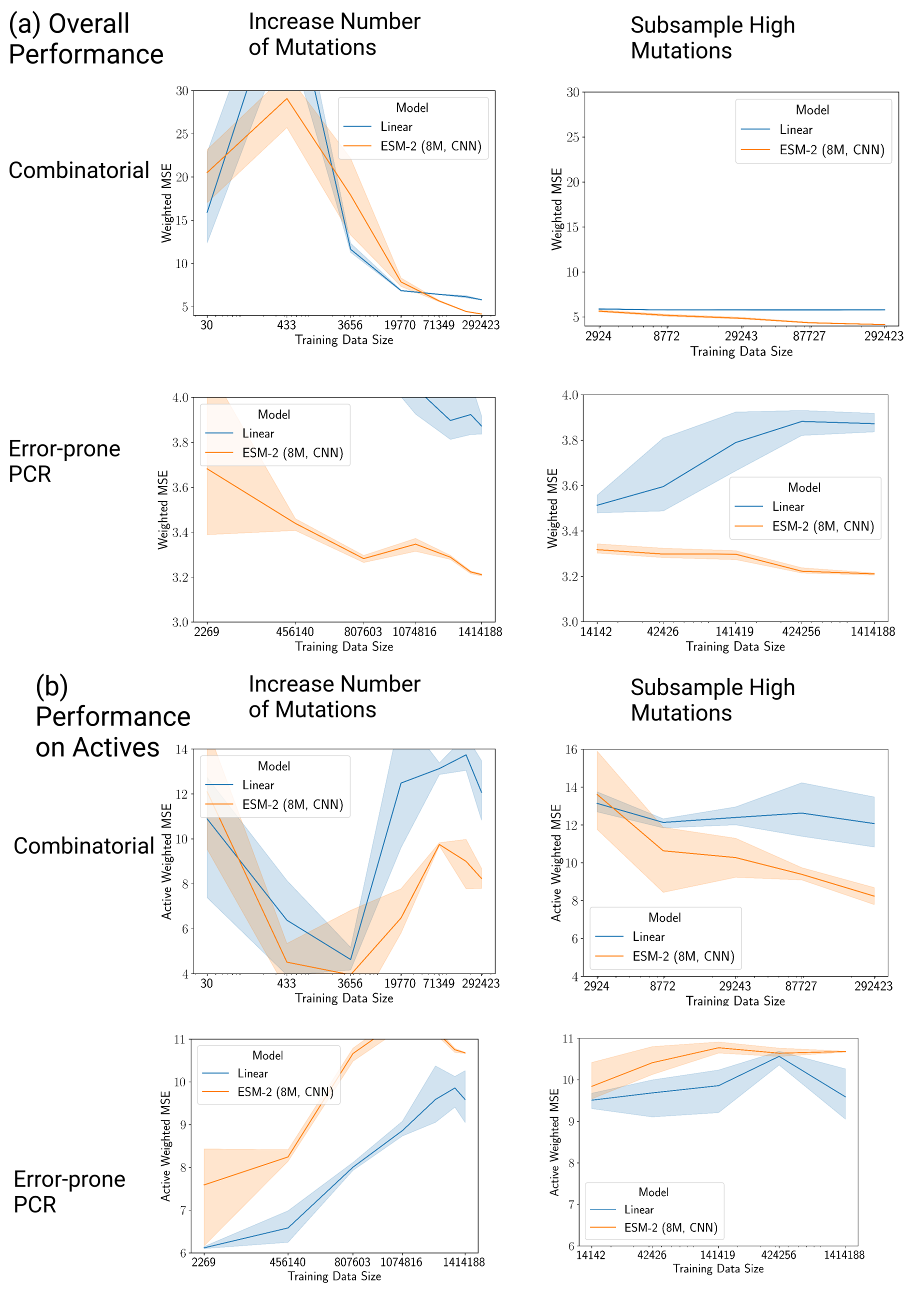


Figure 4.5: Data Scaling and TEV Protease Fitness Prediction. (a) Overall performance (measured by weighted MSE) on the (top) 8-site combinatorial and (bottom) error-prone PCR datasets with (left) scaling by increasing the number of mutations and (right) scaling by subsampling datasets of 7 or fewer mutations. Performance generally improves as data scale increases, but subsampling datasets of higher mutations is substantially more data-efficient than scaling by increasing the number of mutations. (b) In contrast, data scaling appears to worsen performance on actives (measured by active weighted MSE, or weighted MSE on the top 5% of actives in the dataset), under the same conditions. This is due to the increased number of inactives in the dataset with increasing data scale.

4.4.3.1 Data Scale and Model Performance

We begin by confirming that increasing training data size substantially improves model performance. In Figure 4.5a (left), we see that for both the combinatorial and error-prone PCR datasets, increasing training data size from 1 mutation all the way up to 7 mutations improves performance. The improvement is not exponential like in the 4-site landscape, since this is a much larger dataset, but it is still substantial and statistically significant.

However, there is another way of increasing data scale: by sampling increasingly larger subsets of a high-mutant dataset, as opposed to increasing the number of mutations allowed in the training set. In Figure 4.5a (right), we sample 1%, 3%, 10%, and 30% of the 7-mutant training set and examine the impact on model performance. We do see that increasing data scale improves model performance, but models trained on the small datasets here are substantially better than the corresponding low-mutant datasets on the left. This impact is most stunning on the 8-site combinatorial dataset: a training dataset of 2924 7-mutant variants yields a better model than a training dataset of 19770 5-mutants and almost as good a model as a training dataset of 71349 6-mutants. Similar results can be seen for the error-prone PCR dataset, though they are less dramatic.

Our results indicate that how we construct our training datasets is extremely important to maximize efficiency of the training data in improving model performance. Specifically, we want randomly subsampled datasets of variants with relatively high mutation counts from the wild-type, as opposed to thorough coverage of variants with relatively low mutation counts from the wild-type. This can yield an effective one or two orders of magnitude improvement in model performance, which is very significant even with the substantially larger data scale offered by DHARMA. Error-prone PCR is a very natural experimental technique for generating these sorts of datasets, so it is our preferred method of generating training data for protein fitness models. Single-mutant scans or combinatorial libraries, both extremely common methods of generating training data in the field [74, 265], are substantially less effective. These conclusions have only been validated on TEV protease, and therefore may not be applicable to other datasets on other proteases or other protein functions that have smoother landscapes.

Given only this data, it is difficult and largely misguided to ascribe a specific reason to why randomly subsampled datasets of variants with high mutation counts are better for model training; further experiments would be necessary to justify any intuition around these results. For example, one might want to say that epistasis has an important impact on the fitness landscape of TEV protease which explains why high-mutant variants are particularly important as compared to low-mutant variants. However, we see that linear regression on one-hot features (the baseline model) exhibits the same behavior as the ESM-based CNN, just with worse performance, on the combinatorial library and similar behavior on the error-prone PCR library. A linear regression model cannot possibly learn epistatic interactions by definition, so this behavior is not a result of epistasis. While it would be interesting to have a clear, human-understandable reason for these results, one must be very careful to ensure that such a reason actually explains the observed model behavior, not an anthropomorphized perspective of how we would like a given model to perform.

Error-prone PCR is one method of generating a library with sparse overage of a high-mutant landscape, but there are other, more rational methods that are also worth further investigation. For example, one could generate an alternative combinatorial library that looked at a larger number of sites and more amino acids and randomly subsampled to get sparser coverage of higher mutants. One could also look at approaches like variational synthesis that aim to model the DNA synthesis process as well and design the optimal library for a model to learn from [294, 295]. These approaches are beyond the scope of this thesis but would be very interesting when applied to the generation of libraries of millions of variants. For the purposes of this thesis, we will focus on the simpler approach of error-prone PCR.

4.4.3.2 Performance on Active Variants and Reweighting Loss

We now turn to evaluating model performance on the top 5% most active variants in Figure 4.5b. Surprisingly, the trend from the previous part reverses; models trained on larger datasets now tend to be substantially worse at predicting the fitness of the most active variants. This is a clear indication that our models have not truly learned the physics of the interaction between TEV protease and the wild-type substrate but instead learned an approximation from the available experimental data.

This effect is due to a shift in the training set as we increase data size. Most single mutants of TEV protease are active on the wild-type substrate, as expected, and at low mutation counts more variants tend to be active. However, at high mutation counts variants tend to be less and less active, meaning the training set contains a higher proportion of inactive variants. The models adjust for the greater proportion of inactives in the training set by learning more and more about predicting inactive variants, and are therefore less accurate at predicting active variants. We expect this effect to only occur on datasets targeting the wild-type substrate; datasets targeting distant, alternative substrates would have less of an issue since the wild-type protease is likely to be inactive. If we scale instead by subsampling high mutation count variants, in Figure 4.5b (right), we see a much smaller effect on the active weighted MSE; it is already quite poor at small data sizes, and does not dramatically worsen with an increase in data size.

There are a number of relatively standard machine learning techniques to fix issues with imbalanced data, like this one [296–299]. We found that the simplest robust approach was simply to reweight the loss function proportional to β^f, where f is the fitness of a given variant and β is a reweighting power. In Figure 4.6a on the 8-site combinatorial library, we examine the performance of our models on actives as a function of β when we increase the fitness cutoff required to be active. We see that as β increases, models do increasingly well on highly active variants with a clear tradeoff in performance on inactive variants. Based on these results, we set $\beta = 100{,}000$ for further investigation.

With a reweighted loss function, in Figure 4.6c and d we see the effects of data scaling on model performance on the error-prone PCR dataset. Model performance no longer decreases consistently with increasing training data in Figure 4.6c, but the performance on actives does in Figure 4.6d. Therefore, reweighting the loss function is an effective means of improving the performance of our model on predicting fitnesses of active variants and ensures that data scaling

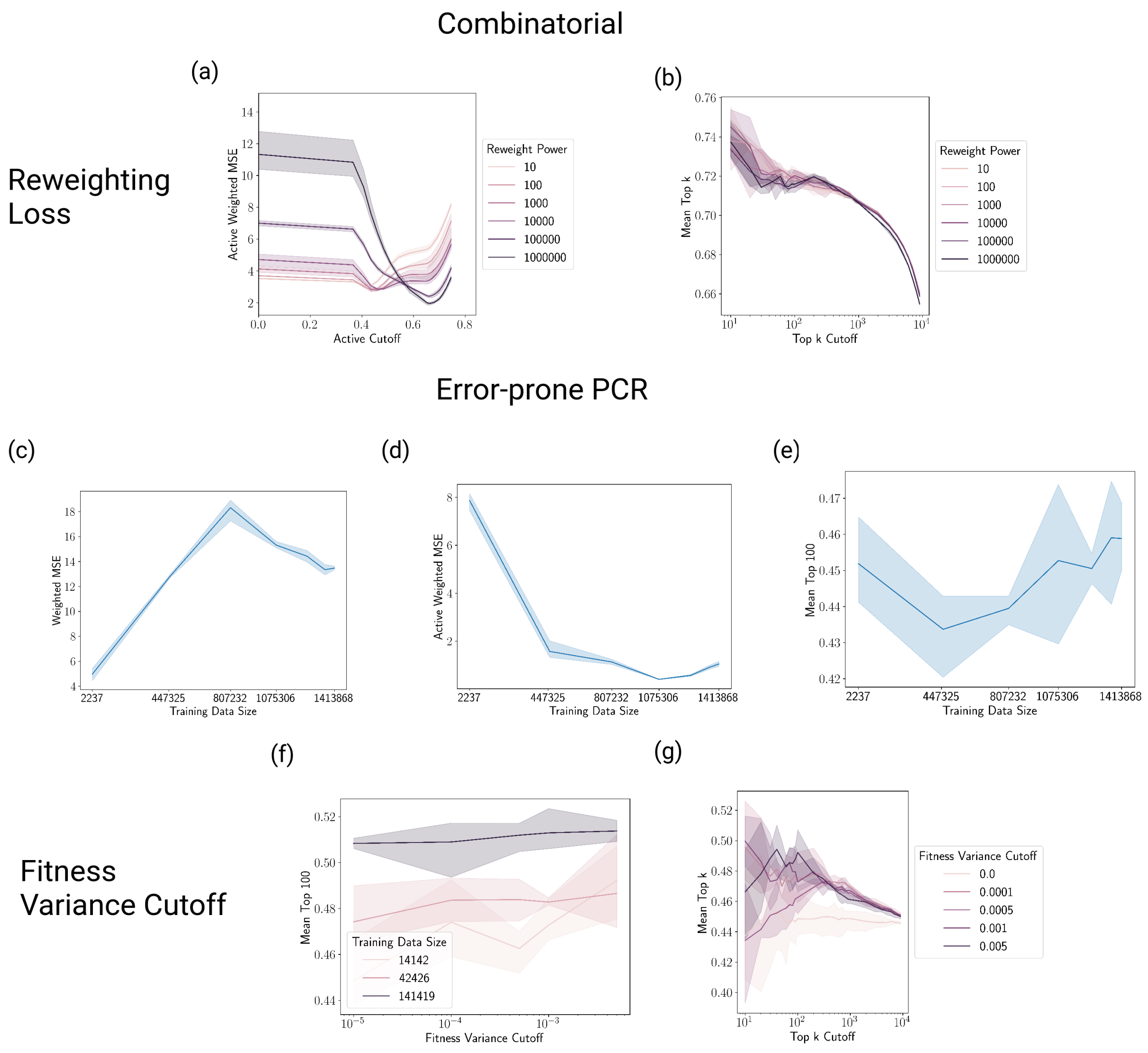


Figure 4.6: Impact of Reweighting Loss Function and Fitness Variance Cutoff on Model Performance. (a) Increasing reweighting power (exponent applied to inverse-variance weights) demonstrates a clear tradeoff between improving performance on actives and worsening performance on inactives in the 8-site combinatorial dataset. (b) The overall mean fitness of top k variants in the 8-site combinatorial dataset does not improve with reweighting power. (c-d) Overall performance, measured by (c) weighted MSE and (d) active weighted MSE (performance on top 5% most active variants) on the error-prone PCR dataset with reweighting loss, scaling by increasing the number of mutations. Performance on actives now consistently improves with increasing training data size. (e) The mean fitness of the top 100 variants does not improve with increasing training data size, even with reweighting loss. (f-g) Impact of fitness variance cutoff (cutoff on the minimum possible fitness variance) on model performance, focusing on mean fitness of top 100 or top k variants. (f) Placing a fitness variance cutoff improves designs, most notably on particularly small datasets (here 1%, 3%, and 10% samples of the 7-mutant training set for the error-prone PCR library). (g) This finding holds for top k variants, not just the top 100 variants (here shown for just the 1% sample on the error-prone PCR library).

does consistently improve model performance on active variants as well.

However, we have one last metric to check: our protein design effectiveness. We will look in more detail at this metric in the next two sections, but for now in Figures 4.6b and 4.6e we find that reweighting the loss function appears to have no substantial effect on mean top 100 predictions or mean top k predictions for any k. Worse still, comparing our results here to the results in Figure 4.7, we see that reweighting the loss function actually worsens our overall design fitness. This must be because the reweighting process encourages the model to produce more false positives which worsen overall design quality. So reweighting the loss function is useful in a very specific application: where we already have designs generated but want to estimate how good our designs are prior to experimental validation. This is an important use case in any setting where experimental validation is expensive – there is no point in validating designs that are unlikely to work – but not an integral part of the actual protein design process.

4.4.3.3 The Fitness Variance Cutoff

We now briefly justify the inclusion of a fitness variance minimum, as described in the methods section. In Figure 4.6f and g we see the impacts of the fitness variance cutoff for the smallest training datasets in the error-prone PCR landscape: the 1%, 3%, and 10% datasets of the seven-mutant training set in Figure 4.6f and just the 1% dataset in Figure 4.6g. Due to computational limitations we did not run this tuning process on any other datasets.

There is substantial improvement with increased fitness variance cutoff in the mean top 100 designs and the mean top k designs for any k on the smallest dataset (the 1% dataset), but this improvement becomes negligible for the larger datasets. We also do not see any significant effect from increasing the fitness variance cutoff, and we did not see any significant effect on the mean-squared-error or any of the other metrics described in this section. We expect the fitness variance cutoff to have the largest impact on mean top 100 fitnesses, since placing a cutoff encourages the model to learn about variants farther away from the wild-type that may be more fit. It should also have larger impact on datasets on alternative substrates where the wild-type is less active, not datasets on the wild-type substrate like the datasets we have here. As a result, finding a positive effect is overall promising for the inclusion of a fitness variance cutoff, but we should not expect it to have a substantial impact on overall model performance. Since we saw no strong effect on model performance from the exact value of the cutoff, we set the cutoff to 0.001.

4.4.3.4 Protein Design Effectiveness

We now turn to the mean fitness of the top 100 variants, a metric that explicitly looks at the effectiveness of a particular model in a protein design context. Previously, we already saw that a modeling decision – reweighting the loss function – could improve performance on active variants while damaging protein design effectiveness. In Figure 4.7a, we look at the mean top 100 fitness for our models on both the combinatorial and error-prone PCR dataset. On the left, we see that our models do tend to improve at protein design with increasing data scale, but we observe diminishing returns past the triple mutant dataset with increased data size. These results suggest that the mean top 100 fitness is less discriminatory as a metric than the weighted MSE; models can improve their weighted MSE without significantly improving their mean top 100 fitness and therefore their protein design effectiveness. Figure 4.7b allows us to verify that this holds true for the mean fitness of the top k variants for any value of k, not just $k = 100$.

On the right of Figure 4.7a and b, we see the impact of data scaling by subsampling higher mutant datasets on the mean top 100 and mean top k fitnesses. In the combinatorial dataset, there is essentially no impact from a larger data size. However, on the error-prone PCR dataset, we see very clear improvements in performance as data size increases, even up to the largest datasets we have previously gathered. This suggests that improving data size does not lead to a plateau in model performance, i.e. increasing data scale will still improve the performance of our protein fitness models at protein design. This effect is clearest when we scale by subsampling mutants on the error-prone PCR dataset, suggesting that datasets should be generated by error-prone PCR experiments or a similar random method of exploration, and they should contain variants with very high numbers of mutations from the wild-type. Given these constraints, we expect that

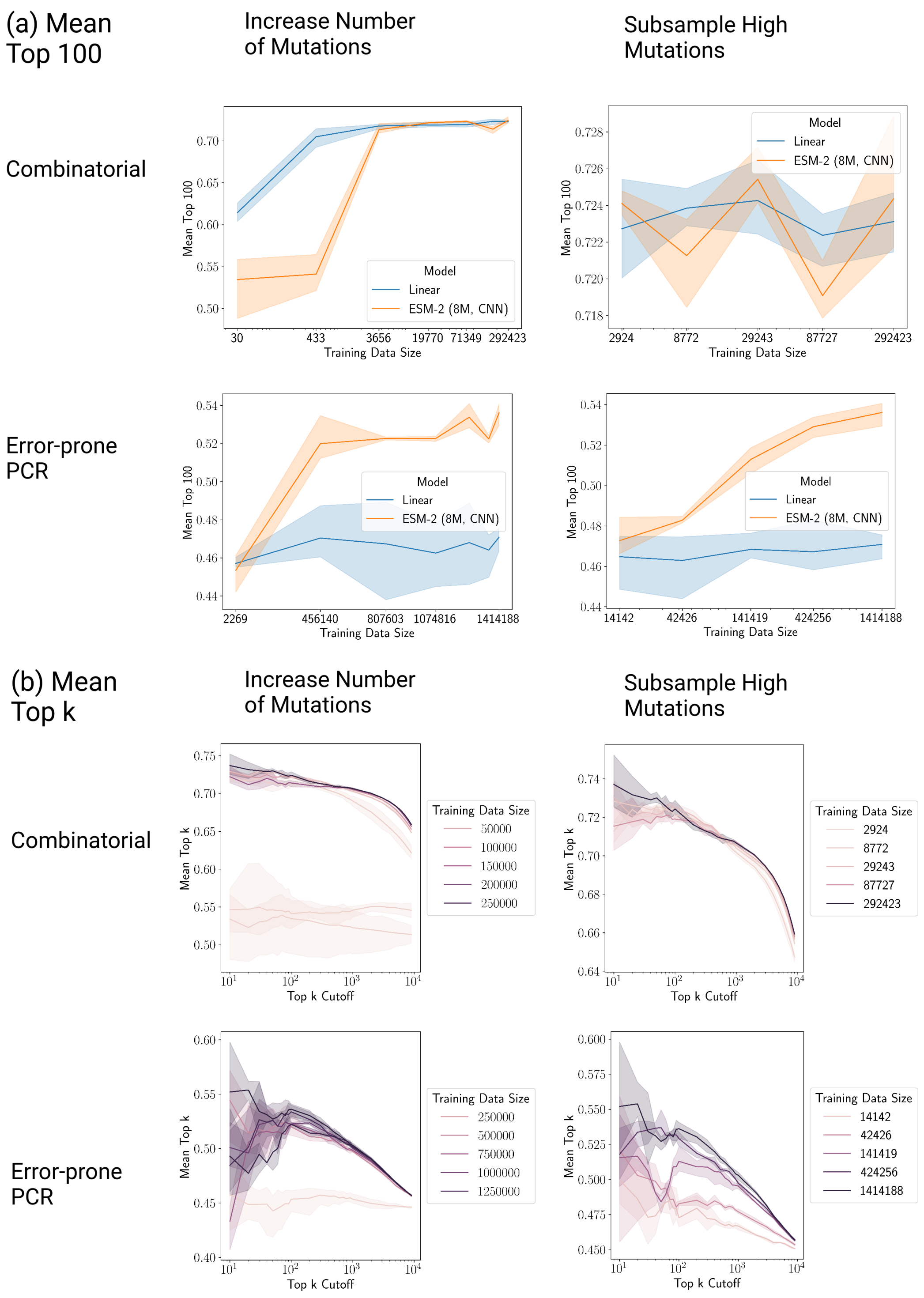


Figure 4.7: Data Scaling and TEV Protease Design Effectiveness. (a) Mean fitness of top 100 variants on the (top) 8-site combinatorial and (bottom) error-prone PCR datasets with (left) scaling by increasing the number of mutations and (right) scaling by subsampling datasets of 7 or fewer mutations. Protein design effectiveness measured by mean fitness of the top 100 variants clearly shows diminishing returns beyond the triple-mutant mark in both datasets, but we do see some improvement even with millions of variants on the error-prone PCR dataset and no sign of a data plateau. (b) These results do not change when we consider the mean fitness of the top k variants instead.

increased data scale will overall improve model performance. Finally, to highlight the importance of the fitness variance cutoff: without placing the cutoff, we found a very clear plateau in model performance at around 500,000 variants. Therefore, to truly unlock the potential of these very large protein fitness datasets, we must use a fitness variance cutoff when training our models.

4.5 Discussion

Our benchmarking results on the 4-site landscapes support three important conclusions about the development of future protein fitness ML models. First, common ML benchmarks like FLIP [266] or ProteinGym [74] need to be corrected to account for noise inherent in single-step selection by methods like FLIGHTED. Correcting these benchmarks will change downstream evaluations of protein models and should improve performance on average, especially for sufficiently large datasets and expressive models like CNNs. Second, data size is currently the most important factor, underscoring the importance of gathering large, high-quality datasets through methods such as DHARMA. Finally, the top model architecture matters much more than the embedding, so practitioners should focus on development of top model architecture and less on scaling up protein language models. Specifically, in one of the first systematic explorations of CNNs as top models, we found that they performed very well while being computationally cost-efficient. We encourage further exploration of the potential improvements that might be achieved by adopting CNNs and other poorly-studied architectures as top models. Based on these results, we selected ESM-2 embeddings and CNNs as a solid model architecture for modeling larger datasets.

Our results on data scaling from the 8-site combinatorial and error-prone PCR TEV landscape tell us how to design datasets for optimal machine learning model performance in designing proteins, at least for TEV protease. Specifically, the simplest possible approach is to use error-prone PCR datasets with a high enough mutation rate to at least generate triple mutants, and we want as many variants as we can possibly get, at least millions of variants. These datasets are easily acquired by DHARMA and FLIGHTED but difficult to acquire by other more traditional low-throughput means. We have also established that different metrics of model performance can yield different results on the same models, so it is important for researchers to specifically target the metrics they are most interested in for a particular application. In particular, model performance on all variants and model performance on active variants can disagree, though this can be fixed through re-weighting the loss function, and model performance on all variants can improve without substantially improving protein design effectiveness. While these conclusions have only been demonstrated for TEV protease, they likely generalize to some degree to other proteases, other enzymatic functions, or other fitness functions of interest; we leave understanding the degree of generalization to future work. With these results, we may now turn to the problem of engineering TEV protease to cut alternative substrates.

Chapter 5

Engineering TEV Protease

5.1 Introduction

We now finally turn to the original problem of this thesis: engineering TEV protease specificity on alternative substrates. To do this, we first need to generate fitness landscapes and train models for predicting fitnesses (i.e. protease activities) on alternative substrates. We examine model performance on these alternative substrates and find some clear weaknesses in our models: they are unable to design effective proteases for more difficult substrates far from the wild-type. We also more clearly observe the importance of the fitness variance cutoff, introduced in the last chapter.

Based on these results, we need one final piece to complete the engineering process: an optimizer, or some approach to picking variants given these models. We review a wide variety of literature approaches ranging from optimized Monte Carlo [300, 301] to Bayesian optimization [104, 107] and diffusion models [117, 123]. Based on a literature review and benchmarking on our data, we show that Monte Carlo optimization is the simplest and best approach for our purposes. We proceed to engineer TEV protease successfully on an alternative substrate, demonstrating specificity against the wild-type and greater specificity compared to anything found in the training set.

5.2 Landscapes on Alternative Substrates

5.2.1 Experimental Data Generation

We now generated fitness landscapes for TEV protease on 4 alternative substrates of varying difficulties. These landscapes were generated using the error-prone PCR library established previously, as we have discovered that this is the optimal method of generating fitness datasets for machine learning.

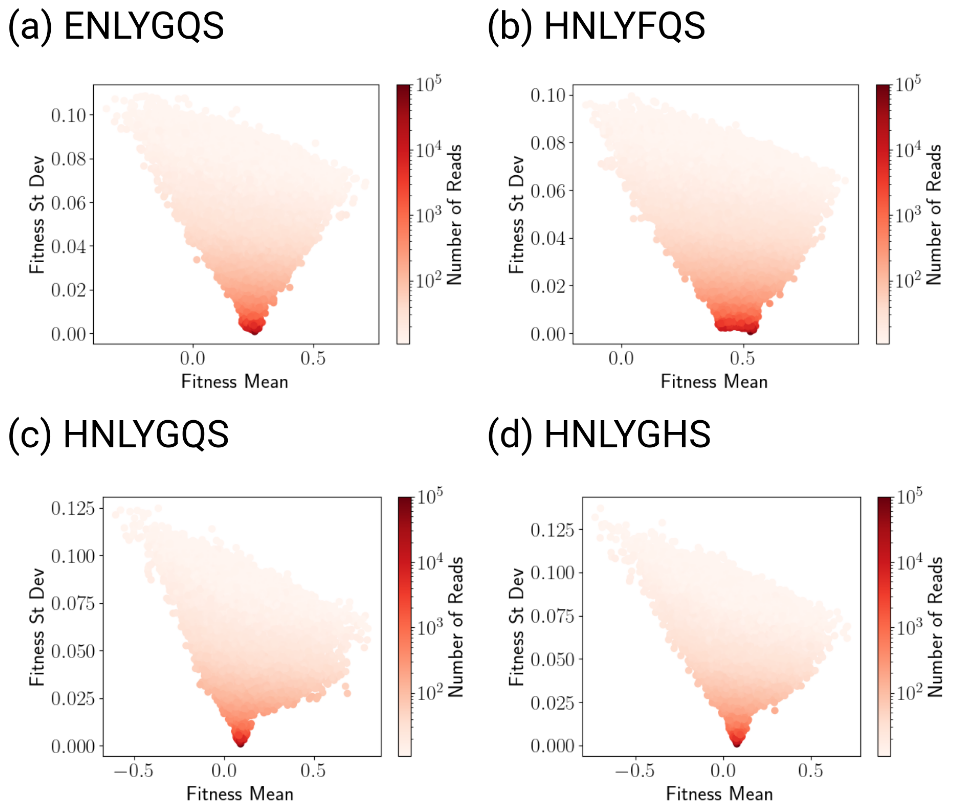


Figure 5.1: TEV Protease Fitness Landscapes on Alternative Substrates. Fitness means, variances, and read counts for an error-prone PCR library of TEV protease against 4 substrates: two single-mutants (ENLYGQS in (a) and HNLYFQS in (b)), one double-mutant (HNLYGQS in (c)) and one triple-mutant (HNLYGHS in (d)).

Figure 5.1 shows characteristics of the fitness landscapes generated by this process on our 4 alternative substrates; for reference, the wild-type substrate is ENLYFQS. We observe that with increasing mutation distance from the wild-type, our library becomes less and less active, as expected. Fitnesses also become more tightly concentrated around the wild-type fitness, since individual mutations have smaller impacts on the fitness of a largely inactive protease. These data demonstrate the substantial difficulty of the machine learning problem before us: for our models to generate new variants that are truly active on alternative substrates, we will have to do substantially better than anything found in the training set.

For the purpose of engineering TEV protease, we trained models on the entire fitness landscape to maximize use of training data. Prior to doing this, we investigated model performance by training models on just variants with 7 or fewer mutations from the wild-type, as was done previously for the wild-type substrate model, and split out a random 10% to use as a validation set. We now examine the performance of those models under the various metrics we have described, where the test set is all variants with at least 8 mutations from the wild-type. We also examine the impact of the fitness variance cutoff, as described previously, on engineering TEV protease to target alternative substrates.

5.2.2 Model Performance

See Table B.14 for raw performance results of these models.

As measured by the weighted mean-squared-error on the test set, all our models perform quite well, with weighted MSEs ranging between 1 and 4. Unfortunately, this high performance is not indicative of our models' protein design capabilities. In Figure 5.2, we examine the mean fitness of the top k variants to evaluate our models' protein design capability. We find that our models are generally decent at protein design on the wild-type substrate and on the two single-mutant substrates, but they are substantially worse on the more challenging double- and triple-mutant substrates. In fact, they show no sign of being able to generate any designs that are substantially more fit than the wild-type on these more challenging datasets. This is not simply because no such variants exist in the test set: there are a small number of high-fitness variants in the test set that our models could have found, but did not.

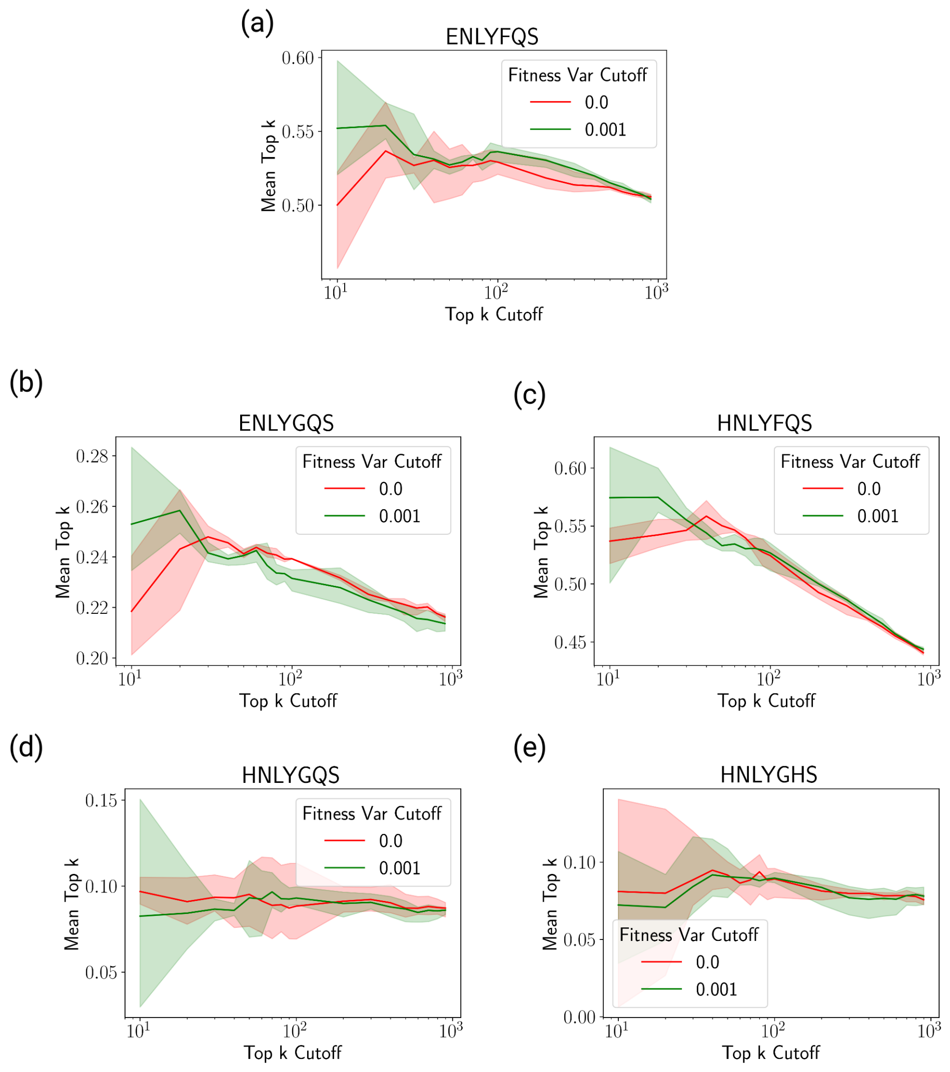


Figure 5.2: Performance of Fitness Models on Alternative Substrates. Performance measured by mean fitness of top k variants, for (a) wild-type substrate and (b-e) 4 alternative substrates as described previously. Our models do enrich for higher fitness for the wild-type substrate and the two single-mutant substrates, but show no signs of enrichment on the more difficult double- or triple-mutant substrates.

We wondered whether the fitness variance cutoff would help solve this issue. The fitness variance of anything close to the wild-type is unnaturally low in all of these datasets, for the same reasons described previously. Since the wild-type protease is not particularly fit on many of these alternative substrates, we thought that introducing the fitness variance cutoff would improve design quality by encouraging models to learn fitnesses of variants far from the wild-type protease.

The fitness variance of the test set, generally farther from the wild-type, is higher and makes the weighted MSE a less rigorous metric of model performance, therefore explaining some of our previous results. However, placing a fitness variance cutoff on training only seems to have a small positive impact on the wild-type and single-mutant substrates.

Another potential issue is the relative lack of active variants in the datasets for the more distant substrates. Wittmann, Yue, and Arnold [100] previously demonstrated that having active variants is important for the success of machine learning models, so that may apply here. However, our models' designs are still underperforming relative to the variants that are present in the test set but not identified, suggesting that this is not the only issue. This is also an issue that is in general not easy to rectify; if we had a reliable way of adding active variants on an arbitrary substrate to the training set, we probably do not need to use machine learning for protease engineering in the first place. So we would like to find an alternative solution.

Based on our results, unfortunately our models do not show substantial capability to generalize beyond the training set and learn which sequences correspond to active proteases on alternative substrates. The substantially larger scale of the training data available to our models has helped performance, but not in the way we want to develop proteases active on alternative substrates. As such, we need more sophisticated and expressive model architectures, likely transformer architectures, that are capable of learning the more complicated landscapes. Alternative training paradigms that increase our model's focus on highly active variants are likely to also be beneficial. Unfortunately, we did not have the time to pursue either of these lines of research, so we leave them to future work.

Given these results, we opted to pursue protein engineering attempts on the best-performing substrate, the single-mutant HNLYFQS. We engineered for both activity and specificity against the wild-type; specificity was measured by subtracting the predicted fitness on HNLYFQS from the predicted fitness on the wild-type substrate ENLYFQS.

5.3 Optimization on a Fitness Landscape

Given a model that predicts the fitness of a given variant, we now need to design optimal variants for this fitness function. This requires an optimization approach. A number of optimizers or generative models that function as optimizers have been developed in the literature; here, we briefly review some of the most common and discuss our primary choice. We have one important additional constraint on our optimizer: we know that our model is most accurate in the immediate neighborhood of TEV protease, since the training data is from that region, and we expect our desired variants to also be in that region. So we are looking for an optimizer that can somehow constrain its output to remain within the immediate neighborhood of TEV protease, with some distance cutoff to be defined later.

5.3.1 Optimization Approaches

5.3.1.1 Monte Carlo and Improvements

Let f be the function we want to optimize (represented by some neural network) over some set S, in this case the set of all variants of TEV protease within some region of the wild-type. One of the most standard methods of optimizing for f involves turning this into a sampling problem, sampling from the distribution $p(x)$ with

$$\log p(x) = \beta f(x) - \log Z, Z = \sum_x e^{\beta f(x)}$$

for some temperature parameter β [301]. Essentially, we treat f as the unnormalized log-probability of some probability distribution p. We may now use Markov Chain Monte Carlo (MCMC) to sample from this probability distribution p, and our samples should be relatively optimal assuming the MCMC process converges [302]. In an MCMC process via the Metropolis-Hastings algorithm, we draw new proposed samples from some proposal distribution $q(x'|x)$ given our present position

x, and accept the proposed updates with probability

$$\min\left(\exp(f(x') - f(x))\frac{q(x|x')}{q(x'|x)}, 1\right).$$

We continue this process until convergence of the MCMC trajectory. The most common variant of MCMC, Gibbs sampling, uses a proposal distribution $q(x'|x)$ that varies one dimension at a time and conditions on all other dimensions [300]. All MCMC-based approaches can trivially be modified to restrict x' to be within some radius of the wild-type, as desired.

MCMC can be very slow computationally; in particular, Gibbs sampling scales proportional to the dimension of the data, which can be quite large for protein sequences [300]. However, we can considerably accelerate this process by using discrete analogs for gradients. Specifically, suppose our inputs have dimension D (the length of the sequence) and $K = 20$ possible values (for each of the amino acids). Then we may try to bias $q(x'|x)$ towards residues in the sequence that we think are more likely to affect fitness, i.e. residues for which the gradient is highest.

To rigorize this process, we use the method known as Gibbs with Gradients [301]. In this process as adapted for protein data, we first compute a difference function that approximates a gradient

$$\widetilde{d}(x)_{ij} = \nabla_x f(x)_{ij} - x_i^T \nabla_x f(x)_i$$

where $\widetilde{d}(x)_{ij}$ is the gradient of flipping the ith dimension of x to j. With a one-hot embedding of a protein sequence, $\nabla_x f(x)_{ij}$ and $\nabla_x f(x)_i$ may be computed exactly since f is completely differentiable. With a protein language-based embedding, f is no longer differentiable due to the initial embedding of amino acids (a non-differentiable step) so we use difference functions to approximate it. Unfortunately, this increases computational costs, but computations of f can be run in batches as necessary.

With $\widetilde{d}(x)_{ij}$, we now compute the proposal distribution

$$q(i, j|x) = \text{Categorical}\left(\text{Softmax}\left(\frac{\widetilde{d}(x)}{2}\right)\right)$$

where the softmax is taken over all D values of i and all $K-1$ possible values of j per residue. We now may compute $q(i,j|x')$ similarly and accept the new proposal with probability

$$\min\left(\exp(f(x')-f(x))\frac{q(i,j|x')}{q(i,j|x)},1\right)$$

as before. Gibbs with Gradients [301] has been previously mathematically proven to have substantially improved convergence properties when compared to MCMC. It has also been used previously in protein design contexts, alongside a graph-based smoothing algorithm to eliminate noisy gradients due to experimental noise [303]; we adopted essentially the same Gibbs with Gradients approach, but omitted the graph-based smoothing since we already have FLIGHTED to deal with experimental noise.

Gibbs with Gradients still makes only one coordinate change at each step of the trajectory, which is highly inefficient for long protein sequences. An alternative approach is the discrete Langevin sampler [304], a variant of the highly efficient Langevin algorithm for sampling continuous spaces. The Langevin sampler in continuous space is

$$x' = x + \frac{\alpha}{2}\nabla f(x) + \sqrt{\alpha}\xi, \xi \sim \mathcal{N}(0, I_{D\times D}).$$

By analogy, Zhang, Liu, and Liu [304] demonstrate that we may define a discrete Langevin proposal distribution

$$q(x'|x) = \prod_{i=1}^{D} \text{Categorical}\left(\text{Softmax}\left(\frac{1}{2}\nabla f(x)_i(x'_i - x_i) - \frac{(x'_i - x_i)^2}{2\alpha}\right)\right).$$

Since this is the product of D distributions, one for each coordinate, it will update every coordinate in a single step of the trajectory. However, these distributions may be computed in parallel and the gradient computations may also be done in parallel, so each individual step of the discrete Langevin sampler is not any more expensive than that of the Gibbs with Gradients sampler.

We now examine the practical performance of these approaches on a protein design task. For this, we use the 4-site TEV library and design variants with optimal cleavage of the wild-type substrate drawn from the 4-mutant test set. The fitness model is the ESM-1b model with a CNN on top trained on the three vs. rest split. We started all three trajectories at WPMV, the point in the test set with the lowest fitness, and ran 10 trajectories for 2000 steps. We used a temperature of 0.01 and a step size of 0.2 for the discrete Langevin sampler.

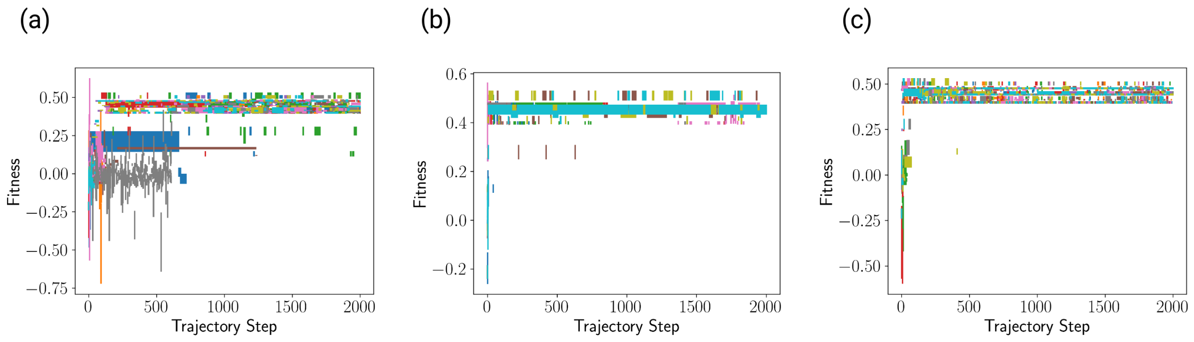


Figure 5.3: Performance of MCMC-Based (Markov chain Monte Carlo) Optimizers on 4-Site TEV Landscape, for (a) Gibbs sampling, (b) Gibbs with gradients, and (c) discrete Langevin sampling. See the Methods section for the details of these optimizers. We see that all 3 approaches converge, but both Gibbs with gradients and discrete Langevin samplers are considerably more efficient than just Gibbs sampling.

Figure 5.3 shows the generated trajectories. All 3 approaches converge to something close to the maximum of the test set. However, Gibbs with gradients and the discrete Langevin sampler are considerably more efficient than plain Gibbs sampling. In this situation, since there are only 4 amino acids to sample over, the speedup is not particularly significant: around 2 hours to 10 minutes on a personal laptop. However, for engineering larger protein sequences, this speedup could become very significant. There was no substantial difference observed between discrete Langevin and Gibbs with Gradients, likely because there were only 4 sites to mutate. However, we found that discrete Langevin samplers had a substantially more difficult time sticking close to the wild-type when allowed to optimize larger protein sequences, so we stuck with Gibbs with Gradients for further protein engineering efforts.

5.3.1.2 Other Strategies

We now briefly review some alternative non-MCMC-based approaches in the literature to optimizers for protein engineering and explain why they are not as well-suited for our particular problem here. Bayesian optimization is one popular approach to biological sequence design problems, focusing on problems where scientists perform design-build-test-learn cycles, as described in the Introduction. In this context, it is generally advantageous to spend some rounds exploring the landscape prior to optimizing the given fitness function. The Bayesian optimization literature has demonstrated considerable impact in this regime, but not in any regime where we have only one round of experimental data generation prior to design [104–114]. Since we were limited to only one round of experimental data generation by external constraints, we did not pursue a Bayesian optimization-based approach.

Another approach is by using a generative model on top of the already-trained predictive model to generate new designs. One potential advantage of this approach would be the incorporation of structural information from a structure-primary or structure-sequence diffusion model, since our predictive models are purely sequence-based. A number of approaches have been proposed in the literature for use of a diffusion model in generating proteins that satisfy certain constraints, as discussed in the Introduction [32, 117, 118, 121–129, 135, 305], but these have not been validated for our use-case of a predictive model on a local sequence landscape around TEV protease. Specifically, we want a diffusion generative model that can generate optimal protein sequences given a predictive model trained to accept only sequence input.

The closest literature match we found was NOS (diffusioN Optimized Sampling), a method of gradient-guided sampling from a diffusion model [306]. NOS trains a predictive model and a diffusion denoising model simultaneously on the same continuous latent space, and then performs the diffusion process adjusted by gradients from the predictive model. It was validated for relatively smooth, global sequence-space functions, like β-sheet percentage, solvent-accessible surface area, or antibody binding [306]. When we tried to adapt NOS to the much more rugged TEV protease landscape, we found that the resulting predictive model had such low accuracy as to be essentially worthless, despite using the same CNN architecture as our previous models. The predictive model must be trained on noisy sequence data, and noisy protease sequence data appears to be very difficult to learn on given the ruggedness of the TEV protease landscape. As a result, effectively integrating diffusion models and large experimental datasets of protease activity or other rugged landscapes remains an open problem beyond the scope of this thesis.

5.3.2 Application to TEV Protease Engineering

5.3.2.1 Methods

Given these optimizers, we now finally turn to the problem of engineering TEV protease to cleave an alternative substrate. We select HNLYFQS, the substrate for which we previously demonstrated optimal performance. We use the same model architecture as described previously (unfortunately without the fitness variance cutoff due to lack of time) and train on the full dataset instead of any sample thereof. We train one ensemble of 3 models on HNLYFQS and one ensemble of 3 models on ENLYFQS, the wild-type substrate. We then aim to optimize activity on HNLYFQS, predicted as the average of the 3 models in the ensemble, and specificity on HNLYFQS against the wild-type substrate ENLYFQS, predicted as the difference in the two ensemble averages. We run Gibbs with Gradients with 100 trajectories for 200 steps each, restricting ourselves to 15 mutations from the wild-type, and setting temperature to 0.01. We then selected the 72 most active and 72 most specific variants from the trajectories for further experimental testing.

Experimental testing was done with DHARMA on both the wild-type and alternative substrate. In order to compare results from the new DHARMA assay with the previous training run, we also included a number of control variants: variants that were also present in the training set. These included the 5 variants with the highest activity and specificity in the training set as well as 5 variants with a range of activities on each of the wild-type and alternative substrates. Comparing these results allowed us to directly evaluate the impact of machine learning by looking at how much the designed variants improved on the best variants found in the training set. We did not compare the validation results directly to the training set to avoid having to account for known batch effects in DHARMA experiments. For the experimental assay, genes encoding the designed proteases were synthesized and pooled together into a mini-library; the same procedure as described before was then performed to run the actual DHARMA assay. Experimental testing was done by my experimental colleague Boqiang Tu.

5.3.2.2 Results

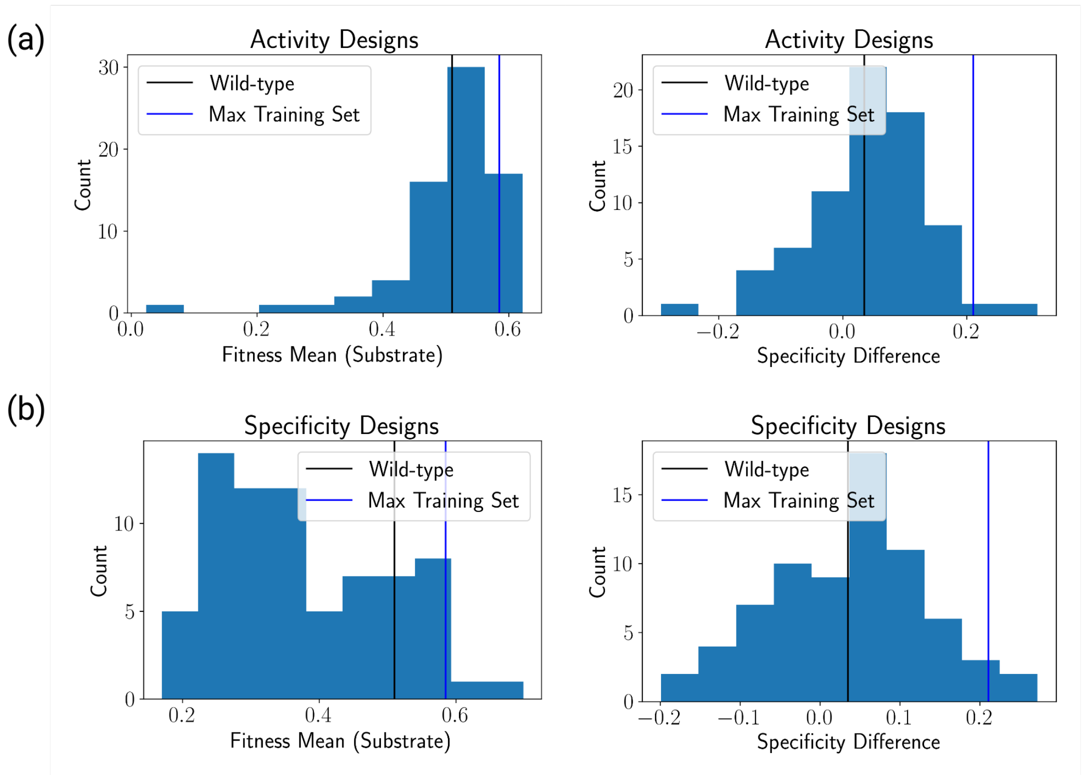


Figure 5.4: Experimental Activity and Specificity of TEV Protease Designs on Alternative Substrate HNLYFQS. Specificity measured as difference between activity on substrate and activity on wild-type. (a) shows activity designs (designs optimized only for activity on HNLYFQS) and (b) shows specificity designs (designs optimized for specificity on HNLYFQS against the wild-type). Our results indicate successful engineering of TEV protease to target the alternative substrate HNLYFQS, both for activity and specificity, when compared to the maximum fitness found in the training set and to the wild-type.

The experimental performance of our engineered variants is shown in Figure 5.4. Our activity designs had better activity than the wild-type in 43 of 72 designs (a 60% success rate) and better activity than the best member of the training set in 8 of 72 designs (an 11% success rate). While we did not engineer for specificity with these designs, they are nevertheless sometimes more specific; we saw 42 of 72 designs with greater specificity than the wild-type and 1 of 72 designs with greater specificity than the best member of the training set.

Our specificity designs also performed well. We saw better specificity than the wild-type in 40 of 72 designs (a 56% success rate) and better specificity than the best member of the training set in 3 of 72 designs (a 4% success rate). The specificity designs were in general less active than the activity designs; they are substantially farther away in sequence space, generally around 13 to 14 mutations away as opposed to the activity designs that are around 3 to 4 mutations away, and have an additional constraint in the optimization, so this is somewhat expected. 14 of 72 designs were more active than the wild-type and 2 of 72 designs were more active than the best member of the training set.

We consistently saw at least some success in generating variants via machine learning that were better than anything observed in the training set, indicating that our machine learning modeling and design process contributed some value to protease engineering beyond simply picking the best member in the training set. Our success rates in the 5 to 10% range seem to indicate that testing around 100 designs will consistently yield some success for protease engineering problems of similar difficulty. On the other hand, it is unclear that our specificity engineering actually provides any benefit over the activity engineering; we saw similar specificities in both sets of designs. This may indicate that subtracting the predicted fitness on the wild-type substrate is actually not a particularly effective means of engineering specificity.

5.4 Discussion

We have demonstrated successful engineering of TEV protease on an alternative substrate for both activity and specificity. Around 5 to 10% of our designs outperformed the best-performing member of the training set, indicating that our machine learning added value to the engineering process beyond the experimental assay, even when millions of variants were tested experimentally. This demonstrates clear value for the use of machine learning in protease engineering. Further *in vitro* testing in non-DHARMA assays is necessary to determine how much more active and specific our designed variants are and how valuable this would be for any potential applications.

However, our retrospective analyses on alternative substrates indicate that this approach would likely not have worked for any other substrate. The machine learning, while performing very well by the weighted MSE, is simply not able to identify active variants on substrates where the majority of the training set is not particularly active. Since there are some active variants in the data that the model is not identifying, this is likely an issue with the modeling as opposed to the experimental data generation. It is likely that more expressive and complicated models like transformers could help take advantage of the large data scale and perform better. Unfortunately, designing and training those models is cost-prohibitive on an academic budget. We will discuss next steps on the protease engineering in the next chapter.

Chapter 6

Conclusion

In this thesis, we have developed a number of powerful machine learning and experimental tools for protein engineering and applied them to the challenging problem of engineering TEV protease specificity. We developed a suite of methods for generating fitness landscapes from high-throughput experiments: a custom latent variable method for continuous evolution data, and FLIGHTED, a Bayesian inference method that applies generally and more specifically to single-step selection experiments and to DHARMA data. Our methods consistently yield robust, accurate fitness landscapes across a wide range of experimental parameter settings. In the case of FLIGHTED, we further generated well-calibrated errors that can be reliably used by experimentalists to optimize signal-to-noise and by machine learning practitioners to improve model performance. These methods also allow us to better understand which experimental techniques are best for training downstream machine learning models; in particular, continuous evolution seems to not be particularly good at generating large quantities of data, single-step selection is generally better at distinguishing inactives, and DHARMA is generally better at distinguishing actives. This knowledge can help protein engineers more effectively pick experimental methods and combine results from different experimental assays into a single protein engineering campaign.

We next examined machine learning approaches to model protein fitness given an experimentally generated landscape. Our results on smaller 4-site landscapes indicate FLIGHTED is generally useful in improving model performance, especially for larger training set sizes and more accurate models. We found that data size is the most important parameter in improving model performance, and that among modeling decisions, top model choice matters the most. As we scaled up, we demonstrated that error-prone PCR with at least triple mutants is the best way to generate protein fitness datasets, and that different performance metrics may disagree on model performance; it is important to focus on metrics like the mean of the top 100 or top k fitnesses as those most closely correspond to performance in protein engineering.

Finally, we used the methods previously developed to engineer TEV protease to cleave alternative substrates. Our results demonstrated successful engineering of TEV protease specificity and improvements in both activity and specificity when compared to the training set. This approach demonstrates the advantages of very large screens such as those made possible by DHARMA as well as the utility of machine learning and modeling fitnesses beyond simply picking the best variant from a screen of even very large size. However, retrospective results indicate that our approach is likely not generalizable to other substrates; more sophisticated and expressive modeling is probably necessary.

6.1 Next Steps

Our most immediate next step is to develop larger protein fitness models in a way that is both reasonably cost-effective and scalable to the large datasets we now have access to. To do this, we could try to look for data and model scaling laws in protein fitness models similar to already-established Chinchilla scaling laws found for large language models [307]. Chinchilla scaling laws postulate that given a model with N parameters and D data points (or tokens for LLMs), the loss of the model is approximately

$$L = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

with some parameters E, α, β to be estimated from data, i.e. training a large number of smaller models [307]. Then, given a fixed compute budget $C \propto ND$, we may compute the optimal values of N and D, giving us the ideal model size and number of epochs to train a given model for (with a given data size). This data scaling approach should apply to protein fitness models as well, specifically letting N be the number of parameters in the top model, and will help us choose the appropriate size of our top model given a computational budget and data size without cost-prohibitive training runs.

We can also consider more expressive model architectures, especially for the top model, beyond convolutional neural networks. The most popular choice here are attention-based architectures like transformers [56]. Transformer-based top model architectures have, to my knowledge, not been substantially explored for protein fitness modeling, but given our data scale they may become a reasonable option.

Another area that deserves further exploration is the use of fine-tuned protein language models, which we briefly explored in the context of 4-site landscapes but did not pursue further. Options like low-rank fine-tuning have proven to be both compute-efficient and effective for large language models [308, 309]; similar methods may prove effective for protein fitness models as well [310]. Fine-tuning has predominantly been used for small datasets in protein fitness modeling and may not prove as effective for large datasets, as we found in Chapter 4, but it is perhaps worth further exploration since our models are not performing as well as we would like. As an alternative to fine-tuning to the experimental data, we can explore other approaches of training protein language models on altered datasets. Training a protein language model on a dataset of all proteases effectively starts by assuming totally homogeneous specificity, since the MSA of all proteases is assumed to be relatively promiscuous. We can use protease-substrate datasets [311] to train alternative protein language models that do not assume homogeneous specificity, and then fine-tune or use a top model on top of these altered protein language models to train to the desired specificity. This may improve performance by using a protein language model that expects to generate proteases of a desired specificity, as opposed to generically generating any protease.

Finally, we may explore more complicated alternative means of data generation beyond the error-prone PCR dataset. We know that we want sparse coverage of a high-mutant landscape, and given the specificity training results we would like to enrich for active variants. One approach is to use experimental techniques like directed evolution or PRANCE to evolve closer to the desired specificity, and then run DHARMA on an error-prone PCR library around a new target. Computational techniques like ancestral reconstruction or mining existing protease-substrate datasets could also complement this approach. In these cases, one would need to be careful to ensure that the machine learning still outperforms any alternative. We could also use more rationally designed libraries, like combinatorial libraries or libraries designed through variational synthesis [294, 295], along with zero-shot predictions of protease fitness on a specific substrate to enhance performance further. These steps may be necessary to engineer TEV protease against more difficult targets.

To truly engineer useful specificity on TEV protease, we need to go beyond specificity just against the wild-type substrate and model interactions with all potential alternative substrates. This requires running a library-on-library screen between a protease and substrate library and then learning a protease-substrate landscape across many different proteases and many different substrates. DHARMA can be scaled up even further to make the library-on-library screen practical, and a slight modification of the DHARMA circuit allows for sequencing both the substrate and protease in the same read, along with the edited canvas sequence. Given our current scale, it seems plausible to generate a combination of around 10^2 substrate sequences and 10^5 protease sequences.

Learning the protease-substrate landscape likely requires substantially more expressive and complicated machine learning models than those we have used here; this problem is the next major problem to be tackled to truly unlock the potential of protease engineering with machine learning. We may simplify the problem by ignoring substrates far from the wild-type, since we expect that all proteases we explore are likely to be inactive on such substrates. Therefore, 10^2 substrate sequences are likely sufficient to explore a substantial portion of substrate space. We may then build a joint cross-attention model, similar to ones already used in protein-protein interaction prediction [312], to effectively learn relationships between proteases and substrates.

Publically available databases such as MEROPS [311] on protease-substrate interactions can help us build these models prior to testing on DHARMA data.

Given a comprehensive protease-substrate model, we can explore more substantial engineering challenges around specificity. We can engineer proteases to be specific against all alternative substrates, or engineer proteases to have specific substrate profiles, using multi-objective optimization. These engineering problems are likely to be substantially more challenging and less likely to work, but with a sufficiently accurate model we should be able to make proteases with alternative substrate specificities that are at least as specific as the wild-type TEV protease. This will unlock many of the potential applications discussed in the Introduction of this thesis.

We have also completely ignored structure for this entire project, despite the many advances in structural modeling that have been made in the past few years. Our single foray into using structure-based and sequence-based diffusion models that incorporate some structural knowledge did not end particularly well. However, there is clearly room for improvement by using structure in the modeling process; combining the power of diffusion model-based approaches or other joint sequence-structure probabilistic models with a large amount of experimental data remains a very important open problem in protein design. Being able to combine these orthogonal approaches that have both proven independently capable will likely supercharge the potential of ML-based protein design.

Finally, there still remains the possibility of completely *de novo* enzyme design, which for functions like specificity seems largely out of reach so far. A TEV protease-substrate model like the one described above opens up possibilities of *de novo* designing proteases that target specific substrates, given sufficient experimental data. It is also plausible that sufficient experimental data generated by systems like DHARMA can unlock *de novo* enzyme design in general, beyond just TEV protease or proteases in particular. Learning how to effectively incorporate this experimental scale with the recent advances in sequence-based and structure-based modeling will likely prove crucial in making *de novo* enzyme design a reality.

Appendix A

Impact of Read Count on 4-Site TEV Landscape

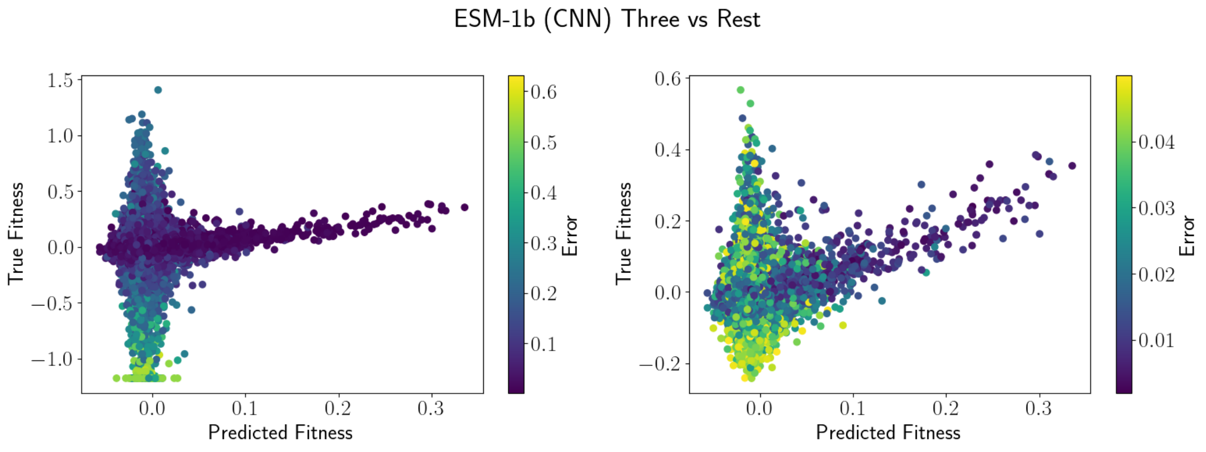


Figure A.1: Sample Model Predictions on TEV Dataset, produced by ESM-1b CNN on three-vs-rest. Left includes all data points and right includes data points filtered with error < 0.05 to highlight most accurate model predictions.

We begin by examining model predictions for a single high-performing model (the ESM-1b CNN on three-vs-rest) in Figure A.1; here, predicted fitness refers to the fitness predicted by the ESM-1b model, and true fitness refers to the FLIGHTED-measured fitness with the error (variance) shown on the colorbar. The model has reasonably high accuracy on predictions with very low variance (< 0.01) and predicts most other data points with high variance to be inactive, suggesting the model has learned to filter by variance when making predictions. Figure 4.1d suggests that the primary determinant of variance is read count, so this implies the model has learned read count on the test set, potentially surprising.

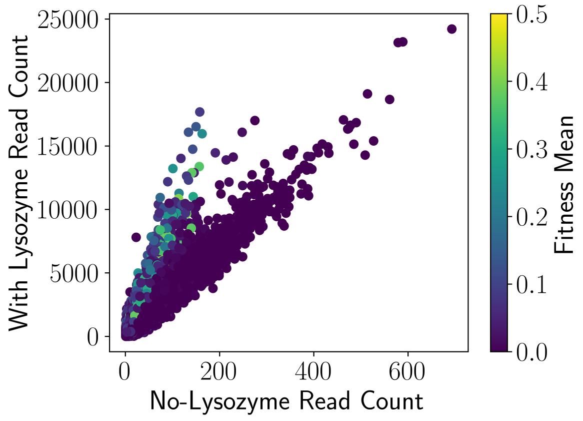


Figure A.2: Read Count With and Without T7 Lysozyme. Active variants have higher read counts with the T7 lysozyme.

We investigated this result further to ensure that there was no accidental data leakage at any point. There are two effects we identified that allow the model to learn read count on the test set. First, the T7 lysozyme is slightly toxic to the cell and when not cleaved (i.e. when the protease is inactive), results in a lower read count. In Figure A.2, we compare the read count in libraries with and without the lysozyme and see that active variants have higher read count in the library with the lysozyme. Since the models are trained to learn fitness, they can also implicitly learn about this effect.

Second, the initial (no-lysozyme) library construction was generated by a potentially biased synthesis process, so the initial read count is learnable, i.e. similar variants have similar read counts. We tested this by training a model (the same ESM-1b CNN) on the initial read counts of a training set consisting of triple mutants and predicting read counts on the quadruple mutants, as shown in Figure A.3. Since read count is learnable, we conclude that models are learning a combination of the initial read count from the biased library and TEV protease fitness when making predictions. This dataset and these models are still generally useful for predicting TEV

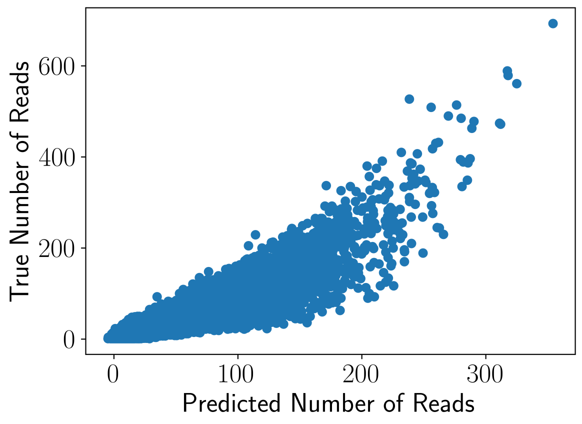


Figure A.3: Control Model Trained on No-Lysozyme Read Count, showing that read count is learnable.

protease activity and benchmarking ML models, but any active variant that had low read count in the initial library would likely be a false negative on any model trained on this dataset. To confirm that the models did learn TEV protease activity, we verified that the models predicted low activity for variants with high read count and low activity, as expected. In the future, we aim to reduce library bias to generate higher-quality TEV protease datasets and more accurate models to mitigate this issue.

Appendix B

Model Performances

Tables B.1 and B.2 contain model performance results on the 4-site GB1 landscape, with and without FLIGHTED-selection (measured by MSE and weighted MSE). Tables B.3 and B.4 contain similar results with model performance measured by weighted Spearman r. Table B.5 contain model performance results on the 4-site combinatorial TEV landscape on the wild-type substrate measured by weighted MSE. Tables B.7, B.8, and B.9 contain model performance results on the 8-sites combinatorial TEV landscape on the wild-type substrate measured by weighted MSE, active weighted MSE, and mean top 100 fitnesses respectively. Tables B.10, B.11, and B.12 contains model performance results on the error-prone PCR TEV landscape on the wild-type substrate measured by weighted MSE, active weighted MSE, and mean top 100 fitnesses respectively. Table B.13 explores the impact of varying the fitness variance cutoff on various parameters.

Model	One vs Rest	One vs Rest FLIGHTED	Two vs Rest	Two vs Rest FLIGHTED
Linear	1.4706 ± 0.0127	12.0096 ± 0.0709	1.5174 ± 0.0020	9.8847 ± 0.0342
TAPE	1.5062 ± 0.0090	11.8502 ± 0.2601	1.9685 ± 0.1161	7.1776 ± 0.3339
ESM-1b	1.4898 ± 0.0004	**11.2226 ± 0.0750**	1.4427 ± 0.0107	5.7642 ± 0.2590
ESM-1v	1.5132 ± 0.0016	14.7296 ± 0.0125	1.5646 ± 0.0153	6.7795 ± 0.2354
ESM-2 (8M)	1.5169 ± 0.0177	11.6852 ± 0.0226	1.4388 ± 0.0081	7.6910 ± 0.1750
ESM-2 (35M)	1.5218 ± 0.0218	11.5808 ± 0.0899	1.4034 ± 0.0103	8.2036 ± 0.0373
ESM-2 (150M)	1.5129 ± 0.0210	11.5738 ± 0.1378	1.4099 ± 0.0057	8.5423 ± 0.0238
ESM-2 (650M)	1.4963 ± 0.0024	11.5768 ± 0.2312	1.4285 ± 0.0288	5.5143 ± 0.2098
ESM-2	1.9206 ± 0.0334	12.0280 ± 0.1538	1.3472 ± 0.0071	5.0572 ± 0.3065
ProtT5	1.5288 ± 0.0148	11.6078 ± 0.0677	1.3567 ± 0.0259	5.4940 ± 0.0949
CARP (600k)	1.5033 ± 0.0132	11.6508 ± 0.0983	1.5659 ± 0.0472	11.2536 ± 0.1272
CARP (38M)	1.5139 ± 0.0088	11.6778 ± 0.1284	1.5768 ± 0.0059	11.0896 ± 0.1535
CARP (76M)	1.4884 ± 0.0145	11.5002 ± 0.2140	1.5607 ± 0.0031	9.1447 ± 0.1564
CARP	1.5070 ± 0.0082	11.7667 ± 0.2218	1.5402 ± 0.0041	9.4377 ± 0.2619
CNN	2.0506 ± 0.2307	13.8411 ± 1.6964	1.3523 ± 0.0539	4.0959 ± 0.1426
TAPE (CNN)	2.0937 ± 0.1898	15.5570 ± 0.5312	1.1251 ± 0.0631	4.9553 ± 0.6760
ESM-1b (CNN)	1.8592 ± 0.1246	14.7160 ± 1.1399	1.1482 ± 0.0239	4.6976 ± 0.4697
ESM-1v (CNN)	2.4676 ± 0.2503	15.0658 ± 1.1423	1.1188 ± 0.0212	5.6638 ± 0.6273
ESM-2 (8M, CNN)	2.3919 ± 0.1627	15.5196 ± 0.4909	1.0913 ± 0.0416	4.8430 ± 0.5947
ESM-2 (35M, CNN)	2.4945 ± 0.4818	14.7227 ± 0.6925	1.0381 ± 0.0769	3.8103 ± 0.1281
ESM-2 (150M, CNN)	2.2499 ± 0.5033	15.6027 ± 1.0554	1.1085 ± 0.1243	4.7428 ± 0.4609
ESM-2 (650M, CNN)	1.8934 ± 0.3340	14.2969 ± 0.7712	1.1476 ± 0.0261	4.7025 ± 0.6881
ESM-2 (CNN)	1.9511 ± 0.1986	14.2011 ± 1.6117	1.3850 ± 0.0382	5.3430 ± 0.8577
ProtT5 (CNN)	1.8554 ± 0.0745	14.9394 ± 0.4143	1.0373 ± 0.0246	5.0151 ± 1.0703
CARP (600k, CNN)	2.6833 ± 0.3544	14.1731 ± 0.3073	1.2257 ± 0.0350	5.2804 ± 0.9685
CARP (38M, CNN)	2.0301 ± 0.2404	15.0381 ± 1.5099	1.1312 ± 0.0228	4.6168 ± 0.4402
CARP (76M, CNN)	1.9932 ± 0.0644	14.6097 ± 0.3420	1.1944 ± 0.0091	3.8172 ± 0.2085
CARP (CNN)	1.7704 ± 0.0174	15.0078 ± 0.8024	1.3109 ± 0.0596	5.4695 ± 0.7086
ESM-1v (Augmented)	1.4845 ± 0.0018	11.7308 ± 0.2460	1.5685 ± 0.0150	10.0828 ± 0.1070
ESM-2 (8M, Augmented)	1.5088 ± 0.0170	11.9113 ± 0.0402	1.6641 ± 0.0156	10.0938 ± 0.0589
ESM-2 (35M, Augmented)	1.5102 ± 0.0220	11.9370 ± 0.0565	1.6291 ± 0.0127	10.1836 ± 0.1647
ESM-2 (150M, Augmented)	1.5230 ± 0.0233	11.9377 ± 0.1009	1.5985 ± 0.0040	9.7887 ± 0.1380
ESM-2 (650M, Augmented)	1.4985 ± 0.0086	11.8261 ± 0.0717	1.6410 ± 0.0120	9.6624 ± 0.1501
ESM-2 (Augmented)	1.5115 ± 0.0318	11.7135 ± 0.1641	1.6180 ± 0.0112	9.6967 ± 0.0494
CARP (600k, Augmented)	**1.4652 ± 0.0021**	11.9570 ± 0.0983	1.5064 ± 0.0039	9.9571 ± 0.1009
CARP (38M, Augmented)	1.4777 ± 0.0186	11.9582 ± 0.0450	1.4982 ± 0.0025	9.9154 ± 0.0556
CARP (76M, Augmented)	1.4757 ± 0.0199	11.7828 ± 0.1579	1.4985 ± 0.0058	9.8734 ± 0.0189
CARP (Augmented)	1.4660 ± 0.0022	12.0082 ± 0.0337	1.5083 ± 0.0061	9.9470 ± 0.0716
EVMutation (Augmented)	1.4708 ± 0.0093	11.6674 ± 0.3938	1.4526 ± 0.0022	7.0896 ± 0.0080
ESM-2 (8M, Finetuned)	1.5318 ± 0.0264	14.9039 ± 2.5033	1.6052 ± 0.0357	4.7209 ± 0.2025
ESM-2 (35M, Finetuned)	1.5500 ± 0.0967	13.5396 ± 0.3245	1.3019 ± 0.0551	3.7613 ± 0.0887
ESM-2 (150M, Finetuned)	1.5245 ± 0.0026	11.4452 ± 0.2846	1.1094 ± 0.0467	**3.3694 ± 0.1247**
ESM-2 (8M, CNN, Finetuned)	2.0855 ± 0.3826	15.5261 ± 0.3309	1.0087 ± 0.0216	4.5865 ± 0.4793
ESM-2 (35M, CNN, Finetuned)	2.2368 ± 0.4315	13.8723 ± 0.8406	**0.8978 ± 0.0305**	3.5521 ± 0.3629
ESM-2 (150M, CNN, Finetuned)	2.8614 ± 0.2362	14.9521 ± 0.7068	1.0092 ± 0.1637	3.9133 ± 0.2355

Table B.1: Raw Model Performance on GB1 Datasets (one-vs-rest and two-vs-rest only). Performance measured by mean-squared-error (no FLIGHTED) or weighted mean-squared-error (FLIGHTED). Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	Three vs Rest	Three vs Rest FLIGHTED	Low vs High	Low vs High FLIGHTED
Linear	1.5172 ± 0.0049	8.1622 ± 0.0122	4.6324 ± 0.0126	7.5193 ± 0.1734
TAPE	0.9591 ± 0.0295	3.6993 ± 0.0375	3.5252 ± 0.0394	2.4557 ± 0.1641
ESM-1b	0.9484 ± 0.0104	3.2003 ± 0.0183	3.5314 ± 0.0218	2.5016 ± 0.1014
ESM-1v	0.9261 ± 0.0185	3.4494 ± 0.0882	3.4070 ± 0.0201	1.7771 ± 0.0768
ESM-2 (8M)	1.0639 ± 0.0142	4.3834 ± 0.0081	3.6068 ± 0.0302	3.4855 ± 0.0535
ESM-2 (35M)	0.9534 ± 0.0022	3.8483 ± 0.0362	3.5095 ± 0.0738	2.9552 ± 0.0789
ESM-2 (150M)	0.9726 ± 0.0052	4.3019 ± 0.1824	3.5763 ± 0.0271	3.0476 ± 0.0451
ESM-2 (650M)	0.7834 ± 0.0036	2.9651 ± 0.0779	3.4553 ± 0.0089	2.3014 ± 0.0535
ESM-2	0.8850 ± 0.0038	2.8603 ± 0.0090	3.4237 ± 0.0502	2.1913 ± 0.0138
ProtT5	0.8706 ± 0.0072	3.0169 ± 0.0259	3.3162 ± 0.0289	2.0257 ± 0.0282
CARP (600k)	1.1682 ± 0.0112	5.5622 ± 0.0866	3.9904 ± 0.0258	4.4635 ± 0.1299
CARP (38M)	1.3215 ± 0.0152	6.4299 ± 0.1360	4.2556 ± 0.0272	5.4571 ± 0.0665
CARP (76M)	1.1813 ± 0.0074	5.3000 ± 0.1357	4.1310 ± 0.0059	4.7139 ± 0.2040
CARP	1.2078 ± 0.0052	5.0631 ± 0.2182	4.1793 ± 0.0399	5.0080 ± 0.1777
CNN	0.5158 ± 0.0101	1.5133 ± 0.0458	3.1096 ± 0.0319	**0.6077 ± 0.0747**
TAPE (CNN)	0.4504 ± 0.0039	1.2584 ± 0.0790	3.1693 ± 0.0720	0.7386 ± 0.0647
ESM-1b (CNN)	0.4406 ± 0.0066	1.2828 ± 0.0583	3.2526 ± 0.1097	0.8580 ± 0.1646
ESM-1v (CNN)	**0.4218 ± 0.0148**	1.0939 ± 0.0647	3.2323 ± 0.1479	0.8102 ± 0.0425
ESM-2 (8M, CNN)	0.4587 ± 0.0071	1.3937 ± 0.1154	**3.1004 ± 0.0596**	0.8299 ± 0.0669
ESM-2 (35M, CNN)	0.4736 ± 0.0061	1.3928 ± 0.0585	3.1440 ± 0.0846	0.6806 ± 0.1123
ESM-2 (150M, CNN)	0.4724 ± 0.0051	1.3727 ± 0.0349	3.1400 ± 0.0445	0.7269 ± 0.1354
ESM-2 (650M, CNN)	0.4396 ± 0.0515	1.2475 ± 0.1657	3.3286 ± 0.0471	0.8302 ± 0.0805
ESM-2 (CNN)	0.4440 ± 0.0259	1.2242 ± 0.1160	3.3670 ± 0.1110	0.8903 ± 0.0651
ProtT5 (CNN)	0.4576 ± 0.0232	1.1711 ± 0.0338	3.2488 ± 0.0403	0.8296 ± 0.0278
CARP (600k, CNN)	0.5965 ± 0.0216	1.7806 ± 0.0977	3.2274 ± 0.0671	0.9752 ± 0.1267
CARP (38M, CNN)	0.4854 ± 0.0094	1.1501 ± 0.0552	3.1777 ± 0.0413	0.8558 ± 0.0859
CARP (76M, CNN)	0.5070 ± 0.0409	1.2360 ± 0.0822	3.2070 ± 0.0067	0.8577 ± 0.0944
CARP (CNN)	0.4661 ± 0.0215	1.3619 ± 0.1444	3.2760 ± 0.0515	0.7709 ± 0.1090
ESM-1v (Augmented)	1.4944 ± 0.0063	7.9110 ± 0.1260	4.6129 ± 0.0221	7.1360 ± 0.1249
ESM-2 (8M, Augmented)	1.5203 ± 0.0064	7.9353 ± 0.0718	4.6053 ± 0.0032	7.3604 ± 0.1191
ESM-2 (35M, Augmented)	1.4982 ± 0.0024	7.9133 ± 0.0427	4.6106 ± 0.0130	7.3177 ± 0.0638
ESM-2 (150M, Augmented)	1.4950 ± 0.0080	7.6113 ± 0.0829	4.5462 ± 0.0280	6.9321 ± 0.0473
ESM-2 (650M, Augmented)	1.5175 ± 0.0049	7.3998 ± 0.1201	4.5873 ± 0.0184	6.8160 ± 0.0256
ESM-2 (Augmented)	1.5265 ± 0.0051	7.7374 ± 0.1413	4.6023 ± 0.0446	7.1383 ± 0.0245
CARP (600k, Augmented)	1.5168 ± 0.0064	8.0217 ± 0.0664	4.5907 ± 0.0208	7.4045 ± 0.1267
CARP (38M, Augmented)	1.5153 ± 0.0088	8.0537 ± 0.0542	4.5971 ± 0.0493	7.4798 ± 0.0527
CARP (76M, Augmented)	1.5177 ± 0.0027	8.2109 ± 0.1684	4.6521 ± 0.0142	7.5263 ± 0.0519
CARP (Augmented)	1.5186 ± 0.0034	8.2201 ± 0.0617	4.6398 ± 0.0729	7.5125 ± 0.1570
EVMutation (Augmented)	1.5126 ± 0.0068	6.6964 ± 0.0356	4.5886 ± 0.0487	7.2021 ± 0.0813
ESM-2 (8M, Finetuned)	0.5035 ± 0.0127	**0.9590 ± 0.0316**	3.4907 ± 0.0767	1.0527 ± 0.0363
ESM-2 (35M, Finetuned)	0.5780 ± 0.0566	1.1488 ± 0.0564	3.5533 ± 0.0036	1.0101 ± 0.0532
ESM-2 (150M, Finetuned)	0.4928 ± 0.0451	1.0775 ± 0.1876	3.6195 ± 0.0081	1.0493 ± 0.0379
ESM-2 (8M, CNN, Finetuned)	0.4613 ± 0.0162	1.0441 ± 0.0517	3.2580 ± 0.0426	0.9389 ± 0.0713
ESM-2 (35M, CNN, Finetuned)	0.4359 ± 0.0166	1.0510 ± 0.0885	3.2956 ± 0.0378	0.7850 ± 0.1027
ESM-2 (150M, CNN, Finetuned)	0.4476 ± 0.0211	1.0069 ± 0.0271	3.2438 ± 0.0467	0.6934 ± 0.0123

Table B.2: Raw Model Performance on GB1 Datasets (three-vs-rest and low-vs-high only). Performance measured by mean-squared-error (no FLIGHTED) or weighted mean-squared-error (FLIGHTED). Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	One vs Rest	One vs Rest FLIGHTED	Two vs Rest	Two vs Rest FLIGHTED
Linear	0.1428 ± 0.0083	0.1218 ± 0.0373	0.3055 ± 0.0339	0.4681 ± 0.0025
TAPE	-0.2465 ± 0.0258	0.2353 ± 0.0216	0.3194 ± 0.0133	0.5384 ± 0.0039
ESM-1b	0.2537 ± 0.0124	-0.1745 ± 0.0224	0.3992 ± 0.0020	0.5478 ± 0.0019
ESM-1v	-0.1011 ± 0.0128	0.1355 ± 0.0009	0.3939 ± 0.0106	0.4209 ± 0.0141
ESM-2 (8M)	-0.0142 ± 0.0578	0.0888 ± 0.0181	0.4168 ± 0.0005	0.5520 ± 0.0027
ESM-2 (35M)	0.0776 ± 0.0451	-0.0452 ± 0.0336	0.4471 ± 0.0011	0.5744 ± 0.0018
ESM-2 (150M)	-0.0879 ± 0.0381	0.0695 ± 0.0142	0.3865 ± 0.0008	0.4818 ± 0.0028
ESM-2 (650M)	-0.0698 ± 0.0016	0.0973 ± 0.0110	0.4562 ± 0.0047	0.6040 ± 0.0026
ESM-2	-0.0393 ± 0.0043	0.0028 ± 0.0353	0.4159 ± 0.0030	0.5801 ± 0.0022
ProtT5	0.2412 ± 0.0343	-0.2332 ± 0.0097	0.4754 ± 0.0022	0.5738 ± 0.0027
CARP (600k)	0.0672 ± 0.1292	0.2548 ± 0.0061	0.3957 ± 0.0048	0.4199 ± 0.0198
CARP (38M)	-0.0006 ± 0.2014	0.0726 ± 0.2527	0.2398 ± 0.0130	0.3346 ± 0.0113
CARP (76M)	-0.0193 ± 0.1066	-0.0585 ± 0.1683	0.1985 ± 0.0116	0.3452 ± 0.0401
CARP	0.0112 ± 0.0523	-0.1820 ± 0.1152	0.2402 ± 0.0102	0.3322 ± 0.0197
CNN	0.1124 ± 0.0501	0.0815 ± 0.0751	0.4918 ± 0.0056	0.6216 ± 0.0367
TAPE (CNN)	0.1654 ± 0.1318	0.0630 ± 0.0566	0.5840 ± 0.0083	0.5910 ± 0.0549
ESM-1b (CNN)	0.1275 ± 0.0760	0.0358 ± 0.0291	0.5196 ± 0.0239	0.5804 ± 0.0306
ESM-1v (CNN)	0.0760 ± 0.0085	0.0097 ± 0.0260	0.5353 ± 0.0138	0.5673 ± 0.0421
ESM-2 (8M, CNN)	0.1909 ± 0.0185	0.1202 ± 0.1170	0.5645 ± 0.0059	0.6326 ± 0.0157
ESM-2 (35M, CNN)	0.1777 ± 0.0303	0.1849 ± 0.1015	0.6014 ± 0.0146	0.6552 ± 0.0132
ESM-2 (150M, CNN)	0.1018 ± 0.0466	0.0066 ± 0.0263	0.5511 ± 0.0384	0.6365 ± 0.0107
ESM-2 (650M, CNN)	0.0507 ± 0.0019	0.0022 ± 0.0228	0.5225 ± 0.0077	0.6108 ± 0.0071
ESM-2 (CNN)	0.0211 ± 0.0255	0.0084 ± 0.0164	0.3205 ± 0.0559	0.5280 ± 0.0188
ProtT5 (CNN)	0.1067 ± 0.0613	-0.0016 ± 0.0089	0.5827 ± 0.0256	0.5564 ± 0.0586
CARP (600k, CNN)	0.1611 ± 0.0063	0.0639 ± 0.0542	0.5221 ± 0.0153	0.6083 ± 0.0299
CARP (38M, CNN)	0.0298 ± 0.0379	0.0019 ± 0.0109	0.5502 ± 0.0154	0.6350 ± 0.0076
CARP (76M, CNN)	0.0048 ± 0.0116	0.0006 ± 0.0119	0.4939 ± 0.0050	0.6206 ± 0.0128
CARP (CNN)	-0.0026 ± 0.0126	0.0014 ± 0.0040	0.3998 ± 0.0402	0.5641 ± 0.0220
ESM-1v (Augmented)	0.0295 ± 0.0143	0.1713 ± 0.0432	0.0877 ± 0.0045	0.4098 ± 0.0137
ESM-2 (8M, Augmented)	-0.0114 ± 0.0062	0.1475 ± 0.0769	0.0148 ± 0.0048	0.3774 ± 0.0143
ESM-2 (35M, Augmented)	0.0371 ± 0.0496	0.1365 ± 0.0321	0.0154 ± 0.0043	0.4099 ± 0.0169
ESM-2 (150M, Augmented)	-0.0089 ± 0.0411	0.1012 ± 0.0720	-0.0211 ± 0.0076	0.4941 ± 0.0107
ESM-2 (650M, Augmented)	-0.0612 ± 0.0247	0.0949 ± 0.0814	-0.0427 ± 0.0041	0.4955 ± 0.0136
ESM-2 (Augmented)	-0.0463 ± 0.0333	0.2230 ± 0.0219	0.0032 ± 0.0077	0.4872 ± 0.0089
CARP (600k, Augmented)	0.1347 ± 0.0061	0.1247 ± 0.0272	0.3124 ± 0.0136	0.4690 ± 0.0054
CARP (38M, Augmented)	0.1475 ± 0.0119	0.1111 ± 0.0262	0.3058 ± 0.0068	0.4747 ± 0.0083
CARP (76M, Augmented)	0.1481 ± 0.0378	0.0914 ± 0.0104	0.3025 ± 0.0042	0.4639 ± 0.0061
CARP (Augmented)	0.1205 ± 0.0035	0.1278 ± 0.0094	0.2977 ± 0.0048	0.4636 ± 0.0092
EVMutation (Augmented)	0.0723 ± 0.0273	0.1590 ± 0.0542	0.3085 ± 0.0154	0.4485 ± 0.0061
ESM-2 (8M, Finetuned)	-0.0172 ± 0.0554	0.2379 ± 0.1502	0.4994 ± 0.0351	0.5598 ± 0.0140
ESM-2 (35M, Finetuned)	**0.3776 ± 0.0015**	**0.4124 ± 0.0006**	0.6026 ± 0.0023	0.6345 ± 0.0043
ESM-2 (150M, Finetuned)	-0.0523 ± 0.0178	0.0779 ± 0.0132	0.5747 ± 0.0169	0.6480 ± 0.0009
ESM-2 (8M, CNN, Finetuned)	0.1486 ± 0.0559	0.1692 ± 0.0582	0.5766 ± 0.0054	0.5943 ± 0.0250
ESM-2 (35M, CNN, Finetuned)	0.2729 ± 0.0636	0.0939 ± 0.0360	**0.6323 ± 0.0117**	0.6665 ± 0.0140
ESM-2 (150M, CNN, Finetuned)	0.1318 ± 0.0330	0.0493 ± 0.0883	0.5869 ± 0.0413	**0.6758 ± 0.0206**

Table B.3: Raw Model Performance on GB1 Datasets (one-vs-rest and two-vs-rest only). Performance measured by weighted Spearman R. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	Three vs Rest	Three vs Rest FLIGHTED	Low vs High	Low vs High FLIGHTED
Linear	0.6215 ± 0.0034	0.7603 ± 0.0012	0.0476 ± 0.0102	0.0556 ± 0.0008
TAPE	0.6362 ± 0.0147	0.7140 ± 0.0031	0.1068 ± 0.0043	0.0908 ± 0.0012
ESM-1b	0.6828 ± 0.0010	0.7464 ± 0.0003	0.1061 ± 0.0006	0.0980 ± 0.0029
ESM-1v	0.6733 ± 0.0031	0.7242 ± 0.0023	0.1257 ± 0.0023	0.1199 ± 0.0045
ESM-2 (8M)	0.6527 ± 0.0014	0.6866 ± 0.0021	0.0999 ± 0.0006	0.0549 ± 0.0022
ESM-2 (35M)	0.6601 ± 0.0017	0.6925 ± 0.0003	0.1245 ± 0.0032	0.0755 ± 0.0048
ESM-2 (150M)	0.6747 ± 0.0008	0.7030 ± 0.0016	0.1038 ± 0.0024	0.0897 ± 0.0014
ESM-2 (650M)	0.7281 ± 0.0007	0.7548 ± 0.0007	0.1314 ± 0.0020	0.1089 ± 0.0022
ESM-2	0.7046 ± 0.0009	0.7422 ± 0.0016	0.1393 ± 0.0090	0.1192 ± 0.0046
ProtT5	0.7059 ± 0.0005	0.7541 ± 0.0009	0.1658 ± 0.0023	0.1110 ± 0.0018
CARP (600k)	0.5973 ± 0.0004	0.6304 ± 0.0037	0.0639 ± 0.0011	0.0536 ± 0.0040
CARP (38M)	0.4997 ± 0.0055	0.5451 ± 0.0039	0.0391 ± 0.0010	0.0362 ± 0.0010
CARP (76M)	0.6070 ± 0.0016	0.6292 ± 0.0040	0.0533 ± 0.0025	0.0441 ± 0.0014
CARP	0.6094 ± 0.0037	0.6664 ± 0.0010	0.0402 ± 0.0025	0.0345 ± 0.0011
CNN	0.7636 ± 0.0010	0.7987 ± 0.0013	0.2034 ± 0.0064	**0.2253 ± 0.0052**
TAPE (CNN)	0.7719 ± 0.0051	0.8047 ± 0.0006	0.1704 ± 0.0202	0.2031 ± 0.0124
ESM-1b (CNN)	0.7767 ± 0.0007	0.7965 ± 0.0013	0.1558 ± 0.0246	0.1901 ± 0.0269
ESM-1v (CNN)	0.7846 ± 0.0018	0.8023 ± 0.0039	0.1554 ± 0.0393	0.1622 ± 0.0305
ESM-2 (8M, CNN)	0.7688 ± 0.0050	0.7989 ± 0.0035	0.2127 ± 0.0039	0.2079 ± 0.0026
ESM-2 (35M, CNN)	0.7650 ± 0.0020	0.7982 ± 0.0020	0.1910 ± 0.0130	0.2166 ± 0.0154
ESM-2 (150M, CNN)	0.7674 ± 0.0026	0.8036 ± 0.0010	**0.2159 ± 0.0049**	0.2232 ± 0.0074
ESM-2 (650M, CNN)	0.7833 ± 0.0020	0.7988 ± 0.0032	0.1438 ± 0.0351	0.1650 ± 0.0285
ESM-2 (CNN)	0.7791 ± 0.0022	0.8020 ± 0.0016	0.1392 ± 0.0495	0.1790 ± 0.0184
ProtT5 (CNN)	0.7799 ± 0.0094	0.8005 ± 0.0022	0.1222 ± 0.0241	0.1467 ± 0.0248
CARP (600k, CNN)	0.7518 ± 0.0024	0.7840 ± 0.0012	0.1745 ± 0.0079	0.2005 ± 0.0080
CARP (38M, CNN)	0.7603 ± 0.0069	0.7997 ± 0.0015	0.1895 ± 0.0225	0.1359 ± 0.0350
CARP (76M, CNN)	0.7706 ± 0.0035	0.7964 ± 0.0055	0.1760 ± 0.0231	0.1582 ± 0.0405
CARP (CNN)	0.7714 ± 0.0108	0.7983 ± 0.0025	0.1626 ± 0.0279	0.1861 ± 0.0138
ESM-1v (Augmented)	0.6105 ± 0.0134	0.7043 ± 0.0081	0.0801 ± 0.0084	0.0840 ± 0.0017
ESM-2 (8M, Augmented)	0.6101 ± 0.0149	0.7438 ± 0.0015	0.0613 ± 0.0065	0.0650 ± 0.0032
ESM-2 (35M, Augmented)	0.6273 ± 0.0110	0.7132 ± 0.0023	0.0604 ± 0.0079	0.0736 ± 0.0010
ESM-2 (150M, Augmented)	0.6342 ± 0.0111	0.6641 ± 0.0029	0.0768 ± 0.0067	0.0797 ± 0.0024
ESM-2 (650M, Augmented)	0.6281 ± 0.0163	0.6100 ± 0.0015	0.0296 ± 0.0069	0.0359 ± 0.0030
ESM-2 (Augmented)	0.6055 ± 0.0152	0.6769 ± 0.0017	0.0430 ± 0.0091	0.0573 ± 0.0012
CARP (600k, Augmented)	0.6253 ± 0.0071	0.7610 ± 0.0023	0.0646 ± 0.0114	0.0535 ± 0.0007
CARP (38M, Augmented)	0.6130 ± 0.0171	0.7594 ± 0.0016	0.0626 ± 0.0163	0.0548 ± 0.0026
CARP (76M, Augmented)	0.6198 ± 0.0110	0.7598 ± 0.0044	0.0374 ± 0.0030	0.0574 ± 0.0013
CARP (Augmented)	0.6152 ± 0.0099	0.7600 ± 0.0025	0.0525 ± 0.0055	0.0559 ± 0.0018
EVMutation (Augmented)	0.6154 ± 0.0133	0.7625 ± 0.0026	0.0153 ± 0.0038	0.0214 ± 0.0007
ESM-2 (8M, Finetuned)	**0.7959 ± 0.0081**	0.7967 ± 0.0022	0.1618 ± 0.0114	0.1488 ± 0.0438
ESM-2 (35M, Finetuned)	0.7828 ± 0.0068	0.7952 ± 0.0007	0.1646 ± 0.0115	0.1804 ± 0.0902
ESM-2 (150M, Finetuned)	0.7858 ± 0.0049	0.7981 ± 0.0032	0.1329 ± 0.0042	0.1212 ± 0.0067
ESM-2 (8M, CNN, Finetuned)	0.7789 ± 0.0059	0.8021 ± 0.0010	0.1906 ± 0.0135	0.1077 ± 0.0142
ESM-2 (35M, CNN, Finetuned)	0.7813 ± 0.0039	0.8058 ± 0.0006	0.1551 ± 0.0022	0.1520 ± 0.0188
ESM-2 (150M, CNN, Finetuned)	0.7743 ± 0.0031	**0.8089 ± 0.0033**	0.2013 ± 0.0070	0.1901 ± 0.0148

Table B.4: Raw Model Performance on GB1 Datasets (three-vs-rest and low-vs-high only). Performance measured by weighted Spearman R. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	One vs Rest	Two vs Rest	Three vs Rest
Linear	526.0933 ± 22.7232	473.0123 ± 30.1776	61.9535 ± 9.9049
TAPE	297.5274 ± 16.9196	42.1769 ± 5.7037	10.1017 ± 2.9131
ESM-1b	165.9262 ± 0.8583	19.2814 ± 2.1384	4.7189 ± 0.0789
ESM-1v	**96.5753 ± 4.8990**	31.6219 ± 8.8151	5.0249 ± 0.2268
ESM-2 (8M)	487.1830 ± 67.0827	77.3104 ± 13.6258	26.0598 ± 11.2005
ESM-2 (35M)	201.0534 ± 14.7650	38.3270 ± 14.7239	5.3898 ± 0.0861
ESM-2 (150M)	253.0745 ± 17.7263	21.2191 ± 4.1567	5.8834 ± 0.7763
ESM-2 (650M)	129.0170 ± 3.0636	15.7421 ± 2.0325	5.0022 ± 0.5990
ESM-2	455.1378 ± 8.2257	21.5850 ± 7.1289	16.4862 ± 17.6678
ProtT5	128.9400 ± 5.0140	16.2017 ± 2.1542	4.5883 ± 0.2502
CARP (600k)	370.3106 ± 61.6088	67.1536 ± 5.6832	17.8028 ± 0.6229
CARP (38M)	779.2183 ± 6.6225	69.2424 ± 6.2080	11.5826 ± 1.0835
CARP (76M)	293.7587 ± 52.0933	59.6290 ± 2.5587	14.5622 ± 0.9285
CARP	661.4456 ± 227.2850	201.7350 ± 12.4057	25.1076 ± 8.3689
CNN	613.7185 ± 242.4505	169.1024 ± 7.8148	6.4333 ± 0.1398
TAPE (CNN)	535.2333 ± 132.6787	119.1340 ± 10.3364	5.5736 ± 0.2825
ESM-1b (CNN)	565.8093 ± 203.6468	40.1009 ± 9.9084	4.5211 ± 0.0431
ESM-1v (CNN)	678.3326 ± 164.8031	28.9555 ± 0.4254	4.7168 ± 0.1845
ESM-2 (8M, CNN)	537.5033 ± 238.1350	56.3152 ± 9.9305	5.3183 ± 0.1928
ESM-2 (35M, CNN)	299.9104 ± 14.8647	26.8592 ± 5.2674	224.8132 ± 191.9686
ESM-2 (150M, CNN)	676.5742 ± 103.0750	31.0988 ± 5.3971	4.7040 ± 0.1121
ESM-2 (650M, CNN)	915.3362 ± 23.6130	66.0526 ± 3.5709	4.8076 ± 0.0419
ESM-2 (CNN)	1361.5717 ± 539.0819	87.6785 ± 10.3623	4.9448 ± 0.1656
ProtT5 (CNN)	900.8455 ± 233.7611	61.6325 ± 3.8367	4.7374 ± 0.0591
CARP (600k, CNN)	479.3737 ± 9.2137	69.0239 ± 4.6387	6.8296 ± 0.4412
CARP (38M, CNN)	685.8954 ± 281.9056	79.6063 ± 9.2690	4.8927 ± 0.0288
CARP (76M, CNN)	581.1124 ± 86.5494	64.4380 ± 2.4957	4.7728 ± 0.1322
CARP (CNN)	797.2115 ± 459.2779	40.6028 ± 0.7333	4.6357 ± 0.1348
ESM-1v (Augmented)	238.4065 ± 5.2917	480.2571 ± 33.7659	63.6243 ± 3.8785
ESM-2 (8M, Augmented)	337.0354 ± 13.8129	492.2340 ± 28.7706	69.7858 ± 4.0155
ESM-2 (35M, Augmented)	216.1071 ± 23.8610	472.1725 ± 53.4941	71.2082 ± 6.5696
ESM-2 (150M, Augmented)	222.0371 ± 4.0988	490.2639 ± 30.5448	61.4350 ± 7.4693
ESM-2 (650M, Augmented)	338.1440 ± 190.0764	481.7004 ± 19.7602	69.2233 ± 11.1288
ESM-2 (Augmented)	226.2928 ± 33.0988	507.1823 ± 49.9358	69.0547 ± 3.8129
CARP (600k, Augmented)	555.9357 ± 65.4261	469.3298 ± 3.0498	65.6611 ± 1.8785
CARP (38M, Augmented)	608.4704 ± 69.0148	449.0583 ± 18.4007	61.7768 ± 5.5423
CARP (76M, Augmented)	562.6250 ± 15.5729	464.0757 ± 13.7924	64.1665 ± 7.0719
CARP (Augmented)	508.6772 ± 74.7882	459.4699 ± 36.2165	66.3554 ± 4.3394
EVMutation (Augmented)	247.1387 ± 20.9441	492.7133 ± 31.5477	62.2780 ± 6.3277
ESM-2 (8M, Finetuned)	260.1472 ± 66.3833	7.6464 ± 1.6408	4.2765 ± 0.0316
ESM-2 (35M, Finetuned)	273.4673 ± 15.2778	**7.5348 ± 0.3203**	4.2672 ± 0.0318
ESM-2 (8M, CNN, Finetuned)	517.6429 ± 199.1142	19.5290 ± 1.4134	4.7530 ± 0.1151
ESM-2 (35M, CNN, Finetuned)	211.5166 ± 46.2842	10.7844 ± 0.5173	**4.2564 ± 0.0468**

Table B.5: Raw Model Performance on TEV 4-Site Datasets. Performance measured by weighted mean-squared-error. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	One vs Rest	Two vs Rest	Three vs Rest
Linear	236.9918 ± 0.2446	39.3453 ± 1.0812	6.8516 ± 0.1647
TAPE	215.1885 ± 1.8393	29.3225 ± 0.9667	8.0847 ± 0.1294
ESM-1b	109.6655 ± 0.5158	27.6254 ± 1.0074	8.0732 ± 1.9349
ESM-1v	168.9161 ± 2.6093	25.2212 ± 0.9906	6.6314 ± 0.1742
ESM-2 (8M)	310.3055 ± 3.6748	42.0031 ± 4.1027	7.8548 ± 0.1206
ESM-2 (35M)	324.9537 ± 3.7162	40.9668 ± 1.0532	8.5839 ± 0.8350
ESM-2 (150M)	207.1491 ± 2.4796	50.2741 ± 3.6836	8.7751 ± 1.2895
ESM-2 (650M)	236.9972 ± 3.8307	32.7483 ± 2.7507	7.8286 ± 0.6503
ESM-2	348.7033 ± 1.3068	62.1407 ± 1.3097	13.3675 ± 5.4377
ProtT5	78.3772 ± 1.6579	38.0986 ± 0.2578	7.9934 ± 0.6638
CARP (600k)	249.8514 ± 5.7466	36.9001 ± 1.2421	9.6572 ± 0.3606
CARP (38M)	318.9603 ± 13.9792	44.8450 ± 1.1494	7.7503 ± 0.9770
CARP (76M)	92.6920 ± 4.0376	36.2969 ± 1.6000	7.2260 ± 0.3863
CARP	**32.5095 ± 1.9709**	130.5106 ± 3.7442	9.2733 ± 0.4920
CNN	261.0534 ± 6.6962	47.8802 ± 9.0689	**6.0198 ± 0.3099**
TAPE (CNN)	178.0944 ± 29.2056	36.2829 ± 4.5467	8.7940 ± 1.9774
ESM-1b (CNN)	112.9677 ± 10.2470	17.0389 ± 2.7825	10.6511 ± 1.7873
ESM-1v (CNN)	101.8298 ± 5.4731	14.8479 ± 0.8872	13.3951 ± 2.6727
ESM-2 (8M, CNN)	343.5329 ± 79.1888	31.4367 ± 2.3780	10.4376 ± 4.1407
ESM-2 (35M, CNN)	118.4350 ± 18.2043	**14.0074 ± 0.5404**	9.4401 ± 1.0544
ESM-2 (150M, CNN)	236.4661 ± 27.6450	22.3992 ± 4.8395	13.5230 ± 4.8441
ESM-2 (650M, CNN)	263.2625 ± 34.4840	18.8169 ± 2.6014	12.0253 ± 1.1506
ESM-2 (CNN)	256.4440 ± 20.1821	14.9588 ± 2.0987	10.4532 ± 0.1970
ProtT5 (CNN)	355.1115 ± 6.7419	23.0450 ± 2.1066	10.2802 ± 4.3830
CARP (600k, CNN)	184.5594 ± 53.4166	52.7369 ± 19.3864	6.8550 ± 0.3311
CARP (38M, CNN)	210.0242 ± 16.8956	46.5654 ± 6.9319	14.1888 ± 2.0797
CARP (76M, CNN)	133.0583 ± 19.6558	29.1659 ± 3.1595	12.5479 ± 4.1431
CARP (CNN)	136.7413 ± 10.0702	20.8244 ± 2.2170	10.4878 ± 0.6368

Table B.6: Raw Model Performance on TEV 4-Site Datasets (C⟶T counts). Performance measured by weighted mean-squared-error, weighted by the FLIGHTED variance. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	Linear	ESM-2 (8M, CNN)						
Reweighting	False	False	True					
Power	-	-	10	100	1000	10000	100000	1000000
One vs Rest	15.9254 ± 6.0082	20.5246 ± 3.1455	-	-	-	-	-	-
Two vs Rest	44.2373 ± 6.5050	29.0849 ± 3.1351	-	-	-	-	-	-
Three vs Rest	11.6440 ± 0.6029	17.9117 ± 4.4371	-	-	-	-	-	-
Four vs Rest	6.8612 ± 0.0619	7.8594 ± 0.4878	-	-	-	-	-	-
Five vs Rest	6.4253 ± 0.0398	5.6542 ± 0.0771	-	-	-	-	-	-
Six vs Rest	6.1556 ± 0.1301	4.4504 ± 0.0371	-	-	-	-	-	-
Seven vs Rest	5.8075 ± 0.0404	4.1527 ± 0.0134	4.1948 ± 0.0504	4.5336 ± 0.0193	5.0859 ± 0.2244	5.7877 ± 0.4546	8.2117 ± 0.1684	12.9117 ± 1.4376
Seven (1%) vs Rest	5.8874 ± 0.0695	5.6459 ± 0.1163	-	-	-	-	-	-
Seven (3%) vs Rest	5.7795 ± 0.0058	5.1785 ± 0.1090	-	-	-	-	-	-
Seven (10%) vs Rest	5.7895 ± 0.0705	4.8516 ± 0.0952	-	-	-	-	-	-
Seven (30%) vs Rest	5.7907 ± 0.0855	4.3563 ± 0.0494	-	-	-	-	-	-

Table B.7: Raw Model Performance on TEV 8-Site Datasets. Performance measured by weighted mean-squared-error.

Model	Linear	ESM-2 (8M, CNN)						
Reweighting	False	False	True					
Power	-	-	10	100	1000	10000	100000	1000000
One vs Rest	10.8884 ± 3.0362	12.0944 ± 2.7172	-	-	-	-	-	-
Two vs Rest	6.3825 ± 2.1843	4.5157 ± 1.1055	-	-	-	-	-	-
Three vs Rest	4.6325 ± 0.5049	3.9521 ± 2.5688	-	-	-	-	-	-
Four vs Rest	12.4844 ± 3.0117	6.4810 ± 1.1226	-	-	-	-	-	-
Five vs Rest	13.1286 ± 0.2584	9.7519 ± 0.0692	-	-	-	-	-	-
Six vs Rest	13.7400 ± 0.7865	8.9992 ± 1.1163	-	-	-	-	-	-
Seven vs Rest	12.0764 ± 1.3297	8.2407 ± 0.4482	6.3710 ± 0.1863	5.3681 ± 0.4166	4.4735 ± 0.5814	4.0755 ± 0.3264	2.7912 ± 0.1785	2.2545 ± 0.1010
Seven (1%) vs Rest	13.1444 ± 0.5459	13.6148 ± 2.1030	-	-	-	-	-	-
Seven (3%) vs Rest	12.1366 ± 0.2867	10.6379 ± 1.9078	-	-	-	-	-	-
Seven (10%) vs Rest	12.3989 ± 0.4959	10.2774 ± 1.0296	-	-	-	-	-	-
Seven (30%) vs Rest	12.6306 ± 1.4502	9.3923 ± 0.3266	-	-	-	-	-	-

Table B.8: Raw Model Performance on TEV 8-Site Datasets. Performance measured by active weighted mean-squared-error (MSE on the top 5% of actives).

Model	Linear	ESM-2 (8M, CNN)						
Reweighting	False	False	True					
Power	-	-	10	100	1000	10000	100000	1000000
One vs Rest	0.6145 ± 0.0109	0.5346 ± 0.0401	-	-	-	-	-	-
Two vs Rest	0.7050 ± 0.0113	0.5411 ± 0.0218	-	-	-	-	-	-
Three vs Rest	0.7178 ± 0.0030	0.7134 ± 0.0066	-	-	-	-	-	-
Four vs Rest	0.7190 ± 0.0030	0.7217 ± 0.0010	-	-	-	-	-	-
Five vs Rest	0.7193 ± 0.0029	0.7234 ± 0.0013	-	-	-	-	-	-
Six vs Rest	0.7234 ± 0.0034	0.7141 ± 0.0045	-	-	-	-	-	-
Seven vs Rest	0.7231 ± 0.0016	0.7244 ± 0.0039	0.7206 ± 0.0034	0.7194 ± 0.0030	0.7192 ± 0.0046	0.7203 ± 0.0027	0.7144 ± 0.0024	0.7160 ± 0.0013
Seven (1%) vs Rest	0.7227 ± 0.0027	0.7241 ± 0.0007	-	-	-	-	-	-
Seven (3%) vs Rest	0.7239 ± 0.0010	0.7213 ± 0.0025	-	-	-	-	-	-
Seven (10%) vs Rest	0.7243 ± 0.0020	0.7254 ± 0.0016	-	-	-	-	-	-
Seven (30%) vs Rest	0.7224 ± 0.0015	0.7191 ± 0.0017	-	-	-	-	-	-

Table B.9: Raw Model Performance on TEV 8-Site Datasets. Performance measured by mean fitness of top 100 variants.

Model	Linear		ESM-2 (8M, CNN)					
Fitness Variance Cutoff	0	0.001	0.000000			0.001		
Resampling	False	False	False		True	False		True
Power	100000	100000	100000	100000	1000000	100000	100000	1000000
Task								
One vs Rest	4.8584 ± 0.0490	4.3137 ± 0.0311	3.4443 ± 0.0435	3.5794 ± 0.0983	3.7511 ± 0.3758	3.6820 ± 0.4043	4.9748 ± 0.4921	5.1247 ± 1.7101
Two vs Rest	5.8096 ± 0.1039	4.8287 ± 0.1905	3.4266 ± 0.0134	34.6441 ± 3.0017	50.2668 ± 2.2236	3.4392 ± 0.0276	12.8330 ± 0.1351	10.1106 ± 1.0115
Three vs Rest	4.4688 ± 0.3979	4.2643 ± 0.0647	3.2685 ± 0.0069	32.4970 ± 0.9286	44.9247 ± 1.5287	3.2825 ± 0.0159	18.3184 ± 0.9160	22.8077 ± 1.1908
Four vs Rest	3.9663 ± 0.0951	4.0422 ± 0.1801	3.2808 ± 0.0139	21.3319 ± 0.5927	30.9776 ± 0.7386	3.3468 ± 0.0291	15.3154 ± 0.2567	19.7290 ± 0.8807
Five vs Rest	4.0150 ± 0.1366	3.8971 ± 0.1027	3.2551 ± 0.0261	17.1037 ± 0.4541	25.3334 ± 1.1894	3.2888 ± 0.0090	14.4285 ± 0.4636	19.5965 ± 1.5153
Six vs Rest	3.8220 ± 0.0804	3.9235 ± 0.1254	3.2246 ± 0.0228	14.2060 ± 0.4440	21.3384 ± 0.1543	3.2231 ± 0.0068	13.3453 ± 0.4112	19.0815 ± 0.5233
Seven vs Rest	3.8583 ± 0.1120	3.8727 ± 0.0415	3.2045 ± 0.0036	13.5062 ± 0.5035	19.8625 ± 1.1333	3.2112 ± 0.0049	13.4773 ± 0.1477	19.2886 ± 0.1979
Seven (1%) vs Rest	3.7307 ± 0.2806	3.5132 ± 0.0400	3.3950 ± 0.0637	11.6475 ± 0.4370	17.4530 ± 1.3970	3.3176 ± 0.0223	12.2138 ± 0.6597	16.7927 ± 0.6346
Seven (3%) vs Rest	3.5869 ± 0.0551	3.5954 ± 0.1851	3.2924 ± 0.0401	10.6589 ± 0.4248	14.4788 ± 1.0426	3.2983 ± 0.0224	11.3420 ± 0.4699	15.8030 ± 0.6752
Seven (10%) vs Rest	3.7807 ± 0.0572	3.7895 ± 0.1297	3.2997 ± 0.0242	12.1222 ± 0.3133	15.6297 ± 0.3794	3.2977 ± 0.0209	12.0173 ± 0.7350	16.0526 ± 0.2765
Seven (30%) vs Rest	3.7886 ± 0.0373	3.8826 ± 0.0562	3.2299 ± 0.0149	15.1048 ± 0.2003	20.4772 ± 0.3672	3.2227 ± 0.0130	12.6328 ± 0.3042	16.8903 ± 0.2194

Table B.10: Raw Model Performance on TEV Error-Prone PCR Datasets. Performance measured by weighted mean-squared-error.

Model	Linear		ESM-2 (8M, CNN)					
Fitness Variance Cutoff	0	0.001	0.000000			0.001		
Resampling	False	False	False		True	False		True
Power	100000	100000	100000	100000	1000000	100000	100000	1000000
Task								
One vs Rest	6.2069 ± 0.1215	6.1199 ± 0.0272	8.3189 ± 0.2689	7.8731 ± 0.3736	7.6561 ± 1.3800	7.5916 ± 1.2397	5.2553 ± 0.7874	5.3323 ± 1.8724
Two vs Rest	7.4595 ± 0.6346	6.5852 ± 0.3725	8.6105 ± 0.0058	1.5638 ± 0.3833	4.1714 ± 0.3669	8.2459 ± 0.1462	1.6788 ± 0.0515	2.7347 ± 0.4415
Three vs Rest	8.8709 ± 0.8800	8.0061 ± 0.0975	10.0722 ± 0.0777	1.1233 ± 0.1074	3.0094 ± 0.2600	10.6641 ± 0.1593	0.5373 ± 0.0383	0.4666 ± 0.0506
Four vs Rest	9.4013 ± 0.4378	8.8604 ± 0.1905	10.8435 ± 0.0867	0.3971 ± 0.0036	0.9574 ± 0.0830	11.4739 ± 0.1812	0.7564 ± 0.0136	0.4266 ± 0.0038
Five vs Rest	10.0468 ± 1.2173	9.5901 ± 0.6934	10.8263 ± 0.1491	0.5633 ± 0.0400	0.5253 ± 0.0521	11.1952 ± 0.0822	0.8384 ± 0.0697	0.4058 ± 0.0285
Six vs Rest	10.9178 ± 0.3821	9.8588 ± 0.3907	10.6598 ± 0.2273	0.8952 ± 0.0809	0.4409 ± 0.0065	10.7523 ± 0.0559	1.0001 ± 0.0915	0.4172 ± 0.0172
Seven vs Rest	10.3870 ± 0.1424	9.5895 ± 0.6161	10.4871 ± 0.0259	1.0402 ± 0.1051	0.5053 ± 0.0871	10.6804 ± 0.0115	0.9723 ± 0.0163	0.3972 ± 0.0085
Seven (1%) vs Rest	9.0428 ± 1.1329	9.5094 ± 0.1878	8.7966 ± 0.0896	1.4091 ± 0.1619	0.5413 ± 0.0989	9.8424 ± 0.4979	1.3519 ± 0.1089	0.6082 ± 0.0333
Seven (3%) vs Rest	10.2637 ± 0.2752	9.6846 ± 0.5011	9.4194 ± 0.3741	1.6298 ± 0.1516	0.8215 ± 0.1407	10.4108 ± 0.3473	1.4738 ± 0.1152	0.6571 ± 0.0775
Seven (10%) vs Rest	9.8276 ± 0.4708	9.8588 ± 0.5642	10.3751 ± 0.3766	1.2717 ± 0.0643	0.6660 ± 0.0467	10.7709 ± 0.1335	1.2651 ± 0.1780	0.5958 ± 0.0357
Seven (30%) vs Rest	10.0800 ± 0.3664	10.5641 ± 0.1815	10.4995 ± 0.2322	0.7352 ± 0.0224	0.4162 ± 0.0221	10.6395 ± 0.1070	1.1256 ± 0.0589	0.5181 ± 0.0094

Table B.11: Raw Model Performance on TEV Error-Prone PCR Datasets. Performance measured by active weighted mean-squared-error (MSE on the top 5% of actives).

Model	Linear		ESM-2 (8M, CNN)					
Fitness Variance Cutoff	0	0.001	0.000000			0.001		
Resampling	False	False	False		True	False		True
Power	100000	100000	100000	100000	1000000	100000	100000	1000000
Task								
One vs Rest	0.4825 ± 0.0086	0.4571 ± 0.0029	0.4685 ± 0.0157	0.4547 ± 0.0226	0.4509 ± 0.0173	0.4535 ± 0.0101	0.4519 ± 0.0120	0.4506 ± 0.0010
Two vs Rest	0.4621 ± 0.0155	0.4704 ± 0.0148	0.5188 ± 0.0023	0.4452 ± 0.0021	0.4534 ± 0.0075	0.5200 ± 0.0128	0.4337 ± 0.0118	0.4450 ± 0.0127
Three vs Rest	0.4619 ± 0.0132	0.4673 ± 0.0264	0.5340 ± 0.0052	0.4477 ± 0.0102	0.4417 ± 0.0045	0.5227 ± 0.0007	0.4395 ± 0.0041	0.4357 ± 0.0052
Four vs Rest	0.4501 ± 0.0087	0.4625 ± 0.0168	0.5259 ± 0.0047	0.4465 ± 0.0102	0.4420 ± 0.0151	0.5227 ± 0.0014	0.4527 ± 0.0222	0.4409 ± 0.0126
Five vs Rest	0.4470 ± 0.0137	0.4680 ± 0.0214	0.5267 ± 0.0091	0.4345 ± 0.0083	0.4437 ± 0.0113	0.5338 ± 0.0065	0.4505 ± 0.0042	0.4448 ± 0.0051
Six vs Rest	0.4559 ± 0.0168	0.4641 ± 0.0124	0.5206 ± 0.0057	0.4618 ± 0.0120	0.4436 ± 0.0094	0.5224 ± 0.0020	0.4591 ± 0.0172	0.4443 ± 0.0075
Seven vs Rest	0.4585 ± 0.0038	0.4709 ± 0.0063	0.5291 ± 0.0079	0.4578 ± 0.0153	0.4347 ± 0.0088	0.5362 ± 0.0060	0.4589 ± 0.0094	0.4473 ± 0.0080
Seven (1%) vs Rest	0.4711 ± 0.0182	0.4648 ± 0.0141	0.4485 ± 0.0156	0.4363 ± 0.0174	0.4326 ± 0.0153	0.4728 ± 0.0100	0.4181 ± 0.0032	0.4188 ± 0.0053
Seven (3%) vs Rest	0.4499 ± 0.0169	0.4629 ± 0.0176	0.4742 ± 0.0136	0.4321 ± 0.0276	0.4268 ± 0.0110	0.4828 ± 0.0017	0.4274 ± 0.0026	0.4383 ± 0.0152
Seven (10%) vs Rest	0.4562 ± 0.0143	0.4684 ± 0.0070	0.5084 ± 0.0022	0.4569 ± 0.0020	0.4427 ± 0.0048	0.5130 ± 0.0093	0.4438 ± 0.0069	0.4324 ± 0.0134
Seven (30%) vs Rest	0.4567 ± 0.0133	0.4673 ± 0.0136	0.5238 ± 0.0110	0.4308 ± 0.0048	0.4478 ± 0.0064	0.5291 ± 0.0050	0.4427 ± 0.0079	0.4442 ± 0.0093

Table B.12: Raw Model Performance on TEV Error-Prone PCR Datasets. Performance measured by mean fitness of top 100 variants.

Model	Task	Fitness Variance Cutoff	Hit Rates	Max Top 100	Mean Top 100	Weighted MSE
ESM-2 (8M, CNN)	Seven (1%) vs Rest	0	0.2500 ± 0.0361	0.8423 ± 0.0226	0.4485 ± 0.0156	3.3950 ± 0.0637
		0.0001	0.2967 ± 0.0153	0.8479 ± 0.0369	0.4740 ± 0.0169	3.3094 ± 0.0357
		0.0005	0.2733 ± 0.0231	0.8512 ± 0.0114	0.4626 ± 0.0095	3.3240 ± 0.0277
		0.001	0.3533 ± 0.0473	0.8285 ± 0.0201	0.4728 ± 0.0100	3.3176 ± 0.0223
		0.005	0.3500 ± 0.0300	0.8490 ± 0.0484	0.4923 ± 0.0158	3.3322 ± 0.0469
	Seven (3%) vs Rest	0	0.3233 ± 0.0153	0.9237 ± 0.0551	0.4742 ± 0.0136	3.2924 ± 0.0401
		0.0001	0.3467 ± 0.0551	0.8499 ± 0.0250	0.4836 ± 0.0094	3.3360 ± 0.0211
		0.0005	0.2933 ± 0.0115	0.8570 ± 0.0567	0.4839 ± 0.0091	3.2944 ± 0.0307
		0.001	0.3300 ± 0.0200	0.8675 ± 0.0057	0.4828 ± 0.0017	3.2983 ± 0.0224
		0.005	0.3533 ± 0.0493	0.8684 ± 0.0294	0.4866 ± 0.0223	3.2806 ± 0.0127
	Seven (10%) vs Rest	0	0.4100 ± 0.0400	0.8552 ± 0.0248	0.5084 ± 0.0022	3.2997 ± 0.0242
		0.0001	0.3967 ± 0.0404	0.8713 ± 0.0437	0.5090 ± 0.0133	3.2886 ± 0.0273
		0.0005	0.4000 ± 0.0300	0.8837 ± 0.0000	0.5119 ± 0.0064	3.3107 ± 0.0291
		0.001	0.4200 ± 0.0557	0.8765 ± 0.0350	0.5130 ± 0.0093	3.2977 ± 0.0209
		0.005	0.3767 ± 0.0058	0.8268 ± 0.0258	0.5138 ± 0.0046	3.2990 ± 0.0171

Table B.13: Raw Model Performance on TEV Error-Prone PCR Datasets. Performance measured by hit rate (as compared to wild-type fitness), maximum fitness of top 100 variants, mean fitness of top 100 variants, and weighted mean squared error.

Substrate	Fitness Variance Cutoff	Weighted MSE	Hit Rates	Mean Top 100	Max Top 100	Wild-Type Fitness
ENLYFQS	0	3.2045 ± 0.0036	0.4533 ± 0.0058	0.5291 ± 0.0079	0.8837 ± 0.0000	0.5386
	0.001	3.2112 ± 0.0049	0.4700 ± 0.0200	0.5362 ± 0.0060	0.8916 ± 0.0137	0.5386
ENLYGQS	0	2.0444 ± 0.0016	0.3733 ± 0.0289	0.2392 ± 0.0001	0.5753 ± 0.0513	0.2553
	0.001	2.0416 ± 0.0014	0.4000 ± 0.0000	0.2315 ± 0.0044	0.6880 ± 0.1232	0.2553
HNLYFQS	0	2.8180 ± 0.0074	0.4700 ± 0.0346	0.5244 ± 0.0103	0.8880 ± 0.0194	0.5302
	0.001	2.8107 ± 0.0142	0.4867 ± 0.0416	0.5267 ± 0.0133	0.8985 ± 0.0274	0.5302
HNLYGHS	0	2.5335 ± 0.0010	0.4567 ± 0.0379	0.0891 ± 0.0103	0.4766 ± 0.0223	0.0765
	0.001	2.5333 ± 0.0005	0.4300 ± 0.0346	0.0896 ± 0.0034	0.6166 ± 0.1477	0.0765
HNLYGQS	0	1.8848 ± 0.0003	0.4500 ± 0.0656	0.0884 ± 0.0226	0.6680 ± 0.2578	0.0907
	0.001	1.8848 ± 0.0006	0.4567 ± 0.0208	0.0930 ± 0.0057	0.5919 ± 0.1917	0.0907

Table B.14: Raw Model Performance on TEV Alternative Substrate Landscapes. Performance measured by weighted mean squared error, hit rate (as compared to wild-type fitness), mean fitness of top 100 variants, and maximum fitness of top 100 variants.

Appendix C

Data and Code Availability

Model weights for FLIGHTED-Selection and FLIGHTED-DHARMA, the FLIGHTED GB1 landscape, and all GB1 models have been released at this Zenodo repository. The FLIGHTED TEV landscape and all TEV models have been released at this Zenodo repository. All code necessary to run FLIGHTED and reproduce all figures in Chapter 3 and the first section of Chapter 4 has been released in at this Github repository. All code necessary to reproduce all other results in this paper is available in this private Github repository; further code and data releases are pending final submission of the DHARMA paper for publication. Please contact the author if you would like access to any of the code and data in this paper that has not been publicly released.

References

[1] T. M. Antalis, T. Shea-Donohue, S. N. Vogel, C. Sears, and A. Fasano. “Mechanisms of Disease: Protease Functions in Intestinal Mucosal Pathobiology”. In: *Nature Clinical Practice. Gastroenterology & Hepatology* 4.7 (July 2007), pp. 393–402. ISSN: 1743-4386. DOI: 10.1038/ncpgasthep0846.

[2] A. J. Glazko. “Effect of Blood Protease and Trypsin Inhibitor on the Clotting Mechanism”. In: *The Journal of Clinical Investigation* 26.3 (May 1947), pp. 364–369. ISSN: 0021-9738. DOI: 10.1172/JCI101818.

[3] S. J. Martin and D. R. Green. “Protease Activation during Apoptosis: Death by a Thousand Cuts?” In: *Cell* 82.3 (Aug. 1995), pp. 349–352. ISSN: 0092-8674. DOI: 10.1016/0092-8674(95)90422-0.

[4] R. P. Dyer and G. A. Weiss. “Making the Cut with Protease Engineering”. In: *Cell Chemical Biology* 29.2 (Feb. 2022), pp. 177–190. ISSN: 2451-9448. DOI: 10.1016/j.chembiol.2021.12.001.

[5] C. Reichhardt, H. M. Jacobs, M. Matwichuk, C. Wong, D. J. Wozniak, and M. R. Parsek. “The Versatile Pseudomonas Aeruginosa Biofilm Matrix Protein CdrA Promotes Aggregation through Different Extracellular Exopolysaccharide Interactions”. In: *Journal of Bacteriology* 202.19 (Sept. 2020), e00216–20. ISSN: 1098-5530. DOI: 10.1128/JB.00216-20.

[6] A. Bluhm, S. Schrempel, S. von Hörsten, A. Schulze, and S. Roßner. “Proteolytic α-Synuclein Cleavage in Health and Disease”. In: *International Journal of Molecular Sciences* 22.11 (May 2021), p. 5450. ISSN: 1422-0067. DOI: 10.3390/ijms22115450.

[7] S. V. Sharma, D. W. Bell, J. Settleman, and D. A. Haber. “Epidermal Growth Factor Receptor Mutations in Lung Cancer”. In: *Nature Reviews Cancer* 7.3 (Mar. 2007), pp. 169–181. ISSN: 1474-1768. DOI: 10.1038/nrc2088. (Visited on 03/31/2025).

[8] L. Labanieh et al. “Enhanced Safety and Efficacy of Protease-Regulated CAR-T Cell Receptors”. In: *Cell* 185.10 (May 2022), 1745–1763.e22. ISSN: 1097-4172. DOI: 10.1016/j.cell.2022.03.041.

[9] Z. Chen et al. “De Novo Design of Protein Logic Gates”. In: *Science* 368.6486 (Apr. 2020), pp. 78–84. DOI: 10.1126/science.aay2790. (Visited on 03/31/2025).

[10] H. K. Chung and M. Z. Lin. “On the Cutting Edge: Protease-Based Methods for Sensing and Controlling Cell Biology”. In: *Nature Methods* 17.9 (Sept. 2020), pp. 885–896. ISSN: 1548-7105. DOI: 10.1038/s41592-020-0891-z. (Visited on 03/31/2025).

[11] X. J. Gao, L. S. Chong, M. S. Kim, and M. B. Elowitz. “Programmable Protein Circuits in Living Cells”. In: *Science* 361.6408 (Sept. 2018), pp. 1252–1258. DOI: 10.1126/science.aat5062. (Visited on 03/31/2025).

[12] Z. Guo, O. Smutok, C. E. Ayva, P. Walden, J. Parker, J. Whitfield, C. E. Vickers, J. P. J. Ungerer, E. Katz, and K. Alexandrov. “Development of Epistatic YES and AND Protein Logic Gates and Their Assembly into Signalling Cascades”. In: *Nature Nanotechnology* 18.11 (Nov. 2023), pp. 1327–1334. ISSN: 1748-3395. DOI: 10.1038/s41565-023-01450-y. (Visited on 03/31/2025).

[13] J. Phan, A. Zdanov, A. G. Evdokimov, J. E. Tropea, H. K. Peters, R. B. Kapust, M. Li, A. Wlodawer, and D. S. Waugh. “Structural Basis for the Substrate Specificity of Tobacco Etch Virus Protease”. In: *Journal of Biological Chemistry* 277.52 (Dec. 2002), pp. 50564–50572. ISSN: 00219258. DOI: 10.1074/jbc.M207224200. (Visited on 05/04/2022).

[14] L. D. Cabrita, D. Gilis, A. L. Robertson, Y. Dehouck, M. Rooman, and S. P. Bottomley. “Enhancing the Stability and Solubility of TEV Protease Using in Silico Design”. In: *Protein Science : A Publication of the Protein Society* 16.11 (Nov. 2007), pp. 2360–2367. ISSN: 0961-8368. DOI: 10.1110/ps.072822507. (Visited on 02/05/2024).

[15] L. Yi, M. C. Gebhard, Q. Li, J. M. Taft, G. Georgiou, and B. L. Iverson. “Engineering of TEV Protease Variants by Yeast ER Sequestration Screening (YESS) of Combinatorial Libraries”. In: *Proceedings of the National Academy of Sciences* 110.18 (Apr. 2013), pp. 7229–7234. DOI: 10.1073/pnas.1215994110. (Visited on 02/05/2024).

[16] M. S. Packer, H. A. Rees, and D. R. Liu. “Phage-Assisted Continuous Evolution of Proteases with Altered Substrate Specificity”. In: *Nature Communications 2017 8:1* 8.1 (Oct. 2017), pp. 1–11. ISSN: 2041-1723. DOI: 10.1038/s41467-017-01055-9. (Visited on 03/22/2022).

[17] T. Kortemme. “*De Novo* Protein Design—From New Structures to Programmable Functions”. In: *Cell* 187.3 (Feb. 2024), pp. 526–544. ISSN: 0092-8674. DOI: 10.1016/j.cell.2023.12.028. (Visited on 02/03/2025).

[18] J. A. Ruffolo and A. Madani. “Designing Proteins with Language Models”. In: *Nature Biotechnology* 42.2 (Feb. 2024), pp. 200–202. ISSN: 1546-1696. DOI: 10.1038/s41587-024-02123-4. (Visited on 02/15/2024).

[19] V. Frappier and A. E. Keating. “Data-Driven Computational Protein Design”. In: *Current Opinion in Structural Biology* 69 (Aug. 2021), pp. 63–69. DOI: 10.1016/J.SBI.2021.03.009. (Visited on 09/07/2021).

[20] X. Pan and T. Kortemme. “Recent Advances in *de Novo* Protein Design: Principles, Methods, and Applications”. In: *Journal of Biological Chemistry* 296 (Jan. 2021), p. 100558. ISSN: 0021-9258. DOI: 10.1016/j.jbc.2021.100558. (Visited on 02/03/2025).

[21] M. H. Hecht, S. Zarzhitsky, C. Karas, and S. Chari. “Are Natural Proteins Special? Can We Do That?” In: *Current Opinion in Structural Biology* 48 (Feb. 2018), pp. 124–132. ISSN: 0959440X. DOI: 10.1016/j.sbi.2017.11.009. (Visited on 03/12/2025).

[22] I. V. Korendovych and W. F. DeGrado. “*De Novo* Protein Design, a Retrospective”. In: *Quarterly Reviews of Biophysics* 53 (2020), e3. ISSN: 0033-5835, 1469-8994. DOI: 10.1017/S0033583519000131. (Visited on 02/03/2025).

[23] L. Regan and W. F. DeGrado. “Characterization of a Helical Protein Designed from First Principles”. In: *Science* 241.4868 (Aug. 1988), pp. 976–978. DOI: 10.1126/science.3043666. (Visited on 03/12/2025).

[24] S. P. Ho and W. F. DeGrado. “Design of a 4-Helix Bundle Protein: Synthesis of Peptides Which Self-Associate into a Helical Protein”. In: *Journal of the American Chemical Society* 109.22 (Oct. 1987), pp. 6751–6758. ISSN: 0002-7863. DOI: 10.1021/ja00256a032. (Visited on 03/12/2025).

[25] M. H. Hecht, J. S. Richardson, D. C. Richardson, and R. C. Ogden. “De Novo Design, Expression, and Characterization of Felix: A Four-Helix Bundle Protein of Native-Like Sequence”. In: *Science* 249.4971 (Aug. 1990), pp. 884–891. DOI: 10.1126/science.2392678. (Visited on 03/12/2025).

[26] B. I. Dahiyat and S. L. Mayo. “De Novo Protein Design: Fully Automated Sequence Selection”. In: *Science* 278.5335 (Oct. 1997), pp. 82–87. DOI: 10.1126/science.278.5335.82. (Visited on 03/12/2025).

[27] B. Kuhlman, G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, and D. Baker. “Design of a Novel Globular Protein Fold with Atomic-Level Accuracy”. In: *Science (New York, N.Y.)* 302.5649 (Nov. 2003), pp. 1364–1368. ISSN: 1095-9203. DOI: 10.1126/science.1089427.

[28] P.-S. Huang, Y.-E. A. Ban, F. Richter, I. Andre, R. Vernon, W. R. Schief, and D. Baker. “RosettaRemodel: A Generalized Framework for Flexible Backbone Protein Design”. In: *PLOS ONE* 6.8 (Aug. 2011), e24109. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0024109. (Visited on 03/12/2025).

[29] C. O. Mackenzie and G. Grigoryan. “Protein Structural Motifs in Prediction and Design”. In: *Current Opinion in Structural Biology* 44 (June 2017), pp. 161–167. ISSN: 1879-033X. DOI: 10.1016/j.sbi.2017.03.012.

[30] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. “The Protein Data Bank”. In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 235–242. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.235.

[31] The UniProt Consortium. “UniProt: The Universal Protein Knowledgebase in 2025”. In: *Nucleic Acids Research* 53.D1 (Jan. 2025), pp. D609–D617. ISSN: 1362-4962. DOI: 10.1093/nar/gkae1010. (Visited on 03/12/2025).

[32] J. B. Ingraham et al. “Illuminating Protein Space with a Programmable Generative Model”. In: *Nature* (Nov. 2023), pp. 1–9. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06728-8. (Visited on 11/15/2023).

[33] D. S. Marks, L. J. Colwell, R. Sheridan, T. A. Hopf, A. Pagnani, R. Zecchina, and C. Sander. “Protein 3D Structure Computed from Evolutionary Sequence Variation”. In: *PLOS ONE* 6.12 (Dec. 2011), e28766. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0028766. (Visited on 04/24/2022).

[34] T. A. Hopf, L. J. Colwell, R. Sheridan, B. Rost, C. Sander, and D. S. Marks. “Three-Dimensional Structures of Membrane Proteins from Genomic Sequencing”. In: *Cell* 149.7 (2012), pp. 1607–1621. ISSN: 00928674. DOI: 10.1016/j.cell.2012.04.012.

[35] A. W. Senior et al. “Improved Protein Structure Prediction Using Potentials from Deep Learning”. In: *Nature* 577.7792 (Jan. 2020), pp. 706–710. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-1923-7. (Visited on 04/24/2022).

[36] J. Jumper et al. “Applying and Improving AlphaFold at CASP14”. In: *Proteins: Structure, Function, and Bioinformatics* 89.12 (2021), pp. 1711–1721. ISSN: 1097-0134. DOI: 10.1002/prot.26257. (Visited on 05/07/2022).

[37] J. Jumper et al. “Highly Accurate Protein Structure Prediction with AlphaFold”. In: *Nature* (July 2021), pp. 1–11. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. (Visited on 07/17/2021).

[38] M. Baek et al. “Accurate Prediction of Protein Structures and Interactions Using a Three-Track Neural Network”. In: *Science* 373.6557 (Aug. 2021), pp. 871–876. DOI: 10.1126/science.abj8754. (Visited on 04/21/2022).

[39] J. H. Lee, P. Yadollahpour, A. Watkins, N. C. Frey, A. Leaver-Fay, S. Ra, K. Cho, V. Gligorijevic, A. Regev, and R. Bonneau. *EquiFold: Protein Structure Prediction with a Novel Coarse-Grained Structure Representation.* 2022. DOI: 10.1101/2022.10.07.511322.

[40] M. Baek, I. Anishchenko, I. R. Humphreys, Q. Cong, D. Baker, and F. DiMaio. *Efficient and Accurate Prediction of Protein Structure Using RoseTTAFold2.* May 2023. DOI: 10.1101/2023.05.24.542179. (Visited on 07/04/2023).

[41] R. Evans et al. *Protein Complex Prediction with AlphaFold-Multimer.* Mar. 2022. DOI: 10.1101/2021.10.04.463034. (Visited on 03/13/2025).

[42] M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov, and M. Steinegger. “ColabFold: Making Protein Folding Accessible to All”. In: *Nature Methods* (May 2022), pp. 1–4. ISSN: 1548-7105. DOI: 10.1038/s41592-022-01488-1. (Visited on 05/30/2022).

[43] J. Abramson et al. “Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3”. In: *Nature* (May 2024), pp. 1–3. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07487-w. (Visited on 05/08/2024).

[44] R. Krishna et al. “Generalized Biomolecular Modeling and Design with RoseTTAFold All-Atom”. In: *Science* 0.0 (Mar. 2024), eadl2528. DOI: 10.1126/science.adl2528. (Visited on 03/07/2024).

[45] J. Ingraham, V. K. Garg, R. Barzilay, and T. Jaakkola. “Generative Models for Graph-Based Protein Design”. In: *NeurIPS* (2019). (Visited on 09/07/2021).

[46] J. Zhou, A. E. Panaitiu, and G. Grigoryan. “A General-Purpose Protein Design Framework Based on Mining Sequence-Structure Relationships in Known Protein Structures”. In: *Proceedings of the National Academy of Sciences* 117.2 (2020), pp. 1059–1068. DOI: 10.1073/pnas.1908723117/-/DCSupplemental. (Visited on 09/07/2021).

[47] A. J. Li, V. Sundar, G. Grigoryan, and A. E. Keating. “TERMinator: A Neural Framework for Structure-Based Protein Design Using Tertiary Repeating Motifs”. In: *Machine Learning in Structural Biology Workshop, Neural Information Processing Systems.* 2021. (Visited on 03/22/2022).

[48] A. J. Li, M. Lu, I. Desta, V. Sundar, G. Grigoryan, and A. E. Keating. "Neural Network-derived Potts Models for Structure-based Protein Design Using Backbone Atomic Coordinates and Tertiary Motifs". In: *Protein Science : A Publication of the Protein Society* 32.2 (Feb. 2023), e4554. ISSN: 0961-8368. DOI: 10.1002/pro.4554. (Visited on 01/10/2024).

[49] J. Dauparas et al. "Robust Deep Learning Based Protein Sequence Design Using ProteinMPNN". In: *Science* 378.6615 (Sept. 2022), pp. 49–56. DOI: 10.1126/science.add2187. (Visited on 06/05/2022).

[50] C. Norn et al. "Protein Sequence Design by Conformational Landscape Optimization". In: *Proceedings of the National Academy of Sciences* 118.11 (Mar. 2021), e2017228118. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2017228118. (Visited on 06/05/2022).

[51] C. Hsu, R. Verkuil, J. Liu, Z. Lin, B. Hie, T. Sercu, A. Lerer, and A. Rives. "Learning Inverse Folding from Millions of Predicted Structures". In: *International Conference on Machine Learning*. Apr. 2022. DOI: 10.1101/2022.04.10.487779. (Visited on 04/11/2022).

[52] S. P. Mahajan, J. A. Ruffolo, and J. J. Gray. *Contextual Protein Encodings from Equivariant Graph Transformers*. July 2023. DOI: 10.1101/2023.07.15.549154. (Visited on 07/27/2023).

[53] R. J. de Haas et al. "Rapid and Automated Design of Two-Component Protein Nanomaterials Using ProteinMPNN". In: *Proceedings of the National Academy of Sciences* 121.13 (Mar. 2024), e2314646121. DOI: 10.1073/pnas.2314646121. (Visited on 03/28/2024).

[54] D. Ding, A. Y. Shaw, S. Sinai, N. Rollins, N. Prywes, D. F. Savage, M. T. Laub, and D. S. Marks. "Protein Design Using Structure-Based Residue Preferences". In: *Nature Communications* 15.1 (Feb. 2024), p. 1639. ISSN: 2041-1723. DOI: 10.1038/s41467-024-45621-4. (Visited on 02/23/2024).

[55] M. H. Høie, A. Hummer, T. H. Olsen, B. Aguilar-Sanjuan, M. Nielsen, and C. M. Deane. "AntiFold: Improved Antibody Structure-Based Design Using Inverse Folding". In: *Bioinformatics Advances* (Mar. 2025), vbae202. DOI: 10.1093/bioadv/vbae202. (Visited on 05/07/2024).

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Neural Information Processing Systems*. Vol. 2017-December. Neural information processing systems foundation, June 2017, pp. 5999–6009. arXiv: 1706.03762. (Visited on 09/24/2021).

[57] T. B. Brown et al. "Language Models Are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. July 2020. arXiv: 2005.14165. (Visited on 04/23/2022).

[58] OpenAI et al. *GPT-4 Technical Report*. Mar. 2024. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs]. (Visited on 03/13/2025).

[59] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church. "Unified Rational Protein Engineering with Sequence-Based Deep Representation Learning". In: *Nature Methods* 16.12 (Dec. 2019), pp. 1315–1322. ISSN: 15487105. DOI: 10.1038/s41592-019-0598-1. (Visited on 03/22/2021).

[60] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song. "Evaluating Protein Transfer Learning with TAPE". In: *Advances in Neural Information Processing Systems*. 2019. DOI: 10.1101/676825. arXiv: 1906.08230.

[61] S. Biswas, G. Khimulya, E. C. Alley, K. M. Esvelt, and G. M. Church. "Low-N Protein Engineering with Data-Efficient Deep Learning". In: *Nature Methods* 18.4 (Apr. 2021), pp. 389–396. ISSN: 1548-7105. DOI: 10.1038/s41592-021-01100-y. (Visited on 03/21/2022).

[62] N. Ferruz and B. Höcker. "Controllable Protein Design with Language Models". In: *Nature Machine Intelligence* 4.6 (June 2022), pp. 521–532. ISSN: 2522-5839. DOI: 10.1038/s42256-022-00499-z. (Visited on 06/28/2022).

[63] B. L. Hie and K. K. Yang. "Adaptive Machine Learning for Protein Engineering". In: *Current Opinion in Structural Biology* 72 (Feb. 2022), pp. 145–152. ISSN: 0959-440X. DOI: 10.1016/j.sbi.2021.11.002. (Visited on 06/08/2022).

[64] K. E. Johnston, C. Fannjiang, B. J. Wittmann, B. L. Hie, K. K. Yang, and Z. Wu. *Machine Learning for Protein Engineering*. May 2023. DOI: 10.48550/arXiv.2305.16634. arXiv: 2305.16634 [q-bio]. (Visited on 07/04/2023).

[65] A. Rives, S. Goyal, J. Meier, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus. "Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences". In: *Proceedings of the National Academy of Sciences* 118.15 (Apr. 2021), p. 622803. DOI: 10.1101/622803. (Visited on 03/17/2021).

[66] J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. "Language Models Enable Zero-Shot Prediction of the Effects of Mutations on Protein Function". In: *Proceedings of the 35th Conference on Neural Information Processing Systems* (2021). (Visited on 12/08/2021).

[67] T. Hayes et al. *Simulating 500 Million Years of Evolution with a Language Model*. Dec. 2024. DOI: 10.1101/2024.07.01.600583. (Visited on 12/31/2024).

[68] A. Madani, B. McCann, N. Naik, N. S. Keskar, N. Anand, R. R. Eguchi, H. Po-Ssu, and R. Socher. *ProGen: Language Modeling for Protein Generation*. Mar. 2020. DOI: 10.1101/2020.03.07.982272. (Visited on 03/22/2021).

[69] R. Rao, J. Liu, R. Verkuil, J. Meier, J. F. Canny, P. Abbeel, T. Sercu, and A. Rives. "MSA Transformer". In: *Proceedings of the 38th International Conference on Machine Learning* (Aug. 2021). DOI: 10.1101/2021.02.12.430858. (Visited on 04/21/2022).

[70] N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, and M. Linial. "ProteinBERT: A Universal Deep-Learning Model of Protein Sequence and Function". In: *Bioinformatics* 38.8 (Apr. 2022), pp. 2102–2110. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btac020. (Visited on 05/16/2022).

[71] N. Ferruz, S. Schmidt, and B. Höcker. "ProtGPT2 Is a Deep Unsupervised Language Model for Protein Design". In: *Nature Communications* 13 (Mar. 2022), p. 4348. DOI: 10.1038/s41467-022-32007-7. (Visited on 05/02/2022).

[72] D. Hesslow, N. Zanichelli, P. Notin, I. Poli, and D. Marks. *RITA: A Study on Scaling Up Generative Protein Sequence Models*. May 2022. arXiv: 2205.05789 [cs, q-bio]. (Visited on 05/16/2022).

[73] E. Nijkamp, J. Ruffolo, E. N. Weinstein, N. Naik, and A. Madani. "ProGen2: Exploring the Boundaries of Protein Language Models". In: *Cell Systems* 14.11 (Oct. 2023), P968–978.E3. DOI: 10.1016/j.cels.2023.10.002. (Visited on 06/29/2022).

[74] P. Notin, M. Dias, J. Frazer, J. Marchena-Hurtado, A. Gomez, D. S. Marks, and Y. Gal. "Tranception: Protein Fitness Prediction with Autoregressive Transformers and Inference-Time Retrieval". In: *Proceedings of the 39th International Conference on Machine Learning*. arXiv, May 2022. (Visited on 05/30/2022).

[75] K. K. Yang, A. X. Lu, and N. Fusi. "Convolutions Are Competitive with Transformers for Protein Sequence Pretraining". In: *Cell Systems* 15.3 (Mar. 2024), p286–294.e2. DOI: 10.1016/j.cels.2024.01.008.

[76] A. Madani et al. "Large Language Models Generate Functional Protein Sequences across Diverse Families". In: *Nature Biotechnology* (Jan. 2023), pp. 1–8. ISSN: 1546-1696. DOI: 10.1038/s41587-022-01618-2. (Visited on 01/26/2023).

[77] N. C. Frey, T. Joren, A. Ismail, A. Goodman, R. Bonneau, K. Cho, and V. Gligorijevic. *Cramming Protein Language Model Training in 24 GPU Hours*. May 2024. DOI: 10.1101/2024.05.14.594108. (Visited on 05/17/2024).

[78] B. Chen et al. "xTrimoPGLM: Unified 100-Billion-Parameter Pretrained Transformer for Deciphering the Language of Proteins". In: *Nature Methods* (Apr. 2025), pp. 1–12. ISSN: 1548-7105. DOI: 10.1038/s41592-025-02636-z. (Visited on 04/03/2025).

[79] M. H. Celik and X. Xie. *Efficient Inference, Training, and Fine-tuning of Protein Language Models*. Oct. 2024. DOI: 10.1101/2024.10.22.619563. (Visited on 10/25/2024).

[80] F. Z. Peng, C. Wang, T. Chen, B. Schussheim, S. Vincoff, and P. Chatterjee. "PTM-Mamba: A PTM-aware Protein Language Model with Bidirectional Gated Mamba Blocks". In: *Nature Methods* (Apr. 2025), pp. 1–5. ISSN: 1548-7105. DOI: 10.1038/s41592-025-02656-9. (Visited on 04/10/2025).

[81] D. Sgarbossa, C. Malbranke, and A.-F. Bitbol. *ProtMamba: A Homology-Aware but Alignment-Free Protein State Space Model*. May 2024. DOI: 10.1101/2024.05.24.595730. (Visited on 05/25/2024).

[82] A. Bhatnagar, S. Jain, J. Beazer, S. C. Curran, A. M. Hoffnagle, K. Ching, M. Martyn, S. Nayfach, J. A. Ruffolo, and A. Madani. *Scaling Unlocks Broader Generation and Deeper Functional Understanding of Proteins*. 2025.

[83] S. Gelman, B. Johnson, C. Freschlin, S. D'Costa, A. Gitter, and P. A. Romero. *Biophysics-Based Protein Language Models for Protein Engineering*. Mar. 2024. DOI: 10.1101/2024.03.15.585128. (Visited on 03/17/2024).

[84] M. Li, Y. Tan, X. Ma, B. Zhong, Z. Zhou, H. Yu, W. Ouyang, L. Hong, B. Zhou, and P. Tan. "ProSST: Protein Language Modeling with Quantized Structure and Disentangled Attention". In: *38th Conference on Neural Information Processing Systems*. May 2024. Chap. New Results, p. 2024.04.15.589672. DOI: 10.1101/2024.04.15.589672. (Visited on 05/17/2024).

[85] M. Pourmirzaei, F. Esmaili, M. Pourmirzaei, D. Wang, and D. Xu. *Prot2Token: A Multi-Task Framework for Protein Language Processing Using Autoregressive Language Modeling*. June 2024. DOI: 10.1101/2024.05.31.596915. (Visited on 06/03/2024).

[86] J. Su, C. Han, Y. Zhou, J. Shan, X. Zhou, and F. Yuan. "SaProt: Protein Language Modeling with Structure-aware Vocabulary". In: *International Conference on Learning Representations*. bioRxiv, Mar. 2024. Chap. New Results, p. 2023.10.01.560349. DOI: 10.1101/2023.10.01.560349. (Visited on 03/11/2024).

[87] J. Su, Z. Li, C. Han, Y. Zhou, J. Shan, X. Zhou, D. Ma, T. Opmc, S. Ovchinnikov, and F. Yuan. *SaprotHub: Making Protein Modeling Accessible to All Biologists*. May 2024. DOI: 10.1101/2024.05.24.595648. (Visited on 05/28/2024).

[88] J. Su, X. Zhou, X. Zhang, and F. Yuan. *ProTrek: Navigating the Protein Universe through Tri-Modal Contrastive Learning*. June 2024. DOI: 10.1101/2024.05.30.596740. (Visited on 06/03/2024).

[89] D. Wang et al. "S-PLM: Structure-aware Protein Language Model via Contrastive Learning between Sequence and Structure". In: *Advanced Science* (Dec. 2024), p. 2023.08.06.552203. DOI: 10.1002/advs.202404212. (Visited on 05/14/2024).

[90] L. Wang, R. Pulugurta, P. Vure, Y. Zhang, A. Pal, and P. Chatterjee. *PepDoRA: A Unified Peptide Language Model via Weight-Decomposed Low-Rank Adaptation*. Oct. 2024. DOI: 10.48550/arXiv.2410.20667. arXiv: 2410.20667. (Visited on 10/29/2024).

[91] K. E. Wu, H. Chang, and J. Zou. *ProteinCLIP: Enhancing Protein Language Models with Natural Language*. May 2024. DOI: 10.1101/2024.05.14.594226. (Visited on 05/18/2024).

[92] Z. Zhang, J. Lu, V. Chenthamarakshan, A. Lozano, P. Das, and J. Tang. *Structure-Informed Protein Language Model*. Feb. 2024. DOI: 10.48550/arXiv.2402.05856. arXiv: 2402.05856 `[cs, q-bio]`. (Visited on 02/19/2024).

[93] L. Zhuo, Z. Chi, M. Xu, H. Huang, H. Zheng, C. He, X.-L. Mao, and W. Zhang. "ProtLLM: An Interleaved Protein-Language LLM with Protein-as-Word Pre-Training". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Feb. 2024. arXiv: 2403.07920 `[cs, q-bio]`. (Visited on 03/14/2024).

[94] Z. Lin et al. "Evolutionary-Scale Prediction of Atomic-Level Protein Structure with a Language Model". In: *Science* 379.6637 (Mar. 2023), pp. 1123–1130. DOI: 10.1126/science.ade2574. (Visited on 03/26/2025).

[95] D. H. Bryant, A. Bashir, S. Sinai, N. K. Jain, P. J. Ogden, P. F. Riley, G. M. Church, L. J. Colwell, and E. D. Kelsic. "Deep Diversification of an AAV Capsid Protein by Machine Learning". In: *Nature Biotechnology* 39.6 (June 2021), pp. 691–696. ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-020-00793-4. (Visited on 05/27/2022).

[96] N. Thomas et al. "Engineering Highly Active Nuclease Enzymes with Machine Learning and High-Throughput Screening". In: *Cell Systems* 16.3 (Mar. 2025). ISSN: 2405-4712, 2405-4720. DOI: 10.1016/j.cels.2025.101236. (Visited on 03/26/2025).

[97] J. T. Rapp, B. J. Bremer, and P. A. Romero. “Self-Driving Laboratories to Autonomously Navigate the Protein Fitness Landscape”. In: *Nature Chemical Engineering* 1.1 (Jan. 2024), pp. 97–107. ISSN: 2948-1198. DOI: 10.1038/s44286-023-00002-4. (Visited on 01/12/2024).

[98] K. Jiang et al. “Rapid in Silico Directed Evolution by a Protein Language Model with EVOLVEpro”. In: *Science* 0.0 (Nov. 2024), eadr6006. DOI: 10.1126/science.adr6006. (Visited on 11/21/2024).

[99] N. C. Frey et al. *Lab-in-the-Loop Therapeutic Antibody Design with Deep Learning*. Feb. 2025. DOI: 10.1101/2025.02.19.639050. (Visited on 02/25/2025).

[100] B. J. Wittmann, Y. Yue, and F. H. Arnold. “Informed Training Set Design Enables Efficient Machine Learning-Assisted Directed Protein Evolution”. In: *Cell Systems* 12.11 (Nov. 2021), 1026–1045.e7. ISSN: 2405-4712. DOI: 10.1016/j.cels.2021.07.008. (Visited on 04/04/2025).

[101] F.-Z. Li, J. Yang, K. E. Johnston, E. Gürsoy, Y. Yue, and F. H. Arnold. *Evaluation of Machine Learning-Assisted Directed Evolution Across Diverse Combinatorial Landscapes*. Oct. 2024. DOI: 10.1101/2024.10.24.619774. (Visited on 10/28/2024).

[102] J. H. Cho, J. J. Collins, and W. W. Wong. “Universal Chimeric Antigen Receptors for Multiplexed and Logical Control of T Cell Responses”. In: *Cell* 173.6 (May 2018), 1426–1438.e11. ISSN: 00928674. DOI: 10.1016/j.cell.2018.03.038. (Visited on 04/24/2022).

[103] N. Gurdo, D. C. Volke, D. McCloskey, and P. I. Nikel. “Automating the Design-Build-Test-Learn Cycle towards next-Generation Bacterial Cell Factories”. In: *New Biotechnology* 74 (May 2023), pp. 1–15. ISSN: 1871-6784. DOI: 10.1016/j.nbt.2023.01.002. (Visited on 03/18/2025).

[104] P. A. Romero, A. Krause, and F. H. Arnold. “Navigating the Protein Fitness Landscape with Gaussian Processes”. In: *Proceedings of the National Academy of Sciences* 110.3 (Jan. 2013), E193–E201. ISSN: 00278424. DOI: 10.1073/pnas.1215251110. (Visited on 03/17/2021).

[105] C. Angermueller, D. Belanger, A. Gane, Z. Mariet, D. Dohan, K. Murphy, L. Colwell, and D. Sculley. “Population-Based Black-Box Optimization for Biological Sequence Design”. In: *Proceedings of the 37 Th International Conference on Machine Learning*. 2020. DOI: 10.1111/j.1365-246X.2006.03227.x. arXiv: 2006.03227 [cs, stat]. (Visited on 06/08/2022).

[106] C. Angermueller, D. Belanger, K. Murphy, D. Dohan, R. Deshpande, and L. Colwell. “Model-Based Reinforcement Learning for Biological Sequence Design”. In: *ICLR*. 2020, p. 23.

[107] L. Cheng, Z. Yang, B. Liao, C. Hsieh, and S. Zhang. *ODBO: Bayesian Optimization with Search Space Prescreening for Directed Protein Evolution*. May 2022. arXiv: 2205.09548 [cs, q-bio]. (Visited on 05/25/2022).

[108] Z. Yang, K. A. Milas, and A. D. White. *Now What Sequence? Pre-trained Ensembles for Bayesian Optimization of Protein Sequences*. Aug. 2022. DOI: 10.1101/2022.08.05.502972. (Visited on 08/08/2022).

[109] J. Yang, R. G. Lal, J. C. Bowden, R. Astudillo, M. A. Hameedi, S. Kaur, M. Hill, Y. Yue, and F. H. Arnold. *Active Learning-Assisted Directed Evolution*. July 2024. DOI: 10.1101/2024.07.27.605457. (Visited on 07/29/2024).

[110] T. T. Tran and T. S. Hy. "Protein Design by Directed Evolution Guided by Large Language Models". In: *IEEE Transactions on Evolutionary Computation* (Aug. 2024), pp. 1–1. DOI: 10.1109/TEVC.2024.3439690. (Visited on 05/02/2024).

[111] J. Subramanian, S. Sujit, N. Irtisam, U. Sain, D. Nowrouzezahrai, S. E. Kahou, and R. Islam. *Reinforcement Learning for Sequence Design Leveraging Protein Language Models*. July 2024. DOI: 10.48550/arXiv.2407.03154. arXiv: 2407.03154 [cs, q-bio]. (Visited on 07/04/2024).

[112] M. González-Duque, R. Michael, S. Bartels, Y. Zainchkovskyy, S. Hauberg, and W. Boomsma. "A Survey and Benchmark of High-Dimensional Bayesian Optimization of Discrete Sequences". In: *NeurIPS 2024 Track on Datasets and Benchmarks*. arXiv, June 2024. arXiv: 2406.04739 [cs, stat]. (Visited on 06/10/2024).

[113] H. Kim, M. Kim, T. Yun, S. Choi, E. Bengio, A. Hernández-García, and J. Park. *Improved Off-policy Reinforcement Learning in Biological Sequence Design*. Oct. 2024. (Visited on 10/08/2024).

[114] A. N. Amin, N. Gruver, Y. Kuang, L. Li, H. Elliott, C. McCarter, A. Raghu, P. Greenside, and A. G. Wilson. "Bayesian Optimization of Antibodies Informed by a Generative Model of Evolving Sequences". In: *International Conference on Learning Representations*. 2025. arXiv: 2412.07763 [stat]. (Visited on 02/03/2025).

[115] P. Notin et al. "ProteinGym: Large-Scale Benchmarks for Protein Design and Fitness Prediction". In: *NeurIPS 2023 Track on Datasets and Benchmarks*. Dec. 2023. DOI: 10.1101/2023.12.07.570727. (Visited on 01/22/2024).

[116] L. Zhang et al. *ProteinAligner: A Multi-modal Pretraining Framework for Protein Foundation Models*. Oct. 2024. DOI: 10.1101/2024.10.06.616870. (Visited on 10/07/2024).

[117] N. Anand, R. Eguchi, and P.-S. Huang. "Fully Differentiable Full-Atom Backbone Generation". In: *ICLR Workshop DeepGenStruct*. 2019, p. 10.

[118] I. Anishchenko et al. "De Novo Protein Design by Deep Network Hallucination". In: *Nature* 600.7889 (Dec. 2021), pp. 547–552. ISSN: 1476-4687. DOI: 10.1038/s41586-021-04184-w. (Visited on 05/16/2022).

[119] B. Lai, J. Xu, and M. McPartlon. *End-to-End Deep Structure Generative Model for Protein Design*. July 2022.

[120] C. Frank et al. "Scalable Protein Design Using Optimization in a Relaxed Sequence Space". In: *Science (New York, N.Y.)* 386.6720 (Oct. 2024), pp. 439–445. ISSN: 1095-9203. DOI: 10.1126/science.adq1741.

[121] N. Anand and T. Achim. *Protein Structure and Sequence Generation with Equivariant Denoising Diffusion Probabilistic Models*. May 2022. arXiv: 2205.15019 [cs, q-bio]. (Visited on 06/03/2022).

[122] J. L. Watson et al. "De Novo Design of Protein Structure and Function with RFdiffusion". In: *Nature* 620.7976 (Aug. 2023), pp. 1089–1100. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06415-8. (Visited on 02/14/2024).

[123] B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. "Diffusion Probabilistic Modeling of Protein Backbones in 3D for the Motif-Scaffolding Problem". In: *International Conference on Learning Representations*. arXiv:2206.04119. arXiv, 2023. arXiv: 2206.04119 [cs, q-bio, stat]. (Visited on 06/10/2022).

[124] J. Yim, B. L. Trippe, V. De Bortoli, E. Mathieu, A. Doucet, R. Barzilay, and T. Jaakkola. "SE(3) Diffusion Model with Application to Protein Backbone Generation". In: *Proceedings of the 40 Th International Conference on Machine Learning*. arXiv, May 2023. DOI: 10.48550/arXiv.2302.02277. arXiv: 2302.02277 [cs, q-bio, stat]. (Visited on 07/04/2023).

[125] A. E. Chu, J. Kim, L. Cheng, G. El Nesr, M. Xu, R. W. Shuai, and P.-S. Huang. "An All-Atom Protein Generative Model". In: *Proceedings of the National Academy of Sciences* 121.27 (July 2024), e2311500121. DOI: 10.1073/pnas.2311500121. (Visited on 07/03/2024).

[126] C. Hsu, C. Fannjiang, and J. Listgarten. "Generative Models for Protein Structures and Sequences". In: *Nature Biotechnology* 42.2 (Feb. 2024), pp. 196–199. ISSN: 1546-1696. DOI: 10.1038/s41587-023-02115-w. (Visited on 02/15/2024).

[127] X. Tang, H. Dai, E. Knight, F. Wu, Y. Li, T. Li, and M. Gerstein. "A Survey of Generative AI for De Novo Drug Design: New Frontiers in Molecule and Protein Generation". In: *Briefings in Bioinformatics* 25.4 (Feb. 2024), bbae338. DOI: 10.1093/bib/bbae338. arXiv: 2402.08703 [cs, q-bio]. (Visited on 02/15/2024).

[128] V. Zambaldi et al. *De Novo Design of High-Affinity Protein Binders with AlphaProteo*. Sept. 2024. DOI: 10.48550/arXiv.2409.08022. arXiv: 2409.08022 [q-bio]. (Visited on 09/13/2024).

[129] N. C. Frey et al. "Protein Discovery with Discrete Walk-Jump Sampling". In: *International Conference on Learning Representations*. arXiv, 2024. arXiv: 2306.12360 [cs, q-bio]. (Visited on 02/15/2024).

[130] J. S. Lee, J. Kim, and P. M. Kim. "Score-Based Generative Modeling for de Novo Protein Design". In: *Nature Computational Science* 3.5 (May 2023), pp. 382–392. ISSN: 2662-8457. DOI: 10.1038/s43588-023-00440-3. (Visited on 07/04/2023).

[131] S. Luo, Y. Su, X. Peng, S. Wang, J. Peng, and J. Ma. "Antigen-Specific Antibody Design and Optimization with Diffusion-Based Generative Models". In: *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. bioRxiv, July 2022.

[132] M. Pacesa et al. *BindCraft: One-Shot Design of Functional Protein Binders*. Oct. 2024. DOI: 10.1101/2024.09.30.615802. (Visited on 10/02/2024).

[133] Y. Jian, C. Wu, D. Reidenbach, and A. S. Krishnapriyan. *General Binding Affinity Guidance for Diffusion Models in Structure-Based Drug Design*. June 2024. DOI: 10.48550/arXiv.2406.16821. arXiv: 2406.16821 [physics, q-bio]. (Visited on 06/25/2024).

[134] Z. Zhang, W. X. Shen, Q. Liu, and M. Zitnik. "Efficient Generation of Protein Pockets with PocketGen". In: *Nature Machine Intelligence* 6.11 (Nov. 2024), pp. 1382–1395. ISSN: 2522-5839. DOI: 10.1038/s42256-024-00920-9. (Visited on 03/26/2025).

[135] L. Cao et al. “Design of Protein-Binding Proteins from the Target Structure Alone”. In: *Nature* (Mar. 2022), pp. 1–10. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04654-9. (Visited on 05/16/2022).

[136] N. R. Bennett et al. “Improving de Novo Protein Binder Design with Deep Learning”. In: *Nature Communications* 14.1 (May 2023), p. 2625. ISSN: 2041-1723. DOI: 10.1038/s41467-023-38328-5. (Visited on 03/26/2025).

[137] S. Vázquez Torres et al. “De Novo Designed Proteins Neutralize Lethal Snake Venom Toxins”. In: *Nature* 639.8053 (Mar. 2025), pp. 225–231. ISSN: 1476-4687. DOI: 10.1038/s41586-024-08393-x. (Visited on 03/14/2025).

[138] W. Ahern et al. *Atom Level Enzyme Active Site Scaffolding Using RFdiffusion2*. Apr. 2025. DOI: 10.1101/2025.04.09.648075. (Visited on 04/11/2025).

[139] A. Lauko et al. “Computational Design of Serine Hydrolases”. In: *Science* 0.0 (Feb. 2025), eadu2454. DOI: 10.1126/science.adu2454. (Visited on 03/14/2025).

[140] Z. Song, Y. Zhao, W. Shi, W. Jin, Y. Yang, and L. Li. “Generative Enzyme Design Guided by Functionally Important Sites and Small-Molecule Substrates”. In: *Proceedings of the 41 St International Conference on Machine Learning*. arXiv, May 2024. arXiv: 2405.08205 [cs]. (Visited on 05/15/2024).

[141] S. L. Lisanza et al. “Multistate and Functional Protein Design Using RoseTTAFold Sequence Space Diffusion”. In: *Nature Biotechnology* (Sept. 2024), pp. 1–11. ISSN: 1546-1696. DOI: 10.1038/s41587-024-02395-w. (Visited on 03/14/2025).

[142] C. Hua, Y. Liu, D. Zhang, O. Zhang, S. Luan, K. K. Yang, G. Wolf, D. Precup, and S. Zheng. *EnzymeFlow: Generating Reaction-specific Enzyme Catalytic Pockets through Flow Matching and Co-Evolutionary Dynamics*. Sept. 2024. DOI: 10.48550/arXiv.2410.00327. arXiv: 2410.00327 [cs, q-bio]. (Visited on 10/02/2024).

[143] S. Minami, N. Kobayashi, T. Sugiki, T. Nagashima, T. Fujiwara, R. Tatsumi-Koga, G. Chikenji, and N. Koga. “Exploration of Novel $A\beta$-Protein Folds through de Novo Design”. In: *Nature Structural & Molecular Biology* (July 2023), pp. 1–9. ISSN: 1545-9985. DOI: 10.1038/s41594-023-01029-0. (Visited on 07/12/2023).

[144] H. Lu et al. “Machine Learning-Aided Engineering of Hydrolases for PET Depolymerization”. In: *Nature* 604.7907 (Apr. 2022), pp. 662–667. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04599-z. (Visited on 04/30/2022).

[145] S. Maes, N. Deploey, F. Peelman, and S. Eyckerman. “Deep Mutational Scanning of Proteins in Mammalian Cells”. In: *Cell Reports Methods* 3.11 (Nov. 2023). ISSN: 2667-2375. DOI: 10.1016/j.crmeth.2023.100641. (Visited on 03/19/2025).

[146] H. Wei and X. Li. “Deep Mutational Scanning: A Versatile Tool in Systematically Mapping Genotypes to Phenotypes”. In: *Frontiers in Genetics* 14 (Jan. 2023). ISSN: 1664-8021. DOI: 10.3389/fgene.2023.1087267. (Visited on 03/19/2025).

[147] M. Gasperini, L. Starita, and J. Shendure. “The Power of Multiplexed Functional Analysis of Genetic Variants”. In: *Nature Protocols* 11.10 (Oct. 2016), pp. 1782–1787. ISSN: 1750-2799. DOI: 10.1038/nprot.2016.135. (Visited on 03/19/2025).

[148] D. Katrekar, Y. Xiang, N. Palmer, A. Saha, D. Meluzzi, and P. Mali. "Comprehensive Interrogation of the ADAR2 Deaminase Domain for Engineering Enhanced RNA Editing Activity and Specificity". In: *eLife* 11 (Jan. 2022). Ed. by T. W. Nilsen, J. L. Manley, N. Papavasiliou, and H. A. Hundley, e75555. ISSN: 2050-084X. DOI: 10.7554/eLife.75555. (Visited on 03/19/2025).

[149] K. K. Chan, D. Dorosky, P. Sharma, S. A. Abbasi, J. M. Dye, D. M. Kranz, A. S. Herbert, and E. Procko. "Engineering Human ACE2 to Optimize Binding to the Spike Protein of SARS Coronavirus 2". In: *Science* 369.6508 (Sept. 2020), pp. 1261–1265. DOI: 10.1126/science.abc0870. (Visited on 03/19/2025).

[150] P. Sharma, V. V. V. R. Marada, Q. Cai, M. Kizerwetter, Y. He, S. P. Wolf, K. Schreiber, H. Clausen, H. Schreiber, and D. M. Kranz. "Structure-Guided Engineering of the Affinity and Specificity of CARs against Tn-glycopeptides". In: *Proceedings of the National Academy of Sciences* 117.26 (June 2020), pp. 15148–15159. DOI: 10.1073/pnas.1920662117. (Visited on 03/19/2025).

[151] J. Tycko et al. "High-Throughput Discovery and Characterization of Human Transcriptional Effectors". In: *Cell* 183.7 (Dec. 2020), 2020–2035.e16. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2020.11.024. (Visited on 03/19/2025).

[152] M. V. Staller, E. Ramirez, S. R. Kotha, A. S. Holehouse, R. V. Pappu, and B. A. Cohen. "Directed Mutational Scanning Reveals a Balance between Acidic and Hydrophobic Residues in Strong Human Activation Domains". In: *Cell Systems* 13.4 (Apr. 2022), 334–345.e5. ISSN: 2405-4712, 2405-4720. DOI: 10.1016/j.cels.2022.01.002. (Visited on 03/19/2025).

[153] M. E. Garrett, H. L. Itell, K. H. D. Crawford, R. Basom, J. D. Bloom, and J. Overbaugh. "Phage-DMS: A Comprehensive Method for Fine Mapping of Antibody Epitopes". In: *iScience* 23.10 (Oct. 2020). ISSN: 2589-0042. DOI: 10.1016/j.isci.2020.101622. (Visited on 03/19/2025).

[154] J. D. Heredia, J. Park, R. J. Brubaker, S. K. Szymanski, K. S. Gill, and E. Procko. "Mapping Interaction Sites on Human Chemokine Receptors by Deep Mutational Scanning". In: *The Journal of Immunology* 200.11 (June 2018), pp. 3825–3839. ISSN: 0022-1767. DOI: 10.4049/jimmunol.1800343. (Visited on 03/19/2025).

[155] Z. Sun and T. Palzkill. "Deep Mutational Scanning Reveals the Active-Site Sequence Requirements for the Colistin Antibiotic Resistance Enzyme MCR-1". In: *mBio* 12.6 (Nov. 2021), e02776–21. DOI: 10.1128/mBio.02776-21. (Visited on 03/19/2025).

[156] T. N. Starr et al. "Deep Mutational Scanning of SARS-CoV-2 Receptor Binding Domain Reveals Constraints on Folding and ACE2 Binding". In: *Cell* 182.5 (Sept. 2020), 1295–1310.e20. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2020.08.012. (Visited on 03/19/2025).

[157] T. N. Starr et al. "Shifting Mutational Constraints in the SARS-CoV-2 Receptor-Binding Domain during Viral Evolution". In: *Science* 377.6604 (July 2022), pp. 420–424. DOI: 10.1126/science.abo7896. (Visited on 03/19/2025).

[158] A. C. Leonard, J. J. Weinstein, P. J. Steiner, A. H. Erbse, S. J. Fleishman, and T. A. Whitehead. "Stabilization of the SARS-CoV-2 Receptor Binding Domain by Protein Core Redesign and Deep Mutational Scanning". In: *Protein Engineering, Design and Selection* 35 (Feb. 2022), gzac002. ISSN: 1741-0126. DOI: 10.1093/protein/gzac002. (Visited on 03/19/2025).

[159] R. Lei, A. H. Garcia, T. J. C. Tan, Q. W. Teo, Y. Wang, X. Zhang, S. Luo, S. K. Nair, J. Peng, and N. C. Wu. "Mutational Fitness Landscape of Human Influenza H3N2 Neuraminidase". In: *Cell Reports* 42.1 (Jan. 2023). ISSN: 2211-1247. DOI: 10.1016/j.celrep.2022.111951. (Visited on 03/19/2025).

[160] J. M. Lee, J. Huddleston, M. B. Doud, K. A. Hooper, N. C. Wu, T. Bedford, and J. D. Bloom. "Deep Mutational Scanning of Hemagglutinin Helps Predict Evolutionary Fates of Human H3N2 Influenza Variants". In: *Proceedings of the National Academy of Sciences* 115.35 (Aug. 2018), E8276–E8285. DOI: 10.1073/pnas.1806133115. (Visited on 03/19/2025).

[161] C. Plesa, A. M. Sidore, N. B. Lubock, D. Zhang, and S. Kosuri. "Multiplexed Gene Synthesis in Emulsions for Exploring Protein Functional Landscapes". In: *Science* 359.6373 (Jan. 2018), pp. 343–347. DOI: 10.1126/science.aao5167. (Visited on 03/19/2025).

[162] L. M. Starita and S. Fields. "Deep Mutational Scanning: Library Construction, Functional Selection, and High-Throughput Sequencing". In: *Cold Spring Harbor Protocols* 2015.8 (Aug. 2015), pdb.prot085225. ISSN: 1940-3402, 1559-6095. DOI: 10.1101/pdb.prot085225. (Visited on 03/19/2025).

[163] Y. Nov. "When Second Best Is Good Enough: Another Probabilistic Look at Saturation Mutagenesis". In: *Applied and Environmental Microbiology* 78.1 (Jan. 2012), pp. 258–262. DOI: 10.1128/AEM.06265-11. (Visited on 03/19/2025).

[164] L. Tang, and R. and Jiang. "Construction of "Small-Intelligent" Focused Mutagenesis Libraries Using Well-Designed Combinatorial Degenerate Primers". In: *BioTechniques* 52.3 (Mar. 2012), pp. 149–158. ISSN: 0736-6205. DOI: 10.2144/000113820. (Visited on 03/19/2025).

[165] J. L. Lin-Goerke, and J. D. and Burczak. "PCR-Based Random Mutagenesis Using Manganese and Reduced dNTP Concentration". In: *BioTechniques* 23.3 (Sept. 1997), pp. 409–412. ISSN: 0736-6205. DOI: 10.2144/97233bm12. (Visited on 03/19/2025).

[166] S. Shafikhani, and V. and Schellenberger. "Generation of Large Libraries of Random Mutants in Bacillus Subtilis by PCR-Based Plasmid Multimerization". In: *BioTechniques* 23.2 (Aug. 1997), pp. 304–310. ISSN: 0736-6205. DOI: 10.2144/97232rr01. (Visited on 03/19/2025).

[167] J. Domingo, G. Diss, and B. Lehner. "Pairwise and Higher-Order Genetic Interactions during the Evolution of a tRNA". In: *Nature* 558.7708 (June 2018), pp. 117–121. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0170-7. (Visited on 03/19/2025).

[168] Y. Fan et al. "Encoding and Display Technologies for Combinatorial Libraries in Drug Discovery: The Coming of Age from Biology to Therapy". In: *Acta Pharmaceutica Sinica B* 14.8 (Aug. 2024), pp. 3362–3384. ISSN: 2211-3835. DOI: 10.1016/j.apsb.2024.04.006. (Visited on 03/18/2025).

[169] W. Mandecki, J. Y.-C. Chen, and N. Grihalde. “A Mathematical Model for Biopanning (Affinity Selection) Using Peptide Libraries on Filamentous Phage”. In: *Journal of Theoretical Biology* 176.4 (Oct. 1995), pp. 523–530. ISSN: 00225193. DOI: 10.1006/jtbi.1995.0218. (Visited on 10/17/2022).

[170] G. P. Smith and V. A. Petrenko. “Phage Display”. In: *Chemical Reviews* 97.2 (Apr. 1997), pp. 391–410. ISSN: 0009-2665. DOI: 10.1021/cr960065d. (Visited on 01/24/2024).

[171] B. Levitan. “Stochastic Modeling and Optimization of Phage Display”. In: *Journal of Molecular Biology* 277.4 (Apr. 1998), pp. 893–916. ISSN: 00222836. DOI: 10.1006/jmbi.1997.1555. (Visited on 10/17/2022).

[172] G. P. Smith. “Filamentous Fusion Phage: Novel Expression Vectors That Display Cloned Antigens on the Virion Surface”. In: *Science* 228.4705 (June 1985), pp. 1315–1317. DOI: 10.1126/science.4001944. (Visited on 03/18/2025).

[173] P. Model and M. Russel. “Filamentous Bacteriophage”. In: *The Bacteriophages*. Ed. by R. Calendar. The Viruses. New York: Plenum Press, 1988. ISBN: 978-0-306-42730-5 978-0-306-42853-1.

[174] J. Rakonjac and P. Model. “Roles of pIII in Filamentous Phage Assembly”. In: *Journal of Molecular Biology* 282.1 (Sept. 1998), pp. 25–41. ISSN: 00222836. DOI: 10.1006/jmbi.1998.2006. (Visited on 05/18/2022).

[175] M. Russel, H. B. Lowman, and T. Clackson. “Introduction to Phage Biology and Phage Display”. In: *Phage Display: A Practical Approach*. Practical Approach Series. Oxford: Oxford University Press, 2004, p. 26. ISBN: 978-0-19-963873-4.

[176] X.-D. Kong, V. Carle, C. Diaz-Perlas, K. Butler, and C. Heinis. “Generation of a Large Peptide Phage Display Library by Self-Ligation of Whole-Plasmid PCR Product”. In: *ACS Chemical Biology* 15.11 (Nov. 2020), pp. 2907–2915. ISSN: 1554-8937. DOI: 10.1021/acschembio.0c00497. (Visited on 03/18/2025).

[177] C. J. Hutchings and A. K. and Sato. “Phage Display Technology and Its Impact in the Discovery of Novel Protein-Based Drugs”. In: *Expert Opinion on Drug Discovery* 19.8 (Aug. 2024), pp. 887–915. ISSN: 1746-0441. DOI: 10.1080/17460441.2024.2367023. (Visited on 03/18/2025).

[178] Y. Zhang. “Evolution of Phage Display Libraries for Therapeutic Antibody Discovery”. In: *mAbs* 15.1 (Dec. 2023), p. 2213793. ISSN: 1942-0862. DOI: 10.1080/19420862.2023.2213793. (Visited on 03/18/2025).

[179] R. W. Roberts and J. W. Szostak. “RNA-peptide Fusions for the in Vitro Selection of Peptides and Proteins”. In: *Proceedings of the National Academy of Sciences* 94.23 (Nov. 1997), pp. 12297–12302. DOI: 10.1073/pnas.94.23.12297. (Visited on 03/19/2025).

[180] M. Newton, Y. Cabezas-Perusse, C. Ling Tong, and B. Seelig. “In Vitro Selection of Peptides and Proteins: Advantages of mRNA Display”. In: *ACS Synthetic Biology* 9.2 (Feb. 2020), pp. 181–190. ISSN: 2161-5063. DOI: 10.1021/acssynbio.9b00419. (Visited on 03/19/2025).

[181] H. Peacock and H. Suga. “Discovery of De Novo Macrocyclic Peptides by Messenger RNA Display”. In: *Trends in Pharmacological Sciences* 42.5 (May 2021), pp. 385–397. ISSN: 01656147. DOI: 10.1016/j.tips.2021.02.004. (Visited on 06/22/2022).

[182] I. Fukuda, K. Kojoh, N. Tabata, N. Doi, H. Takashima, E. Miyamoto-Sato, and H. Yanagawa. "In Vitro Evolution of Single-Chain Antibodies Using mRNA Display". In: *Nucleic Acids Research* 34.19 (Nov. 2006), e127–e127. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkl618. (Visited on 03/19/2025).

[183] H. Arai, S. Kumachi, and N. Nemoto. "cDNA Display: A Stable and Simple Genotype-Phenotype Coupling Using a Cell-Free Translation System". In: *Methods in Molecular Biology (Clifton, N.J.)* 2070 (2020), pp. 43–56. ISSN: 1940-6029. DOI: 10.1007/978-1-4939-9853-1_3.

[184] K. Tsuboyama, J. Dauparas, J. Chen, E. Laine, Y. Mohseni Behbahani, J. J. Weinstein, N. M. Mangan, S. Ovchinnikov, and G. J. Rocklin. "Mega-Scale Experimental Analysis of Protein Folding Stability in Biology and Design". In: *Nature* 620.7973 (Aug. 2023), pp. 434–444. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06328-6. (Visited on 03/19/2025).

[185] P. A. Romero and F. H. Arnold. "Exploring Protein Fitness Landscapes by Directed Evolution". In: *Nature Reviews. Molecular Cell Biology* 10.12 (Dec. 2009), pp. 866–876. ISSN: 1471-0080. DOI: 10.1038/nrm2805.

[186] L. S. Vidal, M. Isalan, J. T. Heap, and R. Ledesma-Amaro. "A Primer to Directed Evolution: Current Methodologies and Future Directions". In: *RSC Chemical Biology* 4.4 (2023), pp. 271–291. DOI: 10.1039/D2CB00231K. (Visited on 03/20/2025).

[187] K. Chen and F. H. Arnold. "Tuning the Activity of an Enzyme for Unusual Environments: Sequential Random Mutagenesis of Subtilisin E for Catalysis in Dimethylformamide." In: *Proceedings of the National Academy of Sciences* 90.12 (June 1993), pp. 5618–5622. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.90.12.5618. (Visited on 04/24/2022).

[188] L. Shi, P. Liu, Z. Tan, W. Zhao, J. Gao, Q. Gu, H. Ma, H. Liu, and L. Zhu. "Complete Depolymerization of PET Wastes by an Evolved PET Hydrolase from Directed Evolution". In: *Angewandte Chemie* 135.14 (2023), e202218390. ISSN: 1521-3757. DOI: 10.1002/ange.202218390. (Visited on 03/20/2025).

[189] L. Giver, A. Gershenson, P. O. Freskgard, and F. H. Arnold. "Directed Evolution of a Thermostable Esterase". In: *Proceedings of the National Academy of Sciences of the United States of America* 95.22 (Oct. 1998), pp. 12809–12813. ISSN: 0027-8424. DOI: 10.1073/pnas.95.22.12809.

[190] I. Sarkar, I. Hauber, J. Hauber, and F. Buchholz. "HIV-1 Proviral DNA Excision Using an Evolved Recombinase". In: *Science (New York, N.Y.)* 316.5833 (June 2007), pp. 1912–1915. ISSN: 1095-9203. DOI: 10.1126/science.1141453.

[191] T. H. Yoo, A. J. Link, and D. A. Tirrell. "Evolution of a Fluorinated Green Fluorescent Protein". In: *Proceedings of the National Academy of Sciences* 104.35 (Aug. 2007), pp. 13887–13890. DOI: 10.1073/pnas.0701904104. (Visited on 03/20/2025).

[192] Z. Wu, S. B. Jennifer Kan, R. D. Lewis, B. J. Wittmann, and F. H. Arnold. "Machine Learning-Assisted Directed Protein Evolution with Combinatorial Libraries". In: *Proceedings of the National Academy of Sciences of the United States of America* 116.18 (Apr. 2019), pp. 8852–8858. ISSN: 10916490. DOI: 10.1073/PNAS.1901979116/SUPPL_FILE/PNAS.1901979116.SAPP.PDF. arXiv: 1902.07231. (Visited on 03/21/2022).

[193] K. K. Yang, Z. Wu, and F. H. Arnold. “Machine-Learning-Guided Directed Evolution for Protein Engineering”. In: *Nature Methods* 16.8 (Aug. 2019), pp. 687–694. issn: 1548-7105. doi: 10.1038/s41592-019-0496-6. (Visited on 03/20/2025).

[194] B. J. Wittmann, K. E. Johnston, Z. Wu, and F. H. Arnold. “Advances in Machine Learning for Directed Evolution”. In: *Current Opinion in Structural Biology* 69 (Aug. 2021), pp. 11–18. issn: 1879033X. doi: 10.1016/j.sbi.2021.01.008. (Visited on 03/09/2021).

[195] R. R. Breaker, A. Banerji, and G. F. Joyce. “Continuous in Vitro Evolution of Bacteriophage RNA Polymerase Promoters”. In: *Biochemistry* 33.39 (Oct. 1994), pp. 11980–11986. issn: 0006-2960, 1520-4995. doi: 10.1021/bi00205a037. (Visited on 04/26/2022).

[196] M. C. Wright and G. F. Joyce. “Continuous in Vitro Evolution of Catalytic Function”. In: *Science* 276.5312 (Apr. 1997), pp. 614–617. issn: 0036-8075, 1095-9203. doi: 10.1126/science.276.5312.614. (Visited on 04/26/2022).

[197] K. M. Esvelt, J. C. Carlson, and D. R. Liu. “A System for the Continuous Directed Evolution of Biomolecules”. In: *Nature* 472.7344 (2011), pp. 499–503. issn: 00280836. doi: 10.1038/nature09929. (Visited on 02/08/2021).

[198] S. M. Miller, T. Wang, and D. R. Liu. “Phage-Assisted Continuous and Non-Continuous Evolution”. In: *Nature Protocols 2020 15:12* 15.12 (Nov. 2020), pp. 4101–4127. issn: 1750-2799. doi: 10.1038/s41596-020-00410-3. (Visited on 03/22/2022).

[199] B. C. Dickinson, M. S. Packer, A. H. Badran, and D. R. Liu. “A System for the Continuous Directed Evolution of Proteases Rapidly Reveals Drug-Resistance Mutations”. In: *Nature Communications* 5.1 (Oct. 2014), p. 5352. issn: 2041-1723. doi: 10.1038/ncomms6352. (Visited on 05/04/2022).

[200] T. R. Blum et al. “Phage-Assisted Evolution of Botulinum Neurotoxin Proteases with Reprogrammed Specificity”. In: *Science* 371.6531 (Feb. 2021), pp. 803–810. issn: 0036-8075. doi: 10.1126/science.abf5972. (Visited on 03/09/2021).

[201] B. P. Hubbard, A. H. Badran, J. A. Zuris, J. P. Guilinger, K. M. Davis, L. Chen, S. Q. Tsai, J. D. Sander, J. K. Joung, and D. R. Liu. “Continuous Directed Evolution of DNA-binding Proteins to Improve TALEN Specificity”. In: *Nature Methods* 12.10 (Oct. 2015), pp. 939–942. issn: 1548-7105. doi: 10.1038/nmeth.3515. (Visited on 03/20/2025).

[202] D. I. Bryson, C. Fan, L.-T. Guo, C. Miller, D. Söll, and D. R. Liu. “Continuous Directed Evolution of Aminoacyl-tRNA Synthetases”. In: *Nature Chemical Biology* 13.12 (Dec. 2017), pp. 1253–1260. issn: 1552-4469. doi: 10.1038/nchembio.2474. (Visited on 03/20/2025).

[203] J. Pu, J. Zinkus-Boltz, and B. C. Dickinson. “Evolution of a Split RNA Polymerase as a Versatile Biosensor Platform”. In: *Nature Chemical Biology* 13.4 (Apr. 2017), pp. 432–438. issn: 1552-4469. doi: 10.1038/nchembio.2299. (Visited on 03/20/2025).

[204] J. H. Hu et al. “Evolved Cas9 Variants with Broad PAM Compatibility and High DNA Specificity”. In: *Nature* 556.7699 (Apr. 2018), pp. 57–63. issn: 1476-4687. doi: 10.1038/nature26155. (Visited on 03/20/2025).

[205] S. M. Miller, T. Wang, P. B. Randolph, M. Arbab, M. W. Shen, T. P. Huang, Z. Matuszek, G. A. Newby, H. A. Rees, and D. R. Liu. "Continuous Evolution of SpCas9 Variants Compatible with Non-G PAMs". In: *Nature Biotechnology* 38.4 (Apr. 2020), pp. 471–481. ISSN: 1546-1696. DOI: 10.1038/s41587-020-0412-8. (Visited on 03/20/2025).

[206] L. Schmidheini, N. Mathis, K. F. Marquart, T. Rothgangl, L. Kissling, D. Böck, C. Chanez, J. P. Wang, M. Jinek, and G. Schwank. "Continuous Directed Evolution of a Compact CjCas9 Variant with Broad PAM Compatibility". In: *Nature Chemical Biology* 20.3 (Mar. 2024), pp. 333–343. ISSN: 1552-4469. DOI: 10.1038/s41589-023-01427-x. (Visited on 03/20/2025).

[207] T. Wang, A. H. Badran, T. P. Huang, and D. R. Liu. "Continuous Directed Evolution of Proteins with Improved Soluble Expression". In: *Nature Chemical Biology* 14.10 (Oct. 2018), pp. 972–980. ISSN: 1552-4469. DOI: 10.1038/s41589-018-0121-5. (Visited on 03/20/2025).

[208] T. B. Roth, B. M. Woolston, G. Stephanopoulos, and D. R. Liu. "Phage-Assisted Evolution of Bacillus Methanolicus Methanol Dehydrogenase 2". In: *ACS Synthetic Biology* 8.4 (Apr. 2019), pp. 796–806. DOI: 10.1021/acssynbio.8b00481. (Visited on 03/20/2025).

[209] B. W. Thuronyi et al. "Continuous Evolution of Base Editors with Expanded Target Compatibility and Improved Activity". In: *Nature Biotechnology* 37.9 (Sept. 2019), pp. 1070–1079. ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-019-0193-0. (Visited on 04/26/2022).

[210] M. F. Richter et al. "Phage-Assisted Evolution of an Adenine Base Editor with Improved Cas Domain Compatibility and Activity". In: *Nature Biotechnology* 38.7 (July 2020), pp. 883–891. ISSN: 1546-1696. DOI: 10.1038/s41587-020-0453-z. (Visited on 03/20/2025).

[211] E. Zhang, M. E. Neugebauer, N. A. Krasnow, and D. R. Liu. "Phage-Assisted Evolution of Highly Active Cytosine Base Editors with Enhanced Selectivity and Minimal Sequence Context Preference". In: *Nature Communications* 15.1 (Feb. 2024), p. 1697. ISSN: 2041-1723. DOI: 10.1038/s41467-024-45969-7. (Visited on 03/20/2025).

[212] J. Zinkus-Boltz, C. DeValk, and B. C. Dickinson. "A Phage-Assisted Continuous Selection Approach for Deep Mutational Scanning of Protein–Protein Interactions". In: *ACS Chemical Biology* 14.12 (Dec. 2019), pp. 2757–2767. ISSN: 1554-8929. DOI: 10.1021/acschembio.9b00669. (Visited on 03/20/2025).

[213] E. Goicoechea Serrano, C. Blázquez-Bondia, and A. Jaramillo. "T7 Phage-Assisted Evolution of Riboswitches Using Error-Prone Replication and Dual Selection". In: *Scientific Reports* 14.1 (Jan. 2024), p. 2377. ISSN: 2045-2322. DOI: 10.1038/s41598-024-52049-9. (Visited on 03/20/2025).

[214] J. L. Doman et al. "Phage-Assisted Evolution and Protein Engineering Yield Compact, Efficient Prime Editors". In: *Cell* 186.18 (Aug. 2023), 3983–4002.e26. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2023.07.039. (Visited on 03/20/2025).

[215] J. C. Carlson, A. H. Badran, D. A. Guggiana-Nilo, and D. R. Liu. "Negative Selection and Stringency Modulation in Phage-Assisted Continuous Evolution". In: *Nature Chemical Biology* 10.3 (Mar. 2014), pp. 216–222. ISSN: 1552-4469. DOI: 10.1038/nchembio.1453. (Visited on 03/20/2025).

[216] A. M. Leconte, B. C. Dickinson, D. D. Yang, I. A. Chen, B. Allen, and D. R. Liu. “A Population-Based Experimental Model for Protein Evolution: Effects of Mutation Rate and Selection Stringency on Evolutionary Outcomes”. In: *Biochemistry* 52.8 (Feb. 2013), pp. 1490–1499. ISSN: 00062960. DOI: 10.1021/bi3016185. (Visited on 06/22/2021).

[217] A. H. Badran and D. R. Liu. “Development of Potent in Vivo Mutagenesis Plasmids with Broad Mutational Spectra”. In: *Nature Communications* 6.1 (Oct. 2015), p. 8425. ISSN: 2041-1723. DOI: 10.1038/ncomms9425. (Visited on 05/09/2022).

[218] J. G. English, R. H. J. Olsen, K. Lansu, M. Patel, K. White, A. S. Cockrell, D. Singh, R. T. Strachan, D. Wacker, and B. L. Roth. “VEGAS as a Platform for Facile Directed Evolution in Mammalian Cells”. In: *Cell* 178.3 (July 2019), 748–761.e17. ISSN: 1097-4172. DOI: 10.1016/j.cell.2019.05.051.

[219] G. Rix, E. J. Watkins-Dulaney, P. J. Almhjell, C. E. Boville, F. H. Arnold, and C. C. Liu. “Scalable Continuous Evolution for the Generation of Diverse Enzyme Variants Encompassing Promiscuous Activities”. In: *Nature Communications* 11.1 (Nov. 2020), p. 5644. ISSN: 2041-1723. DOI: 10.1038/s41467-020-19539-6. (Visited on 03/20/2025).

[220] G. Rix, R. L. Williams, V. J. Hu, A. Spinner, A. (Pisera, D. S. Marks, and C. C. Liu. “Continuous Evolution of User-Defined Genes at 1 Million Times the Genomic Mutation Rate”. In: *Science* 386.6722 (Nov. 2024), eadm9073. DOI: 10.1126/science.adm9073. (Visited on 03/20/2025).

[221] A. M. Paulk, R. L. Williams, and C. C. Liu. “Rapidly Inducible Yeast Surface Display for Antibody Evolution with OrthoRep”. In: *ACS Synthetic Biology* 13.8 (July 2024), pp. 2629–2634. DOI: 10.1021/acssynbio.4c00370. (Visited on 05/25/2024).

[222] E. A. DeBenedictis, E. J. Chory, D. W. Gretton, B. Wang, S. Golas, and K. M. Esvelt. “Systematic Molecular Evolution Enables Robust Biomolecule Discovery”. In: *Nature Methods* (Dec. 2021), pp. 1–10. ISSN: 1548-7105. DOI: 10.1038/s41592-021-01348-4. (Visited on 12/30/2021).

[223] S. K. Lear and S. L. Shipman. “Molecular Recording: Transcriptional Data Collection into the Genome”. In: *Current Opinion in Biotechnology* 79 (Feb. 2023), p. 102855. ISSN: 0958-1669. DOI: 10.1016/j.copbio.2022.102855. (Visited on 03/21/2025).

[224] H. Jang and S. S. Yim. “Toward DNA-Based Recording of Biological Processes”. In: *International Journal of Molecular Sciences* 25.17 (Jan. 2024), p. 9233. ISSN: 1422-0067. DOI: 10.3390/ijms25179233. (Visited on 03/21/2025).

[225] L. C. Greig, M. B. Woodworth, M. J. Galazo, H. Padmanabhan, and J. D. Macklis. “Molecular Logic of Neocortical Projection Neuron Specification, Development and Diversity”. In: *Nature Reviews Neuroscience* 14.11 (Nov. 2013), pp. 755–769. ISSN: 1471-0048. DOI: 10.1038/nrn3586. (Visited on 03/21/2025).

[226] S. J. Franco, C. Gil-Sanz, I. Martinez-Garay, A. Espinosa, S. R. Harkins-Perry, C. Ramos, and U. Müller. “Fate-Restricted Neural Progenitors in the Mammalian Cerebral Cortex”. In: *Science* 337.6095 (Aug. 2012), pp. 746–749. DOI: 10.1126/science.1223616. (Visited on 03/21/2025).

[227] L. Yang, A. A. Nielsen, J. Fernandez-Rodriguez, C. J. McClune, M. T. Laub, T. K. Lu, and C. A. Voigt. “Permanent Genetic Memory with >1 Byte Capacity”. In: *Nature methods* 11.12 (Oct. 2014), p. 1261. DOI: 10.1038/nmeth.3147. (Visited on 03/24/2025).

[228] N. Roquet, A. P. Soleimany, A. C. Ferris, S. Aaronson, and T. K. Lu. “Synthetic Recombinase-Based State Machines in Living Cells”. In: *Science* 353.6297 (July 2016), aad8559. DOI: 10.1126/science.aad8559. (Visited on 03/24/2025).

[229] F. Farzadfard and T. K. Lu. “Genomically Encoded Analog Memory with Precise in Vivo DNA Writing in Living Cell Populations”. In: *Science* 346.6211 (Nov. 2014), p. 1256272. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1256272. (Visited on 03/24/2025).

[230] F. A. Ran, P. D. Hsu, J. Wright, V. Agarwala, D. A. Scott, and F. Zhang. “Genome Engineering Using the CRISPR-Cas9 System”. In: *Nature protocols* 8.11 (Oct. 2013), p. 2281. DOI: 10.1038/nprot.2013.143. (Visited on 03/24/2025).

[231] L. Cong et al. “Multiplex Genome Engineering Using CRISPR/Cas Systems”. In: *Science* 339.6121 (Feb. 2013), pp. 819–823. DOI: 10.1126/science.1231143. (Visited on 04/26/2022).

[232] A. McKenna, G. M. Findlay, J. A. Gagnon, M. S. Horwitz, A. F. Schier, and J. Shendure. “Whole-Organism Lineage Tracing by Combinatorial and Cumulative Genome Editing”. In: *Science* 353.6298 (July 2016), aaf7907. DOI: 10.1126/science.aaf7907. (Visited on 03/24/2025).

[233] S. D. Perli, C. H. Cui, and T. K. Lu. “Continuous Genetic Recording with Self-Targeting CRISPR-Cas in Human Cells”. In: *Science* 353.6304 (Sept. 2016), aag0511. DOI: 10.1126/science.aag0511. (Visited on 03/24/2025).

[234] T. B. Loveless et al. “Lineage Tracing and Analog Recording in Mammalian Cells by Single-Site DNA Writing”. In: *Nature Chemical Biology* 17.6 (June 2021), pp. 739–747. ISSN: 1552-4469. DOI: 10.1038/s41589-021-00769-8. (Visited on 03/24/2025).

[235] A. C. Komor, Y. B. Kim, M. S. Packer, J. A. Zuris, and D. R. Liu. “Programmable Editing of a Target Base in Genomic DNA without Double-Stranded DNA Cleavage”. In: *Nature* 533.7603 (Apr. 2016), pp. 420–424. ISSN: 0028-0836. DOI: 10.1038/nature17946. (Visited on 04/26/2022).

[236] N. M. Gaudelli, A. C. Komor, H. A. Rees, M. S. Packer, A. H. Badran, D. I. Bryson, and D. R. Liu. “Programmable Base Editing of A•T to G•C in Genomic DNA without DNA Cleavage”. In: *Nature* 551.7681 (Nov. 2017), pp. 464–471. ISSN: 1476-4687. DOI: 10.1038/nature24644. (Visited on 04/26/2022).

[237] A. C. Komor, K. T. Zhao, M. S. Packer, N. M. Gaudelli, A. L. Waterbury, L. W. Koblan, Y. B. Kim, A. H. Badran, and D. R. Liu. “Improved Base Excision Repair Inhibition and Bacteriophage Mu Gam Protein Yields C:G-to-T:A Base Editors with Higher Efficiency and Product Purity”. In: *Science Advances* 3.8 (2017), eaao4774. DOI: 10.1126/sciadv.aao4774. (Visited on 04/26/2022).

[238] L. W. Koblan, J. L. Doman, C. Wilson, J. M. Levy, T. Tay, G. A. Newby, J. P. Maianti, A. Raguram, and D. R. Liu. “Improving Cytidine and Adenine Base Editors by Expression Optimization and Ancestral Reconstruction”. In: *Nature Biotechnology* 36.9 (Oct. 2018), pp. 843–846. ISSN: 1546-1696. DOI: 10.1038/nbt.4172. (Visited on 04/26/2022).

[239] J. L. Doman, A. Raguram, G. A. Newby, and D. R. Liu. “Evaluation and Minimization of Cas9-independent off-Target DNA Editing by Cytosine Base Editors”. In: *Nature Biotechnology* 38.5 (May 2020), pp. 620–628. ISSN: 1546-1696. DOI: 10.1038/s41587-020-0414-6. (Visited on 04/26/2022).

[240] A. V. Anzalone et al. “Search-and-Replace Genome Editing without Double-Strand Breaks or Donor DNA”. In: *Nature* 576.7785 (Dec. 2019), pp. 149–157. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1711-4. (Visited on 03/24/2025).

[241] A. V. Anzalone, X. D. Gao, C. J. Podracky, A. T. Nelson, L. W. Koblan, A. Raguram, J. M. Levy, J. A. M. Mercer, and D. R. Liu. “Programmable Deletion, Replacement, Integration, and Inversion of Large DNA Sequences with Twin Prime Editing”. In: *Nature biotechnology* 40.5 (Dec. 2021), p. 731. DOI: 10.1038/s41587-021-01133-w. (Visited on 03/24/2025).

[242] P. J. Chen and D. R. Liu. “Prime Editing for Precise and Highly Versatile Genome Manipulation”. In: *Nature Reviews Genetics* 24.3 (Mar. 2023), pp. 161–177. ISSN: 1471-0064. DOI: 10.1038/s41576-022-00541-1. (Visited on 03/24/2025).

[243] W. Tang and D. R. Liu. “Rewritable Multi-Event Analog Recording in Bacterial and Mammalian Cells”. In: *Science (New York, N.Y.)* 360.6385 (Apr. 2018), eaap8992. ISSN: 0036-8075. DOI: 10.1126/science.aap8992. (Visited on 04/26/2022).

[244] F. Farzadfard, N. Gharaei, Y. Higashikuni, G. Jung, J. Cao, and T. K. Lu. “Single-Nucleotide-Resolution Computing and Memory in Living Cells”. In: *Molecular Cell* 75.4 (Aug. 2019), 769–780.e4. ISSN: 1097-2765. DOI: 10.1016/j.molcel.2019.07.011. (Visited on 03/24/2025).

[245] J. Choi et al. “A Time-Resolved, Multi-Symbol Molecular Recorder via Sequential Genome Editing”. In: *Nature* 608.7921 (Aug. 2022), pp. 98–107. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04922-8. (Visited on 03/21/2025).

[246] B. Tu, V. Sundar, and K. Esvelt. *An Ultra-High-Throughput Method for Measuring Biomolecular Activities.* Mar. 2024. DOI: 10.1101/2022.03.09.483646. (Visited on 03/26/2024).

[247] W. G. Dougherty, T. D. Parks, S. M. Cary, J. F. Bazan, and R. J. Fletterick. “Characterization of the Catalytic Residues of the Tobacco Etch Virus 49-kDa Proteinase”. In: *Virology* 172.1 (Sept. 1989), pp. 302–310. ISSN: 0042-6822. DOI: 10.1016/0042-6822(89)90132-3.

[248] G. Kostallas, P.-Å. Löfdahl, and P. Samuelson. “Substrate Profiling of Tobacco Etch Virus Protease Using a Novel Fluorescence-Assisted Whole-Cell Assay”. In: *PLOS ONE* 6.1 (Jan. 2011), e16136. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0016136. (Visited on 05/04/2022).

[249] R. B. Kapust, J. Tözsér, J. D. Fox, D. Anderson, S. Cherry, T. D. Copeland, and D. S. Waugh. “Tobacco Etch Virus Protease: Mechanism of Autolysis and Rational Design of Stable Mutants with Wild-Type Catalytic Proficiency”. In: *Protein Engineering, Design and Selection* 14.12 (Dec. 2001), pp. 993–1000. ISSN: 1741-0126. DOI: 10.1093/protein/14.12.993. (Visited on 03/24/2025).

[250] K. H. Sumida et al. “Improving Protein Expression, Stability, and Function with ProteinMPNN”. In: *Journal of the American Chemical Society* 146.3 (Jan. 2024), pp. 2054–2061. ISSN: 0002-7863. DOI: 10.1021/jacs.3c10941. (Visited on 03/31/2025).

[251] K. D. Verhoeven, O. C. Altstadt, and S. N. Savinov. “Intracellular Detection and Evolution of Site-Specific Proteases Using a Genetic Selection System”. In: *Applied Biochemistry and Biotechnology* 166.5 (Mar. 2012), pp. 1340–1354. ISSN: 1559-0291. DOI: 10.1007/s12010-011-9522-6. (Visited on 03/24/2025).

[252] C. Renicke, R. Spadaccini, and C. Taxis. “A Tobacco Etch Virus Protease with Increased Substrate Tolerance at the P1’ Position”. In: *PLoS ONE* 8.6 (June 2013), e67915. DOI: 10.1371/journal.pone.0067915. (Visited on 03/24/2025).

[253] S. W. Meister, L. Parks, L. Kolmar, A. M. Borras, S. Ståhl, and J. Löfblom. “Engineering of TEV Protease Variants with Redesigned Substrate Specificity”. In: *Biotechnology Journal* 18.11 (2023), p. 2200625. ISSN: 1860-7314. DOI: 10.1002/biot.202200625. (Visited on 03/24/2025).

[254] C. Lu, J. H. Lubin, V. V. Sarma, S. Z. Stentz, G. Wang, S. Wang, and S. D. Khare. “Prediction and Design of Protease Enzyme Specificity Using a Structure-Aware Graph Convolutional Network”. In: *Proceedings of the National Academy of Sciences* 120.39 (Sept. 2023), e2303590120. DOI: 10.1073/pnas.2303590120. (Visited on 09/22/2023).

[255] M. W. Shen, K. T. Zhao, and D. R. Liu. “Reconstruction of Evolving Gene Variants and Fitness from Short Sequencing Reads”. In: *Nature Chemical Biology* (Oct. 2021), pp. 1–11. ISSN: 1552-4469. DOI: 10.1038/s41589-021-00876-6. (Visited on 10/11/2021).

[256] Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au. “Nanopore Sequencing Technology, Bioinformatics and Applications”. In: *Nature Biotechnology* 39.11 (Nov. 2021), pp. 1348–1365. ISSN: 1546-1696. DOI: 10.1038/s41587-021-01108-x. (Visited on 12/02/2024).

[257] M. Jain, H. E. Olsen, B. Paten, and M. Akeson. “The Oxford Nanopore MinION: Delivery of Nanopore Sequencing to the Genomics Community”. In: *Genome Biology* 17.1 (Nov. 2016), p. 239. ISSN: 1474-760X. DOI: 10.1186/s13059-016-1103-0. (Visited on 05/03/2022).

[258] K. Shafin et al. “Haplotype-Aware Variant Calling with PEPPER-Margin-DeepVariant Enables High Accuracy in Nanopore Long-Reads”. In: *Nature Methods* 18.11 (Nov. 2021), pp. 1322–1332. ISSN: 1548-7105. DOI: 10.1038/s41592-021-01299-w. (Visited on 04/24/2022).

[259] F. Obermeyer et al. “Analysis of 6.4 Million SARS-CoV-2 Genomes Identifies Mutations Associated with Fitness”. In: *Science* 376.6599 (June 2022), pp. 1327–1332. DOI: 10.1126/science.abm1208. (Visited on 03/24/2025).

[260] P. Stefanelli et al. “Co-Circulation of SARS-CoV-2 Alpha and Gamma Variants in Italy, February and March 2021”. In: *Eurosurveillance* 27.5 (Feb. 2022), p. 2100429. ISSN: 1560-7917. DOI: 10.2807/1560-7917.ES.2022.27.5.2100429. (Visited on 03/24/2025).

[261] E. Volz et al. “Assessing Transmissibility of SARS-CoV-2 Lineage B.1.1.7 in England”. In: *Nature* 593.7858 (May 2021), pp. 266–269. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03470-x. (Visited on 03/24/2025).

[262] H. S. Vöhringer et al. “Genomic Reconstruction of the SARS-CoV-2 Epidemic in England”. In: *Nature* 600.7889 (Dec. 2021), pp. 506–511. ISSN: 1476-4687. DOI: 10.1038/s41586-021-04069-y. (Visited on 03/24/2025).

[263] N. G. Davies et al. “Estimated Transmissibility and Impact of SARS-CoV-2 Lineage B.1.1.7 in England”. In: *Science* 372.6538 (Apr. 2021), eabg3055. DOI: 10.1126/science.abg3055. (Visited on 03/24/2025).

[264] F. Campbell et al. “Increased Transmissibility and Global Spread of SARS-CoV-2 Variants of Concern as at June 2021”. In: *Eurosurveillance* 26.24 (June 2021), p. 2100509. ISSN: 1560-7917. DOI: 10.2807/1560-7917.ES.2021.26.24.2100509. (Visited on 03/24/2025).

[265] N. C. Wu, L. Dai, C. A. Olson, J. O. Lloyd-Smith, and R. Sun. “Adaptation in Protein Fitness Landscapes Is Facilitated by Indirect Paths”. In: *eLife* 5.JULY (July 2016). ISSN: 2050084X. DOI: 10.7554/eLife.16965. (Visited on 02/08/2021).

[266] C. Dallago, J. Mou, K. E. Johnston, B. J. Wittmann, N. Bhattacharya, S. Goldman, A. Madani, and K. K. Yang. “FLIP: Benchmark Tasks in Fitness Landscape Inference for Proteins”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks 2021)* (Jan. 2022). DOI: 10.1101/2021.11.09.467890. (Visited on 05/13/2022).

[267] G. M. S. Ross, D. Filippini, M. W. F. Nielen, and G. I. Salentijn. “Unraveling the Hook Effect: A Comprehensive Study of High Antigen Concentration Effects in Sandwich Lateral Flow Immunoassays”. In: *Analytical Chemistry* 92.23 (Dec. 2020), pp. 15587–15595. ISSN: 0003-2700. DOI: 10.1021/acs.analchem.0c03740. (Visited on 12/02/2024).

[268] C. Furusawa, T. Suzuki, A. Kashiwagi, T. Yomo, and K. Kaneko. “Ubiquity of Log-Normal Distributions in Intra-Cellular Reaction Dynamics”. In: *Biophysics* 1 (Apr. 2005), p. 25. DOI: 10.2142/biophysics.1.25. (Visited on 12/02/2024).

[269] V. Sundar, B. Tu, L. Guan, and K. Esvelt. “FLIGHTED: Inferring Fitness Landscapes from Noisy High-Throughput Experimental Data”. In: *Machine Learning in Structural Biology Workshop, Neural Information Processing Systems*. 2023.

[270] V. Sundar, B. Tu, L. Guan, and K. M. Esvelt. “A New Ultra-High-Throughput Assay for Measuring Protein Fitness”. In: *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*. Apr. 2024. (Visited on 03/13/2025).

[271] V. Sundar, B. Tu, L. Guan, and K. Esvelt. *FLIGHTED: Inferring Fitness Landscapes from Noisy High-Throughput Experimental Data*. Mar. 2024. DOI: 10.1101/2024.03.26.586797. (Visited on 03/27/2024).

[272] F.-Z. Li, A. P. Amini, Y. Yue, K. K. Yang, and A. X. Lu. “Feature Reuse and Scaling: Understanding Transfer Learning with Protein Language Models”. In: *Proceedings of the 41st International Conference on Machine Learning*. 2024.

[273] J. Fernandez-de-Cossio-Diaz, G. Uguzzoni, and A. Pagnani. “Unsupervised Inference of Protein Fitness Landscape from Deep Mutational Scan”. In: *Molecular Biology and Evolution* 38.1 (Jan. 2021). Ed. by R. Nielsen, pp. 318–328. ISSN: 1537-1719. DOI: 10.1093/molbev/msaa204. (Visited on 03/08/2021).

[274] A. F. Rubin, H. Gelman, N. Lucas, S. M. Bajjalieh, A. T. Papenfuss, T. P. Speed, and D. M. Fowler. “A Statistical Framework for Analyzing Deep Mutational Scanning Data”. In: *Genome Biology* 18.1 (Aug. 2017), p. 150. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1272-5. (Visited on 09/12/2023).

[275] A. Busia and J. Listgarten. "MBE: Model-Based Enrichment Estimation and Prediction for Differential Sequencing Data". In: *Genome Biology* 24.1 (Oct. 2023), p. 218. ISSN: 1474-760X. DOI: 10.1186/s13059-023-03058-w. (Visited on 03/26/2025).

[276] L. Sesta, A. Pagnani, J. Fernandez-de-Cossio-Diaz, and G. Uguzzoni. "Inference of Annealed Protein Fitness Landscapes with AnnealDCA". In: *PLOS Computational Biology* 20.2 (Feb. 2024), e1011812. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1011812. (Visited on 02/20/2024).

[277] D. Zhu et al. "Optimal Trade-off Control in Machine Learning–Based Library Design, with Application to Adeno-Associated Virus (AAV) for Gene Therapy". In: *Science Advances* 10.4 (Jan. 2024), eadj3786. DOI: 10.1126/sciadv.adj3786. (Visited on 02/08/2024).

[278] M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. "Stochastic Variational Inference". In: *Journal of Machine Learning Research* (May 2013), pp. 1303–1347. DOI: 10.48550/arXiv.1206.7051. arXiv: 1206.7051 [cs, stat]. (Visited on 06/21/2024).

[279] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*. arXiv:1312.6114. arXiv, May 2014. arXiv: 1312.6114 [cs, stat]. (Visited on 06/08/2022).

[280] K. Sohn, H. Lee, and X. Yan. "Learning Structured Output Representation Using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. (Visited on 09/11/2023).

[281] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. "Pyro: Deep Universal Probabilistic Programming". In: *Journal of Machine Learning Research* 20 (2019), pp. 1–6.

[282] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *33rd Conference on Neural Information Processing Systems*. 2019.

[283] S. Matuszewski, M. E. Hildebrandt, A.-H. Ghenu, J. D. Jensen, and C. Bank. "A Statistical Guide to the Design of Deep Mutational Scanning Experiments". In: *Genetics* 204.1 (Sept. 2016), pp. 77–87. ISSN: 1943-2631. DOI: 10.1534/genetics.116.190462. (Visited on 02/07/2024).

[284] C. Z. Mooney, R. D. Duval, and R. Duvall. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. SAGE, Aug. 1993. ISBN: 978-0-8039-5381-9.

[285] A. Raj and A. van Oudenaarden. "Nature, Nurture, or Chance: Stochastic Gene Expression and Its Consequences". In: *Cell* 135.2 (Oct. 2008), pp. 216–226. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2008.09.050. (Visited on 05/19/2022).

[286] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open Source Scientific Tools for Python*. 2001.

[287] P. T. Boggs and J. E. Rogers. "Orthogonal Distance Regression". In: *US Department of Commerce, National Institute of Standards and Technology* (1989).

[288] N. Thomas et al. *Engineering of Highly Active and Diverse Nuclease Enzymes by Combining Machine Learning and Ultra-High-Throughput Screening*. Mar. 2024. DOI: 10.1101/2024.03.21.585615. (Visited on 03/25/2024).

[289] C. Hsu, H. Nisonoff, C. Fannjiang, and J. Listgarten. “Learning Protein Fitness Models from Evolutionary and Assay-Labeled Data”. In: *Nature Biotechnology* (2022). (Visited on 01/19/2022).

[290] A. Elnaggar et al. “ProtTrans: Towards Cracking the Language of Life’s Code Through Self-Supervised Learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.8 (2021). DOI: 10.1101/2020.07.12.199554. (Visited on 05/16/2022).

[291] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives. *Language Models of Protein Sequences at the Scale of Evolution Enable Accurate Structure Prediction*. July 2022.

[292] T. A. Hopf, J. B. Ingraham, F. J. Poelwijk, C. P. I. Schärfe, M. Springer, C. Sander, and D. S. Marks. “Mutation Effects Predicted from Sequence Co-Variation”. In: *Nature Biotechnology* 35.2 (Feb. 2017), pp. 128–135. ISSN: 1546-1696. DOI: 10.1038/nbt.3769.

[293] P. A. Curran. *Monte Carlo Error Analyses of Spearman’s Rank Test*. May 2015. arXiv: 1411.3816 `[astro-ph, physics:physics, stat]`. (Visited on 11/06/2023).

[294] E. N. Weinstein, M. G. Gollub, A. Slabodkin, C. L. Gardner, K. Dobbs, X.-B. Cui, A. N. Amin, G. M. Church, and E. B. Wood. *Manufacturing-Aware Generative Model Architectures Enable Biological Sequence Design and Synthesis at Petascale*. Sept. 2024. DOI: 10.1101/2024.09.13.612900. (Visited on 05/08/2025).

[295] E. N. Weinstein, A. N. Amin, W. S. Grathwohl, D. Kassler, J. Disset, and D. Marks. “Optimal Design of Stochastic DNA Synthesis Protocols Based on Generative Sequence Models”. In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. PMLR, May 2022, pp. 7450–7482. (Visited on 05/08/2025).

[296] G. Lemaıtre and F. Nogueira. “Imbalanced-Learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning”. In: *Journal of Machine Learning Research* 18 (2017), pp. 1–5.

[297] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. “Learning from Class-Imbalanced Data: Review of Methods and Applications”. In: *Expert Systems with Applications* 73 (May 2017), pp. 220–239. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2016.12.035. (Visited on 01/03/2025).

[298] G. M. Weiss. “Foundations of Imbalanced Learning”. In: *Imbalanced Learning*. John Wiley & Sons, Ltd, 2013. Chap. 2, pp. 13–41. ISBN: 978-1-118-64610-6. DOI: 10.1002/9781118646106.ch2. (Visited on 01/03/2025).

[299] H. He and E. A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1558-2191. DOI: 10.1109/TKDE.2008.239. (Visited on 01/03/2025).

[300] S. Geman and D. Geman. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (Nov. 1984), pp. 721–741. ISSN: 1939-3539. DOI: 10.1109/TPAMI.1984.4767596. (Visited on 03/06/2025).

[301] W. Grathwohl, K. Swersky, M. Hashemi, D. Duvenaud, and C. J. Maddison. “Oops I Took A Gradient: Scalable Sampling for Discrete Distributions”. In: *Proceedings of the 38 Th International Conference on Machine Learning*. June 2021. arXiv: 2102.04509 [cs]. (Visited on 02/15/2024).

[302] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* 21.6 (June 1953), pp. 1087–1092. ISSN: 0021-9606. DOI: 10.1063/1.1699114. (Visited on 03/06/2025).

[303] A. Kirjner, J. Yim, R. Samusevich, T. Jaakkola, R. Barzilay, and I. Fiete. “Optimizing Protein Fitness Using Gibbs Sampling with Graph-based Smoothing”. In: *Proceedings of the 40 Th International Conference on Machine Learning*. July 2023. arXiv: 2307.00494 [cs, q-bio, stat]. (Visited on 07/04/2023).

[304] R. Zhang, X. Liu, and Q. Liu. “A Langevin-like Sampler for Discrete Distributions”. In: *Proceedings of the 39 Th International Conference on Machine Learning*. June 2022. arXiv: 2206.09914 [cs, stat]. (Visited on 07/27/2022).

[305] A. E. Chu, L. Cheng, G. E. Nesr, M. Xu, and P.-S. Huang. *An All-Atom Protein Generative Model*. May 2023. DOI: 10.1101/2023.05.24.542194. (Visited on 07/04/2023).

[306] N. Gruver, S. Stanton, N. C. Frey, T. G. J. Rudner, I. Hotzel, J. Lafrance-Vanasse, A. Rajpal, K. Cho, and A. G. Wilson. “Protein Design with Guided Discrete Diffusion”. In: *Neural Information Processing Systems*. May 2023. DOI: 10.48550/arXiv.2305.20009. arXiv: 2305.20009 [cs, q-bio]. (Visited on 07/04/2023).

[307] J. Hoffmann et al. “Training Compute-Optimal Large Language Models”. In: *36th Conference on Neural Information Processing Systems*. arXiv, Mar. 2022. DOI: 10.48550/arXiv.2203.15556. arXiv: 2203.15556 [cs]. (Visited on 04/09/2025).

[308] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations*. arXiv, 2022. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685 [cs]. (Visited on 04/15/2025).

[309] N. Ding et al. “Parameter-Efficient Fine-Tuning of Large-Scale Pre-Trained Language Models”. In: *Nature Machine Intelligence* 5.3 (Mar. 2023), pp. 220–235. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00626-4. (Visited on 04/15/2025).

[310] R. Schmirler, M. Heinzinger, and B. Rost. “Fine-Tuning Protein Language Models Boosts Predictions across Diverse Tasks”. In: *Nature Communications* 15.1 (Aug. 2024), p. 7407. ISSN: 2041-1723. DOI: 10.1038/s41467-024-51844-2. (Visited on 08/29/2024).

[311] N. D. Rawlings, A. J. Barrett, P. D. Thomas, X. Huang, A. Bateman, and R. D. Finn. “The MEROPS Database of Proteolytic Enzymes, Their Substrates and Inhibitors in 2017 and a Comparison with Peptidases in the PANTHER Database”. In: *Nucleic Acids Research* 46.D1 (Jan. 2018), pp. D624–D632. ISSN: 0305-1048. DOI: 10.1093/nar/gkx1134. (Visited on 04/15/2025).

[312] T. Chen, Y. Zhang, and P. Chatterjee. *moPPIt: De Novo Generation of Motif-Specific Binders with Protein Language Models*. Aug. 2024. DOI: 10.1101/2024.07.31.606098. (Visited on 04/15/2025).