Sequential Strategies for Automated Science and Protein Engineering

Trevor S. Frisby

May 2023

CMU-CB-23-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research is funded in part by : Pennsylvania Department of Health (subaward from University of Pittsburgh) Big Data in Health Research under the Commonwealth Universal Research Enhancement (C.U.R.E.) program, National Institutes of Health National Research Service Award Grant #2T32EB009403-11, and National Institutes of Health National Research Service Award Grant #5T32EB009403-10.

To my parents, for your constant love and support.

Acknowledgments

Maybe the real dissertation is the friends we made along the way. I would like to thank all the members of CPCB who helped make the good times good, and the bad times a little less bad. This includes (but is not limited to):

- Omer Acar
- Paul Francoeur
- Emilee Holtzapple
- Jonathan King
- Amanda Kowalczyk
- Samantha Panakkal
- Cathy Su
- Laura Tung
- Jennifer Williams

I regret that the Covid pandemic interfered with some of these friendships, but I appreciate each and every one of you.

I would like to thank my advisor, Chris, for providing mentorship these last six years, and continuing to do so even after your own professional transition from CMU to Amgen. I feel I've grown as a scientist by following your lead, and am thankful for having had this opportunity to work with you.

This thesis would not be possible without the abundance of financial and technical support received along the way. I would like to thank Brian Frezza and the rest of the team at Emerald Cloud Labs for providing access to their cloud laboratory, and without whom the content of Chapter 3 would be vastly different. I'd like to thank Alex Wixtrom for all the one-on-one meetings helping me to navigate the *Command Center* and generally use sophisticated and expensive laboratory equipment that I was most definitely not qualified to use otherwise. I'd probably have (remotely) destroyed the lab without your guidance.

The research in Chapters 3 and 4 was supported by NIH T32 training grant T32 EB009403 as part of the HHMI-NIBIB Interfaces Initiative. The research in Chapter 4 was further supported by a CMU School of Computer Science Cancer Research Fellowship. I would like to thank AMD for the donation of critical hardware and support resources from its HPC Fund that made the work in Chapters 4 and 5 possible.

Once again, I would like to thank both my parents for their continuous support along the way. I would never have gotten this far without you, and I consider myself lucky to know that you will continue to be there for me for the rest of my scientific career.

Abstract

Many scientific processes depend upon sequential decision making. Choosing which experiments to run next, or how to alter an experimental design, or reconfigure experimental instrumentation affects not just the underlying accuracy or quality of the actual experiment, but also the efficiency at which optimal experimental conditions are identified. Especially as the ability to automate certain experimental components becomes more prevalent, practical algorithms that can guide these types of experimental decision making are more important now than ever. In this dissertation, we use machine learning to address such sequential decision making problems in two emerging biological domains— general laboratory experimentation via a Cloud Lab and protein engineering. Towards the first setting, we introduce PROTOCOL, a first-of-its-kind deterministic algorithm that improves experimental protocols via asynchronous, parallel Bayesian optimization. In the latter setting, we describe two methods for selecting protein engineering experiments. First, we show how to formulate Directed Evolution as a regularized Bayesian optimization problem where the regularization term reflects evolutionary or structure-based constraints. Finally, we demonstrate how to use a deep Transformer Protein Language Model to effectively select lead sequences from nanobody repertoires, as well as how to select beneficial single-site mutagenesis experiments that optimize targeted protein functions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automation's impact on the world continues to increase with improving technology. There are many benefits of automation that extend to applications of biological experimentation as well the scientific community at large. The exact role of automation takes one of two forms. The first and perhaps more commonly appreciated form concerns the machinery and systems in place that handle repetitive tasks. For instance, there exist specialized robots that are capable of performing numerous exact pipetting measurements [1], basic biological assays [2], and routine cell biology experiments [3]. Such systems are useful because they reduce cost, increase reproducibility, and free scientists to focus their attention on the broader conceptual problems they are trying to solve [4, 5]. However, these systems are also static. That is, they require a stable environment and are unable to adapt to new or unexpected circumstances. They are not able to learn from the complex systems they involve but rather carry out specific tasks related to the system. This form of automation is not our principal focus.

The other form of automation concerns interacting with a dynamic environment and learning how to make optimal choices within this environment. One of the first examples of this type of automation related to biological experimentation includes the Adam robot scientist, a robot capable of running genomic experiments, formulating hypotheses based on these experiments, then choosing the next experiment to run based on these hypotheses [6]. The abilities to model the hypothesis space from current known data, as well as choose subsequent experiments based on the current hypothesis space exemplify this dynamic form of automation. Automating these aspects of the scientific process yield not only the same benefits as their static counterparts, but also provide an efficient means to learn about and understand the system of focus. These systems can range from cancer genomics [7, 8, 9], to precision medicine [10], to drug discovery [11, 12] and more. Developing algorithms that are able to carry out these dynamic capabilities is thus an important area of focus.

In this dissertation, we develop algorithms of this form in order to make experimental decisions within two important and emerging scientific domains:

1. General experimental biochemistry via a Cloud Laboratory (Chapter 3)
2. Protein design and engineering (Chapters 4 & 5)

In the rest of this chapter, we will briefly introduce these two areas and motivate the basic research objectives we pursue within this dissertation.

1.1 Application areas

1.1.1 Experimental automation in the Cloud Lab

Recently, Carnegie Mellon University and Emerald Cloud Labs partnered together to form the world's first academic Cloud Lab [13, 14]. This union not only illustrates the promise of using automated technologies in research settings, but also how the era of automated science within academia is in its relative infancy. What a Cloud Lab offers is an ability to remotely specify complete laboratory procedures that are carried out by a combination of trained technicians and robotic instruments at an external, secure, fully-stocked facility [15, 16]. Just as cloud computing is intended to provide on-demand access to computing resources, a Cloud Lab provides the on-demand access to sophisticated laboratory equipment. Proponents of the Cloud Lab paradigm have found that it increases reproducibility of experimental work, improves laboratory efficiency, and leads to faster translation from purely research driven work to real-world applications [17]. In combination, these can have the effect of making experimental science a more efficient process, which in turn can speed up the rate of real scientific discovery.

However, there are some limitations that must be overcome before the Cloud Lab can truly democratize science. Some of these limitations are due to differences between research performed in academia versus the types of experiments carried out in industry, where use of automated science is more prevalent. For example, many research experiments rely on highly specialized equipment or experimental procedures that may not be amenable to the types of generalizable automated capabilities offered by current Cloud Labs. Still, other Cloud Lab issues mirror those that any researcher may encounter in their own lab. This includes configuring experimental settings to find those that are best capable of collecting data or producing interpretable results.

In Chapter 3, we propose a Bayesian optimization algorithm to address issues of the second type above. By using the unique capabilities of the Cloud Lab to solve a practical issue that many researchers will face when carrying out research objectives on the Cloud Lab themselves, we hope that we are contributing to the positive change automated science is beginning to bestow upon academia. We believe this work marks the first instance of academic closed-loop science using a Cloud Lab.

1.1.2 Protein engineering

Proteins are an essential component of all living systems. A given protein can be defined by its constituent sequence of amino acids. The cumulative biophysical properties of these amino acids cause a protein to uniquely fold into structures that allow it to carry out necessary and specialized functions. These include acting as enzymes that catalyze vital biochemical reactions, enacting immune responses, facilitating cell signaling pathways, and innumerably many more.

In many settings, it would be desirable to either enhance a protein's ability to carry out its prescribed function, or to synthesize new protein sequences that possess functionalities distinct from those observed by naturally occurring proteins. Examples could include proteins with a specified therapeutic property that target cancer cells, or proteins capable of breaking down certain materials for industrial applications. Objectives like these are addressed by protein engineering [18].

Protein engineers design novel protein sequences that possess some targeted property. One way to describe protein design is as a search over the space of protein sequences constrained by practical considerations such as limited resources like money, time, and technological feasibility. Traditional approaches like directed evolution conduct this search in a purely stochastic and iterative manner. The best random mutants obtained in one generation (i.e. those with highest fitness) serve as seeds for the next, and this continues until a specified stopping criterion is met. While this approach can lead to effective protein designs, it comes at the cost of efficiency— many costly iterations of experimentation are typically required.

This has motivated the creation of approaches that more intelligently navigate the space of protein sequences as a means to more efficiently identify protein sequences with desired properties [19, 20]. This includes those that are strictly computational, such as those based on molecular dynamics simulation and/or machine learning. While many of these approaches show great promise at modeling relationships between amino acid sequence and structure and function, they can also be further detached from the experimental realities and capabilities of most protein engineering laboratories. It is thus important to develop methodologies that are capable of identifying novel protein sequences with desired properties in ways that are consistent with typical protein engineering workflows.

To this end, we describe our machine-learning enhanced version of directed evolution in Chapter 4. We show how to use generative models of protein sequences as well as force-field based modeling to obtain regularization factors that we use within the context of a general-purpose Bayesian optimization protein design routine. In Chapter 5, we show how to use a deep Transformer Protein Language Model to identify novel sequences that have the most *promise*— we use the model's self-attention map to calculate a PROMISE SCORE that weights the relative importance of a given sequence according to predicted interactions with a specified binding partner.

1.2 Summary of contributions

- Chapter 3 introduces the algorithm PROTOCOL (**P**a**R**allel **O**ptimiza**T**i**O**n for **C**l**O**ud **L**aboratories), a bound-based Bayesian optimization approach to configuring experimental parameterizations in a Cloud-lab setting.
  - PROTOCOL is the first demonstration of Bayesian optimization in the Cloud Lab.
  - We apply PROTOCOL in a *simulated* cloud-lab to select four parameter settings that configure two sets of MALDI-ToF MS experiments (Sections 3.4 & 3.4.1).
    - PROTOCOL identified the optimal parameter settings in 9 out of 10 objectives within a 25 experiment budget.
    - Compared to four alternative Bayesian optimization approaches and random sampling, PROTOCOL found the optimal configurations with higher frequency (Tables 3.1 & 3.2) and found its best configuration more quickly (Figure 3.3).
  - We apply PROTOCOL in a *real* cloud lab to select six parameter settings that configure HPLC experiments (Sections 3.6 & 3.6.1).
    - PROTOCOL initially requests experiments that yield the incorrect number of peaks, but eventually requests those the identify a correct number (Figure 3.8).
    - Given a small budget of 18 total experiments, PROTOCOL does not select HPLC parameterizations that yield higher resolutions than those selected by Latin Hypercube sampling (Table 3.3).
- Chapter 4 describes a machine learning-enhanced Directed Protein Evolution approach to protein design via Bayesian optimization with evolutionary and structure-based regularization.

  - We introduce evolutionary-based regularization as a means to bias variant selections towards those that are native-like. We show how to obtain these terms from each of a Markov random field, hidden Markov model, and a transformer protein language model (Section 4.3.1).
  - We introduce structure-based regularization as a means to bias variant selections towards those expected to have high stability. We show how to obtain this term using the FoldX empirical force field (Section 4.3.2).
  - We incorporate these regularization terms into a Bayesian optimization framework (Section 4.4.1). We use this framework for *in silico* protein design using proteins GB1, BRCA1, and Spike (Section 4.5).
    * Overall, we find that ML-assisted DE improves upon the traditional single mutation walk and recombination approaches in designing high fitness GB1 variants by an average of 45% (Figure 4.3).
    * Given a sufficient number of experimental rounds and a complete picture of the system being optimized, structure-based regularization identifies variants with highest fitness (Figure 4.4).
    * Both evolution and structure-based regularization promotes site-specific exploration of unexplored sequence space (Section 4.5.5).

- Chapter 5 describes how the transformer protein language model ESM-1b can be used to identify protein sequences with targeted properties, and thus serves as a powerful tool for different protein engineering objectives.
  - We introduce a joint encoding scheme for a protein sequence and its binding partner as input for the ESM-1b transformer model (Section 5.2.1).
  - We introduce the PROMISE SCORE as a measure of a sequence's likeliness to possess a targeted property (Section 5.2.2). The PROMISE SCORE is calculated

using ESM-1b's attention mechanism, and accounts for intramocular and/or intermolecular interactions between the encoded sequences.

- We use the PROMISE SCORE for *in silico* selection of lead sequences from two nanobody repertoires. We also use the PROMISE SCORE to select single-site mutagenesis experiments within the BRCA1-BARD1 and Spike-ACE2 protein complexes to optimize for downstream binding-dependent activity and binding affinity, respectively.
  - Across most encoding schemes, we find that sequences with high PROMISE SCORE tend to be "strong" sequences (Figure 5.2).
  - An encoding scheme with no linker or short Alanine linker yields experimental selections that are most enriched with strong sequences (Figures 5.3 & 5.4).
- We introduce a per-residue PROMISE SCORE, and find that it provides insights into protein-target interactions (Section 5.3.3).
  - Across both nanobody repertoires, 3 out of 8 validated epitopes are enriched with high scoring residues. With BRCA1-BARD1, residues in the known binding sites of each protein are enriched with high scoring residues. With Spike-ACE2, residues in the known binding site of ACE2 are enriched with high scoring residues (Figure B.8).
- We demonstrate that fine-tuning ESM-1b improves model AUC when training models that predict nanobody binding strength (Figure 5.7).

1.3 Summary of publications

The content of Chapter 3 appears in:

> [21] Trevor S. Frisby, Zhiyun Gong, and Christopher James Langmead. "Asynchronous parallel Bayesian optimization for AI-driven cloud laboratories". In: *Bioinformatics* 37.Suppl_1 (July 2021), pp. i451–i459

The content of Chapter 4 appears in:

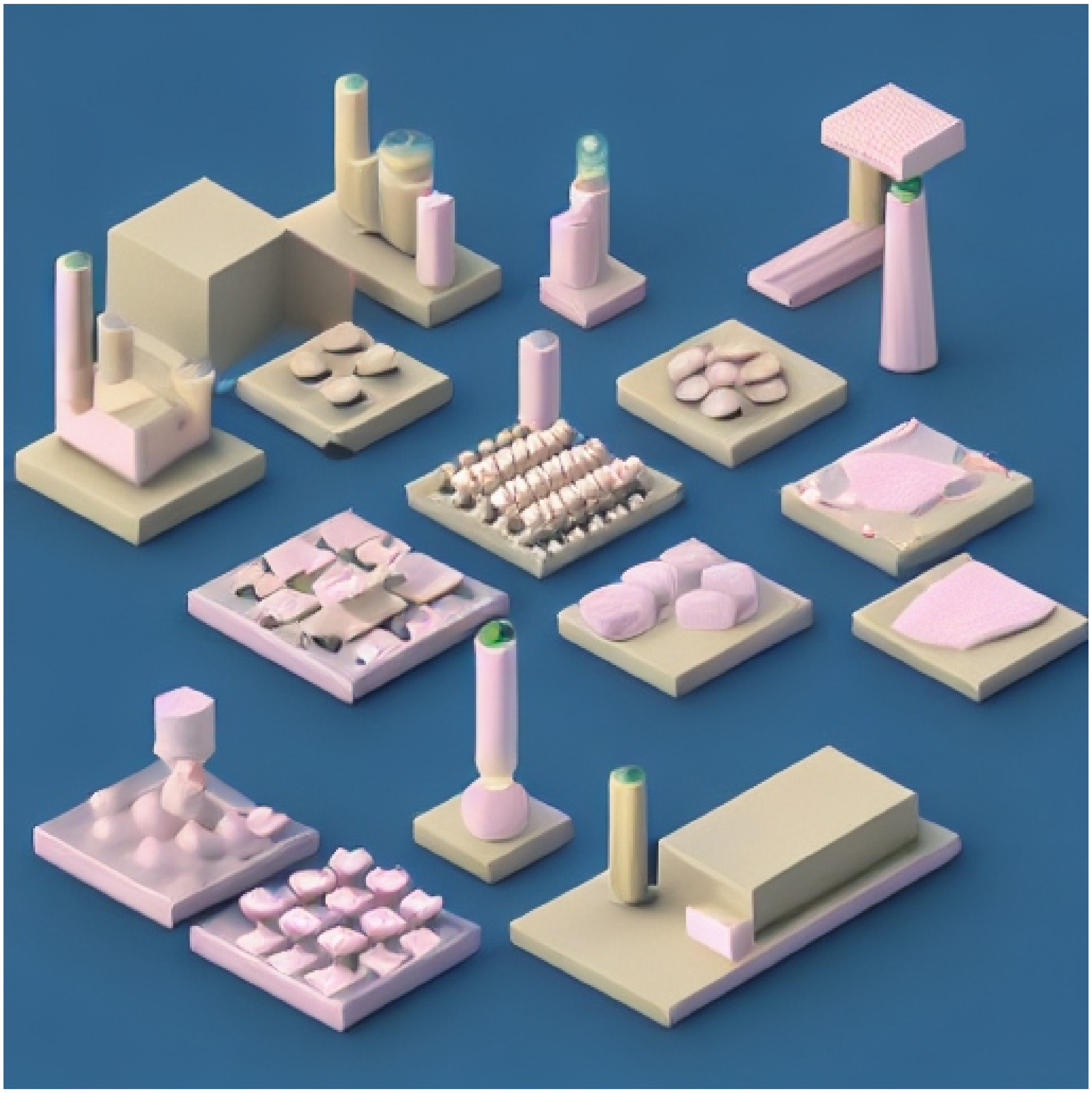
> [22] Trevor S. Frisby and Christopher James Langmead. "Bayesian optimization with evolutionary and structure-based regularization for directed protein evolution". In: *Algorithms for Molecular Biology* 16.1 (July 2021), p. 13
>
> [23] Trevor S. Frisby and Christopher James Langmead. "Fold Family-Regularized Bayesian Optimization for Directed Protein Evolution". In: *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Vol. 172. Leibniz International Proceedings in Informatics (LIPIcs). 2020, 18:1–18:17

The content of Chapter 5 appears in:

> [24] Trevor S. Frisby and Christopher James Langmead. "Identifying promising sequences for protein engineering using a deep Transformer Protein Language Model". In: *bioRxiv* (2023)

Non-dissertation research: In addition to the above body of work, I also pursued the following research direction during my PhD. Its content is not included in this dissertation:

> [25] Trevor S. Frisby et al. “Harvestman: a framework for hierarchical feature learning and selection from whole genome sequencing data”. In: *BMC Bioinformatics* 22.1 (Apr. 2021)



Chapter 2

Background

This chapter will introduce the basic tools and methodologies that this dissertation builds upon. We will expand on these preliminaries in the subsequent chapters when necessary, or restate for clarity. After all, *repetitio mater studiorum est*.

2.1 Sequential model-based optimization

Algorithms used for real-life research decision making have to learn quickly from an often limited data supply. Within machine learning, algorithms of this form fit broadly within the subfield of active learning. Whereas passive supervised learning strategies focus on learning a model from set-in-place labeled instances [26, 27], active learning focuses instead on identifying which unlabeled instances to obtain a label for according to a data access model. This access model describes how the algorithm chooses unlabeled instances to query [28].

Running actual experiments often presents a large or even prohibitive burden in terms of financial and time costs. This is because most experiments involve a large number of trials which can consume a large number of resources. A primary goal of this dissertation is to introduce methods that reduce these burdens by limiting the number of necessary trials to find the best solution. While in practice there may exist overly simplistic heuristics to help combat this problem [29], sequential model-based optimization provides a means to formalize and solve the problem using machine learning.

Sequential model-based optimization routines use a surrogate function, S, to approximate a hard to compute ground-truth function, f. In a typical biology research setting, f corresponds to a costly experiment or sets of experiments. Optimizing over S obtains the expected best experimental design for f according to the current available data. By updating a model of the system M with newly acquired data $(x_+, f(x_+))$, we sequentially learn a model of the system while identifying informative instances. In effect, we efficiently navigate the experimental design space to quickly obtain one that is "best", where "best" can be quantified by a number of metrics, including expected improvement and probability of improvement [30].

Sequential model-based optimization has previously been applied to hyperparameter optimization problems. Hutter, Hoos, and Leyton-Brown [31] demonstrate the Random Online Aggressive Racing (ROAR) algorithm and its extension Sequential Model-

Based Algorithm Configuration (SMAC) as a means to automatically configure parameters involved in SAT solving and mixed-integer programming problems. Bergstra et al. [32] use sequential model-based optimization to optimize hyperparameters of Deep Belief Networks, and introduce the tree-structured Parzen estimator approach. This strategy uses kernel density estimates over parameter configurations to iteratively select configurations that maximize expected improvement. Sequential model-based optimization can similarly be applied to sequentially optimize experimental protocol configuration. Rubens et al. [33] use a gradient descent like algorithm to optimize over flow reactor parameters for automatic polymer synthesis.

2.1.1 Bayesian optimization with Gaussian processes

In this dissertation, we perform sequential model-based optimization by means of Bayesian optimization [34, 35]. Bayesian optimization is a sequential strategy for optimizing black-box objective functions that has been used in a variety of contexts, including robotics [36], particle physics [37], and hyper-parameter optimization in deep learning [32].

Bayesian optimization is used to maximize an unknown function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subset \mathbb{R}^d$:

$$\max_{x \in \mathcal{X}} f(x) \tag{2.1}$$

As f is unknown, the Bayesian approach is to treat it as a random function and place a prior over it, $P(f)$. This prior captures the belief about the behaviors of this function, and a typical choice is to use a Gaussian process prior. A Gaussian process [38] is a distribution over functions of the form $GP : \mathcal{X} \to \mathbb{R}$ such that for finite N, any set $\{g(x_n), g \sim GP, x_n \in \mathcal{X}\}_{n=1}^{N}$ induces a multivariate Gaussian distribution on $\mathbb{R}^N$. In other words, a Gaussian process is the infinite-dimensional generalization of the multivariate Gaussian distribution. As a multivariate Gaussian is defined by its mean $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^{k \times k}$, a Gaussian process is defined by a mean *function* $\mu : \mathcal{X} \to \mathbb{R}$

and *kernel function* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that quantifies the joint variability between any pair $(x_i, x_j) \in \mathcal{X} \times \mathcal{X}$. Common choices for K includes the radial basis function (RBF) and Matérn kernels.

The relationship between a Gaussian process and the Gaussian distribution allows us to compute useful quantities in closed form. Suppose we have observations

$$y = [f(x_1), f(x_2), \ldots, f(x_N)]^T$$

and want to identify the marginal distribution of $f(x_+)$ for new test case $x_+ \in \mathcal{X}$. Where we assume $f \sim GP(\mu, K)$, let

$$k_+ = [K(x_1, x_+), K(x_2, x_+), \ldots, K(x_N, x_+)]^T$$

as well as let

$$\Sigma = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & \cdots & K(x_N, x_N) \end{bmatrix}.$$

We then have

$$P(f(x_+) \mid y) \sim \mathcal{N}(k_+^T \Sigma^{-1} y, K(x_+, x_+) - k_+^T \Sigma^{-1} k_+)$$

where $\mathcal{N}(\mu, \sigma^2)$ refers to a normal distribution with mean μ and variance σ^2. Together, these mean and covariance functions can be used to compute posterior probabilities over function values, and related quantities, such as upper and lower confidence values.

These estimates can be used by an *acquisition function* $a : \mathcal{X} \to \mathbb{R}$ in order to identify untested points that will provide information relevant to finding the optimal value of f. The acquisition function specifies a method that balances exploration of feature space $\mathcal{X}$ and exploitation of the current GP estimate for unknown function f. Perhaps the most direct approach is to use the Upper Confidence Bound acquisition function, which has two terms that correspond to each of these components:

$$UCB(x_+) = \mu(x_+) + \beta\sigma^2(x_+). \tag{2.2}$$

The mean function μ accounts for exploitation of the current GP estimate. Exploration is accounted for by the model's uncertainty, given by variance function σ^2, where β is a parameter that controls the degree of tradeoff between exploration and exploitation.

Other acquisition functions account for *improvement*— they try to select x_+ that improves upon the current best observation, y_{best} [39]. The net improvement of x_+ is given by:

$$I(x_+) = \max(f(x_+) - y_{best}, 0)$$

One strategy is to choose x_+ that has the highest probability for improvement. This is the probability of improvement (PI) acquisition function, and it is given by:

$$\begin{aligned} PI(x_+) &= P(I(x_+) > 0) \\ &= P(f(x_+) > y_{best}) \\ &= \psi\left(\frac{\mu(x_+) - y_{best} - \xi}{\sigma^2(x_+)}\right) \end{aligned} \tag{2.3}$$

where $\psi(\cdot)$ is the cumulative distribution function for the standard Gaussian, and ξ is a parameter that controls the degree of exploration.

Notably, PI does not account for the magnitude of improvement, just the probability that there is any improvement. A method that does account for the magnitude is the Expected Improvement (EI) acuisition function:

$$\begin{aligned} EI(x_+) &= \mathbb{E}[I(x_+)] \\ &= \int_{-\infty}^{\infty} I(x_+)\phi(z)\mathrm{d}z \end{aligned}$$

where $\phi(z)$ is the probability density function of the standard Gaussian. This integral can be solved exactly, and simplifies into the following EI acquisition function expression:

$$EI(x_+) = \delta(x_+)\psi\left(\frac{\delta(x_+)}{\sigma^2(x_+)}\right) + \sigma^2(x_+)\phi\left(\frac{\delta(x_+)}{\sigma^2(x_+)}\right) \tag{2.4}$$

where $\delta(x_+) = \mu(x_+) - y_{best} - \xi$. For some intuition behind this expression, notice that EI increases as does the difference $\mu(x_+) - y_{best}$ (i.e. belief under the GP that $f(x_+)$

improves upon y_{best}) and $\sigma^2(x_+)$ (i.e. uncertainty, meaning that observing these untested samples would contribute to exploration).

Regardless of the choice for acquisition function, the iterative, repeating process of finding $x_+ = \arg\max_x a(x)$, evaluating $y_+ = f(x_+)$, and updating the Gaussian process with observation (x_+, y_+), signifies the sequential aspect of Bayesian optimization.

2.2 Generative modeling

Much of the work presented in this dissertation relies upon *generative modeling*. Broadly speaking, machine learning models can be categorized as either generative or discriminative, where the distinction is given by the probability distribution the model describes. Given independent variable X and target variable Y, a discriminative model is one that directly models the conditional probability $P(Y \mid X = x)$. It is a parameterized model that learns to associate a label $y \in Y$ to observation $x \in X$ by optimizing the model's parameters to make accurate predictions on known data. Well known discriminative models includes logistic regression, support vector machines, and random forests.

This contrasts with a generative model, which instead learns the joint probability distribution $P(X, Y)$. It is a statistical model that describes the distribution of the data itself, typically obtained via maximum likelihood estimation. By applying Bayes' theorem, it is possible to recover the conditional probability distribution $P(Y \mid X)$, meaning that generative models can still be used for discriminative tasks. Generally speaking, given sufficient training data, it is expected that a discriminative model will be more accurate than a generative one. However, it has been shown that in settings with limited training data, generative models actually tend to perform better than discriminative ones [40].

The use of generative modeling in this dissertation is motivated by their ability to work well in low-data settings as well as their ability to model long-range dependencies in biological sequencing data. The types of generative model used in this dissertation includes the Gaussian process, Hidden Markov model, Markov Random Field, and deep

Transformer. In Section 6.2, we will provide further detail on other capabilities of generative models within the context of future directions.

2.2.1 Hidden Markov models and Markov random fields

Importantly, generative models are effective at describing complex dependencies and patterns that exist within biological data. As an example, models that capture the evolutionary relationships between a set of amino acid sequences from a particular protein family may be used to ascertain whether or not an unidentified sequence is likely to belong to the same protein family. Two traditional classes of generative model that have been applied to problems like this are the Hidden Markov Model (HMM) [41] and the Markov Random Field (MRF) [42], both of which encode a joint distribution over residue types at each position in a primary sequence. They are both probabilistic graphical models, meaning they represent conditional dependencies between random variables as edges connecting nodes on a graph. An HMM is a *directed* graph, where a sequence of observed variables Y_n are explained through directed connections from a set of hidden states X_n. An HMM makes two main assumptions of conditional independence between variables based on the Markov property:

1. The probability of a particular state $X_i = x_i$ depends only on the previous state $X_{i-1} = x_{i-1}$:
$$P(x_i \mid x_0, \ldots, x_{i-1}) = P(x_i \mid x_{i-1})$$

2. Observation $Y_i = y_i$ depends only on the state $X_i = x_i$ that produced the observation:
$$P(y_i \mid x_0, \ldots, x_i, \ldots, x_N, y_0, \ldots, y_i, \ldots, y_N) = P(y_i \mid x_i)$$

With respect to protein sequences, these are a fairly strict set of assumptions. A consequence is that an HMM will not account for long-range dependencies within a protein sequence.

An MRF is an *undirected* graphical model. Dependencies between variables are represented by an undirected edge between two adjacent nodes. Conditional independence between variables is given by three Markov properties:

1. Any non-connected variables are conditionally independent given all others.
2. A variable is conditionally independent to all others given its connected neighbors.
3. Subsets of variables are conditionally independent given a separating subset (i.e. a connected set of nodes that completely separate the two subsets).

The MRF thus is able to capture long-range dependencies, though comes at the cost of greater model complexity relative to an HMM. The GREMLIN algorithm [42] specifies a method to learn both the structure and parameters for an MRF model over amino acid sequences within a multiple sequence alignment in a tractable manner.

2.2.2 Deep generative models

Most recent progress in generative modeling focuses on deep generative models. These models use deep neural networks to learn complex probability distributions and representations that encode the different long-range dependencies found within data, including biological sequences. Many different varieties of model type and architecture have been developed, including variational autoencoders [43], generative adversarial networks [44], and transformers [45]. This dissertation focuses on the transformer architecture.

As originally described by Vaswani et al. [45], the transformer uses an encoder-decoder structure[1]. The encoder takes an input sequence and produces a *representation* for the sequence— this is a real-valued vector that encodes the patterns and dependencies that exist within the input sequence. The decoder takes the encoder's output,

[1]In some use cases, the transformer may consist of only the encoder or decoder

and uses it to create an output that fits some intended domain, such as text completion, neural translation, or image generation. Both the encoder and decoder consist of sequentially stacked transformer blocks, which themselves consist of a self-attention mechanism followed by a feed-forward neural network.

Self-attention is the key component that allows the model to account for long-range relationships in the input sequence. This mechanism calculates weights that denote the relative importance of each element of the input sequence. These weights are computed dynamically, and account for all pairwise interactions in one fell swoop via straightforward matrix operations. This contrasts with recurrent and convolutional models, which require many steps to represent long-range dependencies. Passing the output of the self-attention mechanism to a feed-forward neural network ensures that the representation is able to capture non-linear relationships as well.

Chapter 3

Automating Experimental Protocols with the Cloud Lab

As described in the Chapter 1 introduction, automation can include processes that are static or dynamic. In this chapter, we describe a sequential algorithm that encompasses both aspects. This algorithm, PROTOCOL, is intended to dynamically configure experimental parameters by taking advantage of the programmable nature of static laboratory equipment within a Cloud Laboratory.

3.1 The Cloud Laboratory

Most standard laboratory techniques have been automated, which in turn has enabled the development of commercial robotic *Cloud Laboratories* [1], and the emergence of a new paradigm of *Science-as-a-Service*. Analogous to cloud computing, cloud labs let scientists outsource the management and maintenance of a set of resources— automated scientific instruments, and thus devote more time and money to their research. In addition to these administrative and economic benefits, cloud labs also significantly increase the reproducibility of scientific research, due to the use of robotics. For these reasons, one can anticipate an increase in the utilization of cloud labs by scientists in academia and industry alike, at least for certain tasks.

Existing cloud labs are largely open-loop, in the sense that humans must specify every detail of the experimental protocols to be executed by the robots. However, it is not difficult to imagine an AI-driven cloud lab that automatically finds optimal instrument settings and/or experimental conditions, so as to maximize throughput and data quality, or to minimize costs. Eventually, such systems might lead to the widespread use of general-purpose "robot scientists" capable of making novel discoveries autonomously, as first demonstrated in 2004 [6]. Towards these ends, this chapter introduces a method, called PROTOCOL (**P**a**R**allel **O**ptimiza**T**i**O**n for **C**l**O**ud **L**aboratories), to perform closed-loop optimization of experimental protocols against a user-defined objective.

PROTOCOL builds on recent work in Bayesian Optimization (BO) [34, 35], which is a sequential strategy for optimizing black-box (i.e. unknown) functions. The technique is Bayesian because it places a prior distribution over the objective function, and then computes posteriors at the end of each round, based on the outcome of an algorithmically-selected function evaluation. The key differences between BO methods are the means by which they represent the distribution over functions, and the way that they select the next design configuration to test. Gaussian Processes are a very common choice

[1] E.g. Emerald Cloud Lab [46] and Strateos [47]

for specifying the distribution, and that is what is used in this chapter. The selection strategy, sometimes called the *acquisition function*, will define a utility function and then searches for a design with (approximately) maximal utility. PROTOCOL introduces a novel acquisition strategy that is matched to the features of a cloud lab environment.

Bayesian Optimization is often used in application domains where the evaluation of the objective function is extremely slow or expensive, such as hyperparameter optimization in deep learning (e.g. Bergstra et al. [32]). Performing wet-lab experiments is also time-consuming, even under the best of circumstances. But cloud labs comprise a set of shared resources, and so experiments often sit in a queue waiting for specific instruments to become available. That is, the benefits of outsourcing the management and maintenance of a wet-lab to the cloud are somewhat offset by increased cycle times, on a per-experiment basis. On the other hand, a suitably equipped cloud lab may facilitate *parallel searches* for optimal conditions. PROTOCOL's acquisition function takes advantage of such parallelism by selecting *batches* of designs to test, while performing closed-loop, asynchronous Bayesian Optimization.

We evaluated PROTOCOL on two test scenarios. The first optimized four instrument parameters for MALDI-TOF mass spectrometry in a simulated cloud lab (but using real data). The second optimized five instrument parameters *and* the solvent ratio for HPLC in a real cloud lab. PROTOCOL outperforms conventional BO methods dramatically on the MALDI-ToF data, given the same budget. On the real cloud lab, PROTOCOL makes progress toward finding a high-resolution chromatogram.

3.2 Background and related work

At the heart of most biological discoveries are a set of precise and highly-specialized laboratory-based methods used to conduct experiments from which meaningful conclusions are drawn. In many settings, especially experimental chemistry and biochemistry, the goal of configuring an experimental design is to identify a set of independent variables that maximize (or minimize) a chosen dependent variable under certain experimental conditions. In essence, this is a "tuning" or "calibration" stage, where an experimental setup is honed in order to obtain usable data, making this a critical and necessary component of many experiments. Let $\Theta = \{\theta_1, \cdots, \theta_d\}$ be the length d set of independent variables where θ_i corresponds to the parameterization of independent variable i and let $f : \Theta \to \mathbb{R}$ be a function mapping the independent variable configuration space to corresponding dependent variable values. Where $Y^* \in \mathbb{R}$ corresponds to optimal dependent variable values, we can state this goal succinctly as

$$Y^* = \underset{\Theta}{\operatorname{argmax}}\, f(\Theta) \tag{3.1}$$

In domains with plentiful data, this problem is easily solvable by learning the function f with standard regression and supervised machine learning. However, identifying an optimal Θ^* is typically a first step taken towards finding Y^*, meaning prior data to learn from is scarce or absent entirely. Drawing from past experiments that used similar setups or techniques is also not straightforward, as changes to samples or instrumentation used may lead to entirely different behavior over the measured variables. Grid searches or naïve one-by-one tuning of parameters are common heuristic solutions to this problem, but they are inefficient, and can lead to large numbers of experiments that do not contribute towards the goal of finding Y^*. Since there is often great cost associated with generating such data in terms of time, money, and resources, we want to find Θ^* in as few experimental steps as possible.

3.2.1 Bound-based Bayesian optimization

Problems like those of Equation 3.1 can be solved by using Bayesian optimization with Gaussian processes (see Section 2.1.1). This requires a specified acquisition function. A common feature of these methods— and a potential weakness, is that they require access to a finite sampling procedure which affects both the runtime and the ultimate resolution the optimization procedure— the finer the resolution, the more computationally expensive the optimization. Recently, however, an algorithm that does not require sampling during BO was introduced. That algorithm, called IMGPO (Infinite Metric Gaussian Process Optimization) [48], performs serial BO and comes with convergence guarantees. Our method extends IMGPO to the asynchronous parallel setting.

IMGPO builds upon previous work in bound-based optimization methods [49, 50], and uses a divide-and-search strategy based on estimated bounds, like DIRECT [51]. It proceeds by growing a hierarchical partitioning tree over an n-dimensional search space while maintaining a GP model conditioned on observations previously requested by the algorithm. The tree is grown by iterating over it in a top-down fashion and choosing whether or not to evaluate the center of intervals/hyperrectangles associated with each node in the tree. Selected intervals may be further divided into three subintervals along the hyperrectangle's longest dimension, resulting in three new leaf nodes in the tree.

The decision to evaluate and divide is made by comparing multiple bounds on the unknown ground-truth value of a given interval's center. These bounds are defined by the upper confidence bounds (UCB) of the GP model[2], as well as the ground-truth values of observed interval centers. In general, when the UCB of the current iterate is greater than the current best observed value, the algorithm will request to evaluate the current interval's center, and divide the interval. Ultimately, the sequence of selected interval centers that are evaluated converge to the ground-truth function's optimal value. The IMGPO authors prove that their algorithm achieves exponential convergence over continuous search spaces with respect to simple regret, given by $R(x^+) = sup_{x \in \mathcal{X}} f(x) - f(x^+)$, where x^+ is the configuration found by the algorithm, without the need for sampling of the input space. Full technical details of the algorithm and proofs are provided by Kawaguchi, Kaelbling, and Lozano-Pérez [48] . We emphasize that the IMGPO algorithm performs serial optimization, in that it only requests the evaluation of one design at a time.

[2]Here, we assume that we are trying to maximize the objective function. If attempting to minimize a function, one uses lower confidence bounds (LCB).

3.3 Our method: *Protocol*

PROTOCOL adopts the hierarchical partitioning tree schema and interval division criteria employed by IMGPO. The primary innovation used by PROTOCOL is the calculation of a *frontier* from which up to k experiments can be chosen to run in parallel. Here, k is the maximum number of experiments a given cloud lab user is authorized to run at the same time. The frontier consists of the center points of the set of *potentially optimal hyperrectangles* in the n-dimensional search space (i.e. those that may contain the optimum of the objective function). The idea of maintaining a set of potentially optimal hyperrectangles is borrowed and adapted from the DIRECT algorithm [51] for derivative-free global, serial (non-Bayesian) optimization. One of the points on the frontier will always be the point that IMGPO would have selected (in the serial optimization setting), given access to the same set of observations of the objective function. Hence, PROTOCOL inherits the same guarantees as IMGPO, with respect to exponential convergence.

The remaining points on the frontier are identified by computing the convex hull over a two-dimensional encoding of the sub-volumes associated with all non-evaluated leaf nodes in the partition tree. The two coordinates for each sub-volume are the corresponding node's depth in the tree (which is inversely proportional to the size of the sub-volume), and the UCB of the objective function within that region. The intuition behind maintaining a frontier based on sub-volumes of different sizes is that those volumes represent different trade-offs between *exploration* of the input space— to gather information from under-sampled regions (i.e. those corresponding to relatively large volumes), and *exploitation*— to search in the vicinity of the best design observed thus far (i.e. those corresponding to relatively small volumes). Every BO acquisition function makes a trade-off between exploration and exploitation; PROTOCOL's strategy is to select batches of experiments that individually make different trade-offs. The use of the UCB is justified based on the well-established principle of optimism under uncertainty [52]. Using the convex hull ensures the algorithm avoids requesting experiments the GP model believes to be suboptimal, while considering nodes at each depth promotes choosing intervals representing varied portions of the input space.

The selection strategy used by PROTOCOL can be described as a three step process:

1. **Identify intervals eligible for division**. This step follows an anologous one in IMGPO, where the algorithm traverses the tree and attempts to identify one interval *at each depth* of the tree that is eligible to be divided. If a function evaluation is necessary, PROTOCOL will calculate a frontier, and multiple experiments may be requested accordingly.

2. **Prune the chosen intervals**. This step also follows directly from IMGPO. Intervals selected in Step 1 are added to a list if they contain a UCB greater than the value associated with any smaller interval in the tree. Essentially, this determines whether or not progress made in other portions of the tree suggests the algorithm should continue to divide in that area or elsewhere.

3. **Select and divide intervals**. This step differs from IMGPO. Here, any intervals that passed the initial two steps are divided into three subintervals by splitting along the longest dimension. Any newly created intervals whose center would be evaluated according to IMGPO are added to an experimental queue. From the remaining experiments, a frontier is calculated, and the queue is filled up to the maximum level of parallelization with experiments that lie on the frontier. In the event that there are more experiments on the frontier than space in the queue, those with highest UCB are chosen. In the event that there aren't enough intervals to fill the queue (i.e. the size of the frontier is $< k$), then the algorithm simply requests only the $< k$ identified intervals.

By construction, one of the experiments selected in Step 3 would have been chosen by the IMGPO algorithm (in the serial optimization setting) given access to the same set of observations. Thus, PROTOCOL has the same convergence guarantees as IMGPO.

In particular, both algorithms achieve exponential convergence: $R(x^+) \in O(\lambda^{N+N_{GP}})$, where $\lambda < 1$, N is proportional to the number of evaluations of the objective (here, the number experiments performed), and N_{GP} is the number of evaluations of a Gaussian Process model.

At the end of a given pass through the tree, the GP hyperparameters are updated, and the algorithm repeats until a prescribed maximum number of evaluations are made.

3.3.1 Descriptive example of algorithm

To illustrate how the algorithm works, we optimize the 1 dimension sinusoidal function $f(x) = \frac{1}{2}(\sin 13x \sin 27x + 1)$ defined over the unit interval (this function is visualized in Figure 3.1). This function has multiple local maxima over this domain, where the global maximizer is given by $x \approx 0.868$. We use a hierarchical tree $\mathcal{T}$ to maintain and visualize the division scheme over the input space. Each node corresponds to a subinterval obtained after division of its parent interval node. Each interval is either associated to the ground-truth function evaluated at the interval center, if it has been previously selected, or the UCB evaluated at the center according to the GP model otherwise.

Figure 3.2-top shows the hierarchical tree obtained by the algorithm at three different time points. In these figures, the horizontal axis refers to the input space for the function to be optimized, and the nodes have been fixed along this axis according to each interval's center coordinate. The ground-truth function optimizer is indicated by the star along the axis at $x \approx 0.868$.

The leftmost figure shows $\mathcal{T}_0$, the tree after initial iteration 0. The root node at depth 0 corresponds to the initial interval center located at the center of the input space ($x = 0.5$). After evaluating the function at this value, the interval is divided into three subintervals with centers $x = 1/6$, $x = 0.5$, and $x = 5/6$. These intervals correspond to the three nodes at depth 1 in the tree. Since the middle interval has the same center coordinate as its parent, it is associated with its ground-truth value, whereas the other

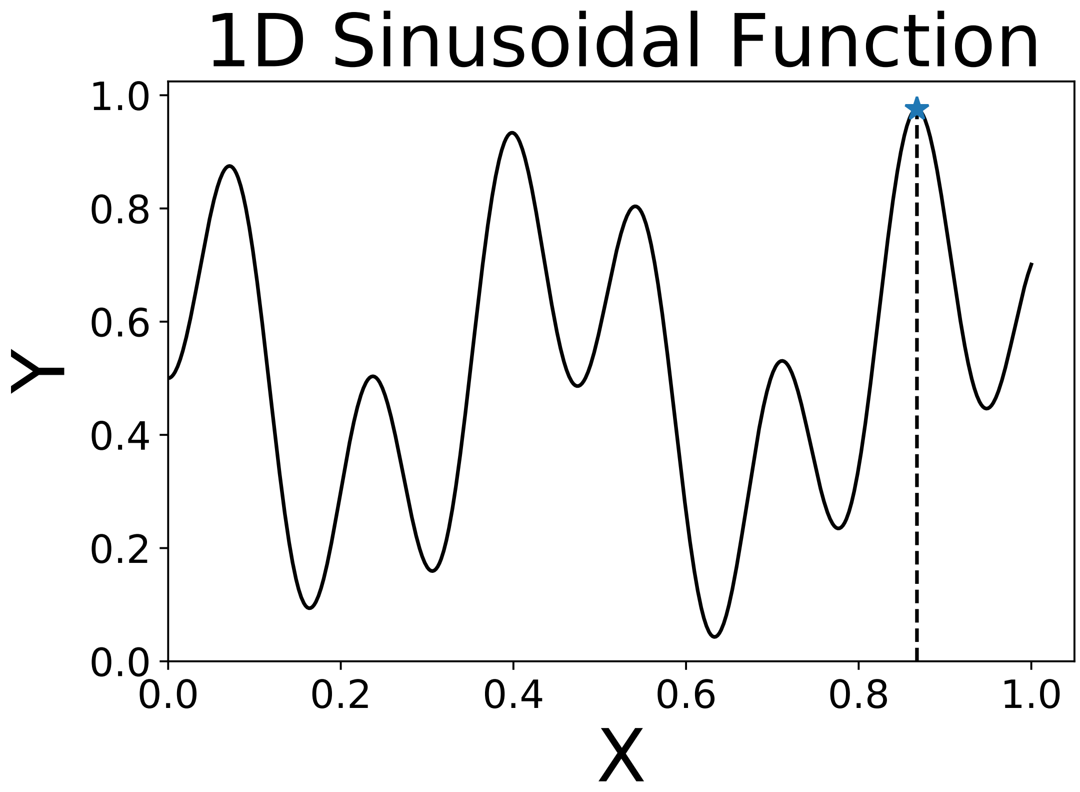


Figure 3.1: The function $y = \frac{1}{2}(\sin 13x \sin 27x + 1)$ used to demonstrate PROTOCOL. The function is maximized at $x \approx 0.868$, indicated by the blue star and dashed line.

nodes are associated with the UCB of a GP model conditioned on this observed data point. Proceeding through Step 1, the algorithm will iterate over each depth level in the tree and try to identify the best candidate node to divide while keeping track of the current best observed value, $\nu_{\max}$. At depth 0, there is only one node, and that node has already been divided, so no candidate is chosen at this level. By default, $\nu_{\max}$ will be set to the ground-truth value of this interval center. At depth 1, none of the nodes have been divided. The algorithm will identify the interval with the best associated value, and one of the following will occur:

(i) The selected interval has a ground-truth center value greater than or equal to $\nu_{\max}$. This interval is then added to the candidate list for dividing, and $\nu_{\max}$ is set to this center value.

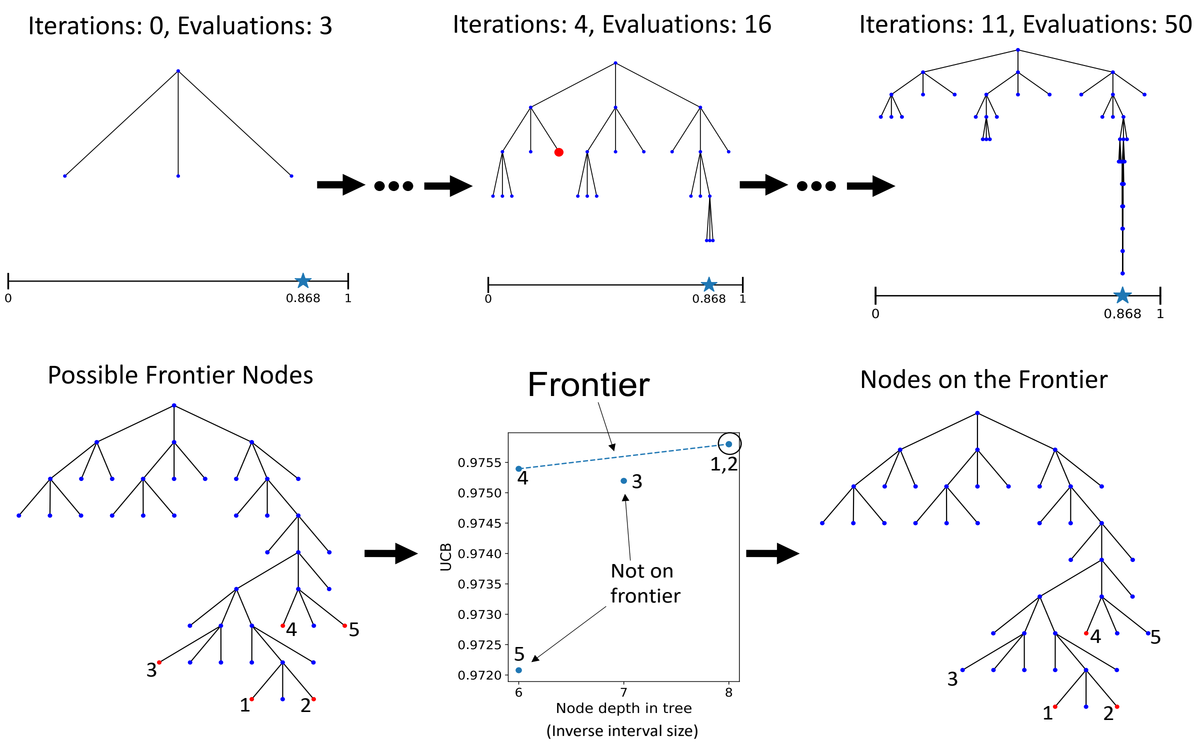


Figure 3.2: **Top row**. Shown are hierarchical trees produced by PROTOCOL at three different time points while optimizing a 1D sinusoidal function (see text for explanation). The nodes are fixed along the horizontal axis according the center coordinate of the interval they represent. The function optimizer, $x \approx 0.868$, is indicated by the star along the horizontal axis. **Bottom row**. A visualization of the frontier calculated by PROTOCOL in relation to the hierarchical tree. The enumerated red nodes on the left indicate intervals whose center coordinate are used to calculate the frontier. The central diagram shows the frontier, where intervals 1, 2, and 4 lie on the frontier but intervals 3 and 5 do not. Note that the depth of the tree is inversely proportional to the size of the interval. The red nodes on the right denote those intervals that lie on the frontier, and are those whose center coordinates will be requested for evaluation.

(ii) The selected interval has a ground-truth center value less than $\nu_{\max}$. In this case, no candidate is added to the list at this depth, and $\nu_{\max}$ is not updated.

(iii) The selected interval has a UCB-based center value, rather than ground-truth (i.e. it has not been previously evaluated). If this happens, a frontier will be calculated and up to k many experiments will be requested (at this early stage of the algorithm, there are only two possible intervals available for consideration, so the frontier would not be invoked). Once experiments are completed, the ground-truth observations are then associated with their corresponding interval, and the algorithm will again iterate over the depth 1 nodes until (i) or (ii) occur.

If the tree has greater depths, the algorithm moves on to the next depth in the tree, and proceeds until all depths have been visited.

The center of Figure 3.2-top shows the progression of the algorithm after four more iterations. To illustrate Step 2, suppose that the interval represented by the indicated red node at depth 3 has been selected by Step 1. This step will decide whether or not to keep this interval in the list of intervals to be divided. A tree $\mathcal{T}'$ rooted at the node given by this interval is grown to at most a pre-specified depth by using the same division scheme employed by the algorithm— dividing each interval into three smaller intervals by splitting along the longest dimension. The UCB of each interval center is calculated and associated with its node. The UCB values in $\mathcal{T}'$ are then compared to the center values of nodes at depth greater than 3 in $\mathcal{T}_4$. If $\mathcal{T}'$ contains a UCB greater than the center of *any* of these intervals from $\mathcal{T}_4$, then the interval is kept in the list. Otherwise, it is removed.

At the conclusion of Step 2, all intervals that remain selected are then divided. Upon division of a given interval, two of the newly created intervals will have unevaluated centers, while the middle interval will inherit the ground-truth value of its parent. The algorithm then decides whether or not to evaluate each of the two unevaluated centers by comparing the UCB evaluated at the center to $v_{\max}$. If the UCB is less than $v_{\max}$, then the UCB is used as the interval center. Otherwise, the ground-truth value must be obtained. This is where most calls to calculate the frontier occur.

Figure 3.2-bottom shows an example frontier. For visual clarity, the horizontal axis has not been directly fixed according to the interval coordinates. In the leftmost figure, nodes 1 and 2 are the two nodes created upon division, and nodes 3, 4, and 5 are the others that are considered for the frontier (i.e. they are the remaining leaf nodes whose centers have not been previously evaluated). The middle figure visualizes the frontier, where nodes 1, 2, and 4 are selected. Nodes 1 and 2 are selected because they were the two new nodes created by division (they happen to fall on the frontier, but would have been evaluated regardless). Node 3 is not selected because it does not fall on the upper convex hull, and node 5 is not selected because it is dominated by node 4, which has a greater UCB and is at the same depth in $\mathcal{T}$. The selected interval centers are then evaluated, and their ground-truth values are associated with each corresponding node.

The algorithm repeats until a prescribed number of evaluations are made. The rightmost figure in Figure 3.2-top shows the the final tree $\mathcal{T}_{11}$ obtained after 50 function evaluations. Notice how the width of the tree indicates that the algorithm was able to explore the initially unknown search space, while the depth of the tree is largely concentrated near the optimizer $x \approx 0.868$. This shows that the GP model was able to quickly guide the division towards this global optimal solution.

3.4 Simulated Cloud Lab with *real* data— MALDI-ToF MS protocol optimization

Matrix-assisted laser desorption/ionization time of flight (MALDI-ToF) mass spectrometry is a laboratory method used to characterize the contents of a sample, and is used across many scientific domains [53, 54]. The end result of this experiment is a spectrum that is used to identify the components within the sample. Significantly, a variety of instrument settings must be specified, and these adjustable parameters affect the quality of the resulting data. Typical user-specified parameters include: (i) the accelerating voltage, (ii) the grid voltage, (iii) the pulse delay, and (iv) the number of laser shots per spectrum.

When performing MALDI-ToF spectrometry, it is common to perform a brute-force parameter sweep in order to identify the configuration that produces the highest quality spectrum. We were provided access to the data produced by two MALDI-ToF parameter sweeps for two separate samples. The biological context for these experiments was a study for the use of enzyme-polymer conjugation for chymotrypsin enzyme replacement therapy [55, 56]. The two samples used in these experiments included one with native chymotrypsin (CT), and the other with a chymotrypsin-polymer conjugate (CT-polymer). The goal of the parameter sweeps was to identify configurations that produce easily identifiable signals from each sample. Since the CT-polymer conjugate is a more complex sample, it should be expected to be harder to obtain such an identifiable signal.

Each data set consists of 120 MALDI-ToF spectra produced via a manual, brute-force grid search over the four user-specified parameters named previously. We ran PROTOCOL in a simulated cloud lab environment to demonstrate that the algorithm can identify the optimal parameter configuration in many fewer experiments. By simulated, we mean that the results of each experiment submitted to the job queue is simply fetched from the given data sets.

Our experiments considered several different definitions of spectral quality. We used the MATLAB Bioinformatics Toolbox to calculate peak height (intensity), peak width, and signal-to-noise ratio (SNR) of each spectrum. In general, a strong signal will include a large peak, narrow width, and small SNR. Additionally, as a means of combining these properties into objective measures that quantify multiple properties at the same time, we also used two linear combinations of these endpoints:

$$\text{Combo1} = \text{Peak Height} + \frac{1}{\text{Peak Width}} \tag{3.2}$$

$$\text{Combo2} = \text{SNR} + \text{Combo1} \tag{3.3}$$

The peak height, peak width, and SNR measurements were first scaled to the unit interval to ensure that different underlying distributions of each endpoint did not skew the objective.

For both datasets, we ran *in-silico* experiments using PROTOCOL that optimized for each of these endpoints. For each endpoint, this entails sequentially observing ground-truth values for experimental configurations according to the scheme outlined in Section 3.3. Each of the four input parameters are scaled to the unit interval, which transforms the input space onto the unit hypercube, as is done with IMGPO. An important difference between this setting and the example using the 1D sinusoidal function is that we choose experiments from one of 120 possible configurations. To do this, we use the division procedure outlined previously, but instead of evaluating the center coordinate of an interval, we calculate the Euclidean distance between the center coordinate and each transformed parameterization, and assign the closest parameter setting to the interval, where each parameterization is only allowed to be used once. In general, PROTOCOL can also handle any non-continuous variables in this way, so long as the variables are numerically encoded.

In our experiments, we follow IMGPO by using a GP with Matérn kernel with $\nu = 5/2$, given by:

$$k_M(x, x') = g\left(\sqrt{\frac{5\|x - x'\|^2}{l}}\right) \tag{3.4}$$

where

$$g(z) = \sigma^2(1 + z + \frac{z^2}{3})\exp(-z) \tag{3.5}$$

Again following IMGPO, we initialized the hyperparameters $\sigma^2 = 1$ and $l = 0.25$. These hyperparameters are optimized by maximizing the log marginal likelihood at the end of each iteration. Where y is a vector of observed ground truth, N is the number of observations, parameter $\theta = (\sigma^2, l)$, and $K_M(\theta)$ the kernel parameterized by θ, this likelihood is given as:

$$\log p(y|\theta) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log\det|K_M(\theta) + \sigma^2 I| - \frac{1}{2}y^T(K_M(\theta) + \sigma^2 I)^{-1}y \tag{3.6}$$

Given appropriate priors over the hyperparameters, they may be sampled from this distribution, though we omit this procedure from our experiments.

Other IMGPO-specific hyperparameters were set to their default settings. We set the level of parallelization to 4, meaning that PROTOCOL could request as many as four experimental conditions to observe at a time. In each experiment, we allowed the algorithm

Table 3.1: The frequency that each algorithm identifies the optimal experimental parameterization with the Native CT data for each endpoint.

Algorithm	Height	Width	SNR	Combo1	Combo2
PROTOCOL	100%	100%	0%	100%	100%
TS	43%	9%	40%	35%	38%
EI	35%	11%	69%	40%	38%
PI	36%	40%	63%	40%	38%
UCB	42%	9%	42%	36%	29%
Random	25%	19%	24%	23%	18%

Table 3.2: The frequency that each algorithm identifies the optimal experimental parameterization with the CT-polymer conjugate data for each endpoint.

Algorithm	Height	Width	SNR	Combo1	Combo2
PROTOCOL	100%	100%	100%	100%	100%
TS	42%	50%	41%	21%	31%
EI	38%	54%	53%	25%	26%
PI	39%	56%	52%	25%	26%
UCB	43%	53%	37%	28%	28%
Random	21%	31%	19%	22%	18%

to select a total of 25 observations. This corresponds to having only run 25 experiments in the simulated lab, as opposed to the complete set of 120, as was done in reality.

As points for comparison in a parallel setting, we also performed batch-mode GP optimization on the same data using standard acquisition procedures, including Thompson sampling (TS) [57], UCB, EI, and PI. Each of these methods choose 4 configurations to observe according to the acquisition procedure, update their GP model (including updating GP hyperparameters), then choose again using the updated model. The initial setting of the GP in each of these were the same as with PROTOCOL, and fully exhausting the 25 experiment request budget was used as the stopping criterion. We repeated each 100 times with different randomly selected training sets of size equal to the level of parallelization (in this case, 4). This imitates the simplest way one could initiate each of these procedures in a real cloud lab setting.

3.4.1 MALDI-ToF simulated Cloud Lab results

With both MALDI-ToF datasets, we find that PROTOCOL is able to find the optimal parameter configurations in most situations. Tables 3.1 and 3.2 show the frequency that PROTOCOL and other GP optimization algorithms identify the optimal parameter configuration for each endpoint. As PROTOCOL is a deterministic algorithm, it can only take values 100% or 0%. The other frequencies are calculated as the number of times each found the optimal setting out of the 100 repeated experiments. There was only one case where PROTOCOL did not find the optimal configuration (SNR, CT dataset), whereas the other GP-based algorithms have low success rates (mean = 34%; median = 36%; st.dev = 13.4%; max = 69%). Notably, PROTOCOL was able to identify the optimal configuration for each endpoint with the more difficult CT-polymer dataset. Comparably, the performance of the other GP optimization regimes tended to decrease with the CT-polymer dataset compared to just native CT.

To succinctly describe the selection behaviors of each algorithm, we focus our next analyses on the peak height endpoint, though similar summaries could be made for others. Figure 3.3-top shows the average progress of each algorithm in identifying the optimal experimental parameterization. That is, it shows the best observed value as a function of number of experiments requested and conducted. Experiments with both native CT and CT-polymer conjugates yield similar patterns. Initially, PROTOCOL lags behind the other GP optimization algorithms, but then quickly rises to the top and identifies better protocols (in the case of peak height, the *best* available protocol).

We emphasize that the apparent success of the random procedure is due to the relatively few experiments available (120) to be chosen from 25 times without replacement. Tables 3.1 and 3.2 show that the random procedure finds the true optimum much less frequently than the other conventional BO approaches, as expected.

As already described, the comparison GP algorithms require an initial training batch, and are only able to identify the optimal parameterization a fraction of the time depend-

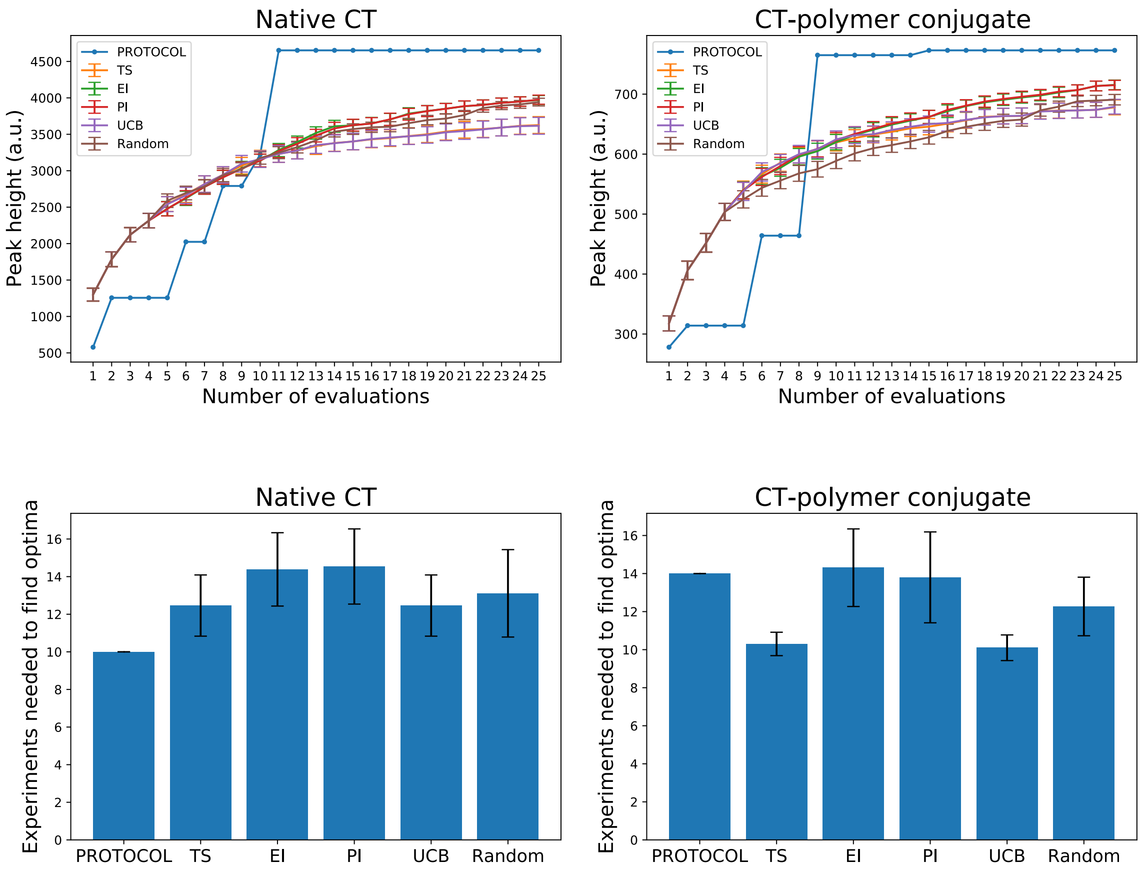


Figure 3.3: **Top row**. The ground truth peak height of observed MALDI-ToF experimental configurations is shown as a function of the number of total evaluations. The error bars in the non-PROTOCOL curves denote a mean ± 1 SEM calculated over 100 trials initialized with different randomly chosen training sets of size 4 (which is equal to the allowed level of parallelization). **Bottom row**. Again with the peak height endpoint, these show the number of evaluations each algorithm requested before identifying the optimal configuration. For the non-PROTOCOL algorithms, only the subset of the 100 trials that actually identified the optimal configuration are used. Error bars denote ± 1 SEM over this subset of trials.

ing on this initial training data. Figure 3.3-bottom focuses on the subset of trials that were able to correctly identify the optimal protocol parameterization. Over these trials, it shows how many experiments were requested on average before the optimal configuration was chosen. The native CT data PROTOCOL needed only 10 experiments, while the other methods required on average 12-14 experiments. This suggests that even when the other GP algorithms are able to identify the optimal experimental parameterization, PROTOCOL is capable of identifying the optimal solution more quickly.

With the CT-polymer conjugate data, TS and UCB seem to identify the optimal configuration in fewer experiments compared to all other methods (although, as previously mentioned and shown in Table 3.2, they find such optimal configurations less than 57% of the time). To investigate this behavior further, Figure 3.4 visualizes choices made by PROTOCOL compared to two trials that used GP-UCB— one that identified the optimal protocol and one that did not (results with TS are similar). In the figure, the configurations are enumerated along the horizontal axis, with the peak height of the spectra produced by the given experiment along the vertical axis. In general, configuration numbers closer to each other correspond to experimental configurations that are more similar to each other.

PROTOCOL's initial experiment corresponds to an experiment that is far from optimal. Still, the algorithm is able to quickly survey the input space, and converge on experimental configurations that yield large peak heights. With the UCB algorithm, the behavior is largely dependent on the initial training set that was randomly chosen. When there are training instances that are similar to the optimal configuration, the algorithm successfully identifies the optima, but is prone to converging on local optima more similar to the training data otherwise. This suggests that PROTOCOL is better at escaping local optima than the comparison algorithms.

Figure 3.6 visualizes experiments where we varied the level of parallelization from $k = 1$ to $k = 10$ using both MALDI-ToF datasets as well as with three commonly used optimization functions. For each data and choice of k, PROTOCOL performs the best.

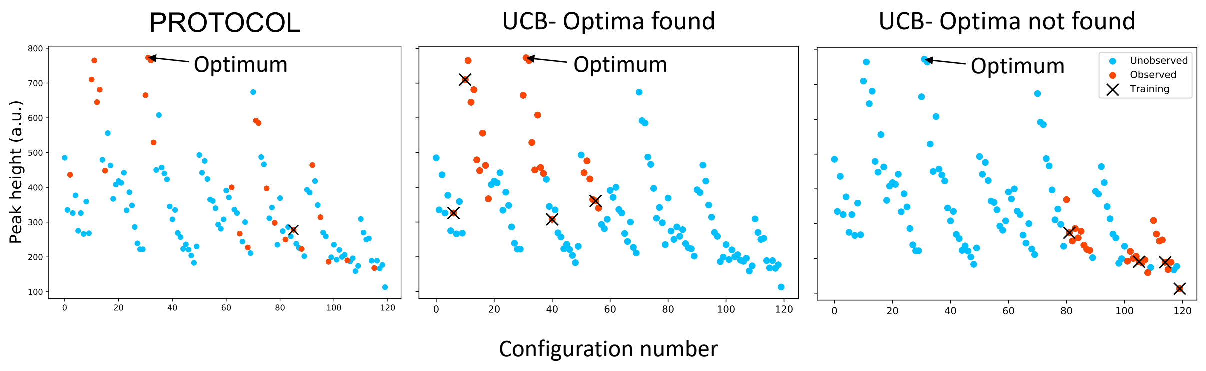


Figure 3.4: CT-polymer conjugate ground-truth peak heights for MALDI-ToF parameterizations selected by PROTOCOL and the GP-UCB algorithm. Two cases are shown for the GP-UCB algorithm— one where the algorithm identified the configuration that led to the maximum peak height, and one that did not. For each, the initial evaluation points are indicated by an 'x'. Whereas the initial point evaluated by PROTOCOL is a consequence of the algorithm (the central point of the input space), GP-UCB depends on an initial training set. The ability of GP-UCB to identify the optimal configuration is influenced by this initial set.

3.5 Simulated Cloud Lab with synthetic data— optimization test functions

In addition to running simulated cloud-lab experiments using real MALDI-ToF data, we also ran analyses using three general-purpose optimization functions, including the illustrative 1D sinusoidal function (Figure 3.1) as well as the Hartmann 3D and 6D functions. In order to perform the conventional GP-based approaches on these test functions, it is necessary to first sample over the input space ($\mathbb{R}^d$ where d equals 1, 3, or 6) in order to obtain a pool of inputs each algorithm can select from during optimization. The number of samples marks a trade-off between the granularity of the resulting optimization and computational expense. We created pools of size 5000, 4000, and 4096 for each of the 1D, 3D, and 6D functions respectively, where each data point corresponds to a lattice point on a d-dimensional grid. As done with our MALDI-ToF experiments, the conventional approaches used includes GP-EI, GP-PI, GP-UCB, and Random choice. Due to computational feasibility, we omit Thompson Sampling from these examples.

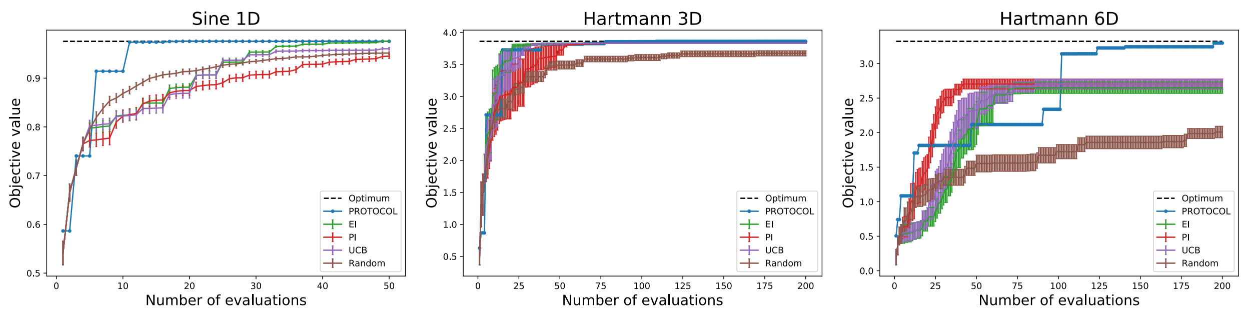


Figure 3.5: The objective values of selected function inputs shown as the number of total evaluations for 1D, 3D, and 6D test functions. The level of parallelization is 4 throughout. The error bars in the non-PROTOCOL curves denote a mean $\pm$ 1 SEM calculated over 10 (3D and 6D) or 100 (1D) trials initialized with different randomly chosen training sets of size 4 (which is equal to the allowed level of parallelization).

Figure 3.5 shows how all approaches perform when the level of parallelization is set to 4 (this analysis is analogous to that with the MALDI-ToF data shown in the top of Figure 3.3). With both the 1D sinusoidal and the Hartmann 3D functions, PROTOCOL more quickly approaches its optimal selection, and finds an input for both functions that exceeds those found by the conventional approaches. In the case of the Hartmann 6D function, while the conventional approaches are quicker to find their optimal solutions (aside from the random choice procedure), PROTOCOL ultimately finds better inputs over the course of the allotted 200 evaluations. The need to perform the initial sampling over the input space is what ultimately hinders the abilities of the conventional approaches to optimize each function— it is likely that better inputs exist in the non-sampled space. PROTOCOL's division scheme avoids this sampling altogether, and allows the resolution of the input space to adapt with the optimization procedure, and allows it to identify inputs that better optimize these three test functions.

In most practical settings, we imagine that most would have access to a modest level of parallelization. We thus ran experiments where the allowable level of parallelization ranged from 1 to 10. In Figure 3.6, we show the regret of each approach as a function of the level of parallelization. Simple regret is given by $R(x^+) = sup_{x \in \mathcal{X}} f(x) - f(x^+)$, where x^+ is the configuration found by the algorithm, and log regret is the base-10

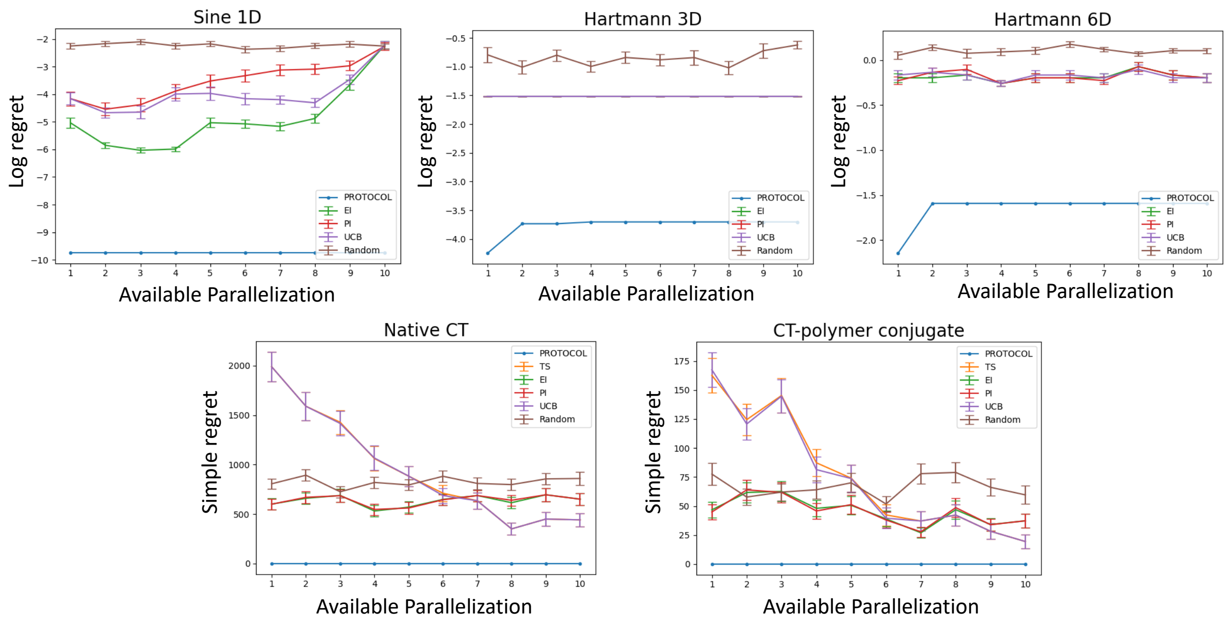


Figure 3.6: PROTOCOL's performance as a function of the level of parallelization (batch size). **Top row**. PROTOCOL outperforms conventional BO approaches when optimizing a 1D, 3D, and 6D test function. Whereas the accuracy of the conventional approaches are limited by a required initial sampling over the input space, PROTOCOL's division scheme does not need this, and thus better optimizes the function given the same experimental budget, regardless of the batch size. **Bottom row**. PROTOCOL finds the optimal MALDI-ToF configuration on both the Native CT and CT-polymer conjugate data across all tested batch sizes, and consistently outperforms the conventional BO approaches. Note that when the batch size is 1, PROTOCOL and IMGPO will make the same selections.

logarithm of simple regret.

With all the test functions, as well as the two real MALDI-ToF data sets, PROTOCOL performs better than the conventional approaches at any given level of parallelization, k. With respect to the Hartmann 3D and 6D functions, PROTOCOL performs clearly best when $k = 1$. In that case, PROTOCOL by construction will choose the same experiments that would be chosen by the IMGPO algorithm. Thus, with the Hartmann 3D and 6D functions, the extra experiments selected as a result of allowed parallelization over the course of the optimization impeded PROTOCOL's ability to find an optimal solution. This suggests that the added exploration for each $k > 1$ did not yield any promising search direction, and came purely at the expense of exploitation. Still, even in these cases, PROTOCOL manages to easily perform better than the conventional approaches. In each of the other three optimization problems we investigate, two of which used real-world experimental data, we did not observe this phenomenon.

By construction, k denotes the *maximum* number of experiments that PROTOCOL may request in a particular round. If fewer experimental configurations lie on the frontier, than PROTOCOL simply requests only those experiments. This differs from the conventional approaches, which always choose k many experiments according to the respective acquisition function. On one hand, this means that PROTOCOL has the ability to act more conservatively than allowed when the GP model suggest that strictly fewer than k experiments are necessary, or the hierarchical tree is too shallow. On the other hand, this means that PROTOCOL may take longer to exhaust a given experimental budget. Figure 3.7 visualizes this trade-off for each of the objectives we have optimized with PROTOCOL. An experimental round refers to when the algorithm requests a batch of experiments. As an example, a purely serial procedure ($k = 1$) given a budget of 50 experiments would take 50 experimental rounds. If the same procedure instead had $k = 2$ and fully utilized this level of parallelization in each round, it would take 25 experimental rounds.

PROTOCOL exhibits similar behavior in each optimization problem. As the allowable batch size increases, the total experimental rounds initially decrease before eventually leveling off. This indicates that PROTOCOL is most efficient when given modest levels of parallelization (a case we believe to be most common in experimental settings). In practice, the gap between the "Full batches" and PROTOCOL corresponds to the amount of time proportional to the complexity of the experiment needed to satisfy an experimental budget for a given level of parallelization. Of course, exhausting an experimental budget more quickly does not necessarily translate into better experimental outcomes— PROTOCOL after all better optimizes each objective we tested compared to the conventional approaches, which use full batches. Still, in cases where the experiments are most time consuming, there could be greater benefit or financial incentive in minimizing the number of experimental rounds. We leave work towards reducing the gap while maintaining PROTOCOL's optimization capabilities for future work.

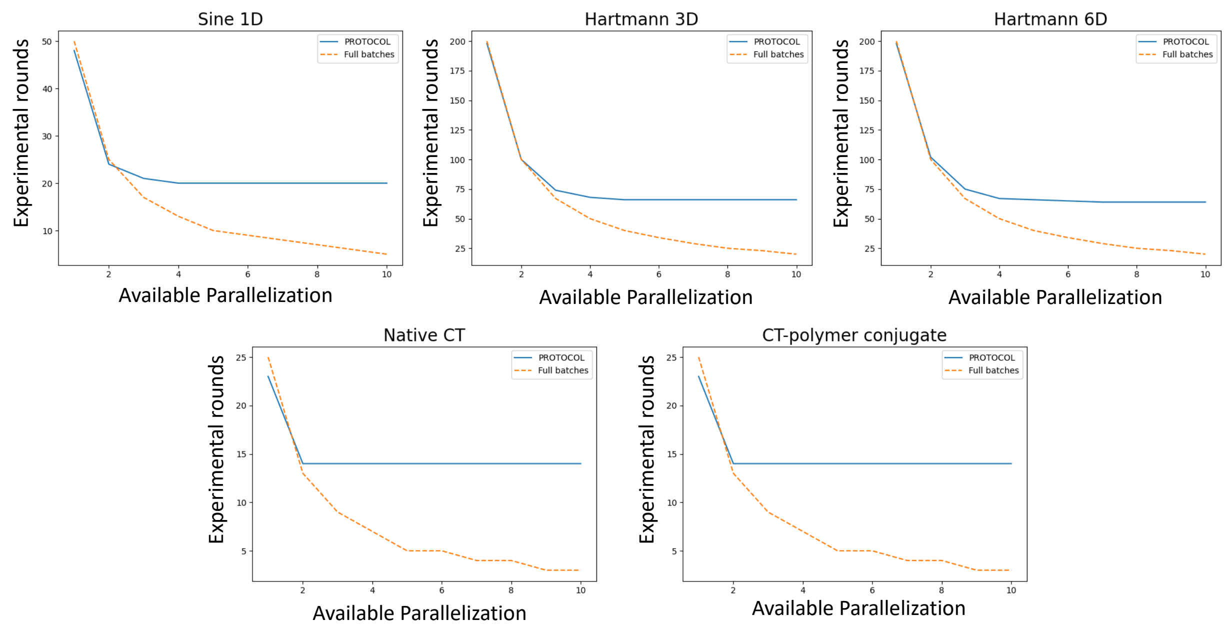


Figure 3.7: The number of experimental rounds as a function of the allowable level of parallelization for PROTOCOL compared to procedures that use full batches (e.g. conventional GP-based methods) on test objective functions (**Top row**) and real MALDI-ToF data (**Bottom row**).

3.6 Real-world Cloud Lab— HPLC protocol optimization

High performance liquid chromatography (HPLC) is an analytical chemistry technique used to separate and quantify components of complex mixtures. The method uses pressurized liquid solvent to force a sample through a column containing specialized solid adsorbent material. This material interacts with each component of the sample differently, causing each to travel through the column at different rates, thus separating the mixture. A detector measures the absorbance of each sample as they elute at different times. Chromatograms are generated from these measurements, which allows for identification of each component [58].

We used PROTOCOL to optimize an HPLC experimental design in a real cloud lab setting using Emerald Cloud Labs (ECL). This means that whenever PROTOCOL requested an experimental configuration to observe, we remotely executed an actual experiment to be performed at ECL's laboratory in South San Francisco, CA. Specifically, the HPLC experiment involved the separation of a three component mixture of the organic compounds phenol, toluene, and 2,5-xylenol in a water/methanol solvent. The accuracy of the separation and quantification steps are sensitive to multiple parameters. In each experiment, we chose a setting for each of the following parameters:

- **Flow rate**— The speed of the fluid through the HPLC pump. Selected from the range 0.2-2mL/min.
- **Injection volume**— The physical quantity of sample loaded into the flow path for measurement. Selected from the range 1-50μL.
- **Column temperature**— The temperature of the HPLC column. Selected from the range 25-45°C.
- **Absorbance wavelength**— The wavelength used by the detector to identify samples flowed through the column. Selected from 260-285nm.
- **Solvent ratio**— The proportion of methanol to water in the solvent. Selected from 65:35, 70:30, 75:25, and 80:20.
- **Gradient**— The proportion of solvent:sample flowed through the column over time. Selected from a nonlinear (the default ECL setting), constant, quick linear, linear, and slow linear setting.

Unlike the MALDI-ToF experiments where we wanted to optimize the signal from a single sample with a single component, here we want to simultaneously optimize for the sample (by adjusting solvent ratios) *and* the instrument settings that best resolve the three compounds mixed within a sample. To do this, we used the resolution of the chromatogram (R_S) as our objective function. This is a commonly used metric to describe HPLC spectra, and is given by:

$$R_S = \sum_{i=1}^{n-1} \frac{c(t_{i+1} - t_i)}{w_i + w_{i+1}} \tag{3.7}$$

where n is the number of peaks, t_i refers to the time component i elutes from the HPLC column, w_i refers to the half-height width of component i, and c is a constant that arises from assuming each peak takes the shape of a Gaussian. Essentially, this objective quantifies how clearly distinguished all adjacent peaks are from each other. The larger the value, the more clear the separation. We used ECL's built in software in order to pick peaks from the chromatogram.

We had access to three threads on ECL, meaning we could run up to three experiments at a time. We thus allowed PROTOCOL to select up to this many experiments per request. Since each possible parameter configuration was either chosen from a finite list (gradient and solvent ratio) or subject to a finite level of precision when measuring (flow rate, injection volume, column temperature, and absorbance wavelength), we used the same strategy as with the MALDI-ToF experiments when assigning a parameter configuration to an interval within PROTOCOL's hierarchical tree. The input space was similarly scaled to the unit hypercube, and we used the same GP and IMGPO specific hyperparameters detailed in Section 3.4.

Since we were running real experiments with a limited number of threads on ECL, we did not have the time or resources to conduct experiments according to alternative acquisition functions (TS, EI, etc.) as with the MALDI-ToF simulated cloud lab data. Thus, we also used Latin Hypercube Sampling (LHS) as a baseline to select experiments to conduct on ECL. Unlike with PROTOCOL and other conventional BO strategies, this is not a sequential process, but rather a form of randomly sampling a prescribed number of experiments to conduct. This allowed us to submit all the experiments at once, and allow them to complete according to available resources without the need for further intervention. We then compared the results of the LHS versus PROTOCOL in a separate set of experiments. In total, we executed 18 experiments selected by PROTOCOL, and (separately) 18 experiments selected via LHS on ECL. Given our available resources, this was the number of experiments we estimated could be run in 2 month's time (1 month for each approach).

Table 3.3: The top 3 HPLC spectra resolutions according to configurations chosen by PROTOCOL, LHS, and Sim.-LHS, which refers to a simulated LHS method (see text for details), where mean ± 1 standard deviation are shown over 500 runs. "Best" refers to the greatest observed resolution.

Algorithm	Best	2^{nd} Best	3^{rd} Best	Average
PROTOCOL	17.4	16.4	8.5	14.1
LHS	44.1	11.5	2.9	19.5
Sim.-LHS	15.8 ± 2.6	12.2 ± 1.5	7.4 ± 1.5	11.8

3.6.1 HPLC real-world Cloud Lab results

PROTOCOL and LHS are both able to identify HPLC configurations that yield high resolution. Table 3.3 shows the top three scoring resolutions obtained from configurations selected by both methods. While PROTOCOL's top scoring configuration yielded a resolution of 17.4, strikingly, LHS selected a configuration that obtained a resolution of 44.1. Since LHS is ultimately a random sampling procedure, it was unexpected that it was able to identify such a highly resolved spectrum. This led us to investigate precisely how unlikely this finding was.

We accomplished this through simulation. We trained a random forest regression model with the 36 experiments obtained from PROTOCOL and LHS, using the measured resolution as the label. We then repeated the LHS sampling procedure used to generate experiments 500 times. This allowed us to predict the resolution of these samples using the random forest regression model. The result of these 500 LHS simulations are shown by Sim.-LHS in Table 3.3 as a mean ± 1 standard deviation.

We find that the Sim.-LHS results do not yield resolutions anywhere near as large as the 44.1 found by the experiments on ECL. Rather, the best prediction on average had a resolution of 15.8. For reference, PROTOCOL's best is about one standard deviation larger than this value. Furthermore,the maximum predicted resolution over all 500 simulations was only 26.9, giving a 39% difference in the maximum resolution observed in the ECL experiments. This is evidence that the configuration that yielded a 44.1 was highly unlikely to have been chosen.

Table 3.4: HPLC parameters selected by PROTOCOL and LHS that yielded highest resolution.

Parameter	PROTOCOL	LHS
Flow rate (mL/min)	1.0	0.8
Injection volume (μL)	38.3	6.1
Column temperature (°C)	35	26.7
Absorbance wavelength (nm)	273	284
Solvent ratio	75:25	65:35
Gradient	quick linear	nonlinear

The best experimental configurations selected by PROTOCOL and LHS were quite different. Table 3.4 lists the best settings selected by both methods. PROTOCOL not finding a similar configuration suggests that, within this search space, more than 18 experiments were needed. We emphasize that the choice to perform 18 experiments was a product of time and resource constraints.

Figure 3.8 shows peaks that correspond to experiments selected by PROTOCOL and the outlier LHS result. The panel on the left is typical of the chromatogram one obtains using a configuration chosen at random (including those *typically* chosen via LHS). The panel on the right is an example of a reasonably high quality chromatogram, albeit one that is unlikely to have been observed when using LHS, as previously argued. The middle panel is the best one found by PROTOCOL given a budget of 18 experiments. It is clearly an improvement over the left panel. In particular, the built-in automatic peak picking and resolution-calculating software only identified two peaks in the left panel, but identifies three peaks in the middle panel (the correct number). Still, the middle panel is far from optimal. We hypothesize that given a larger experiment budget, PROTOCOL would have continued to find better configurations. Moreover, as shown in the bottom row of Table 3.3, the typical best resolution obtained via LHS is actually lower than that of the middle panel. That is, the middle panel is probably representative of what one might obtain via a LHS with a budget of 18 experiments over this search space.

As a final note, we emphasize the advantage PROTOCOL's use of parallelism has over

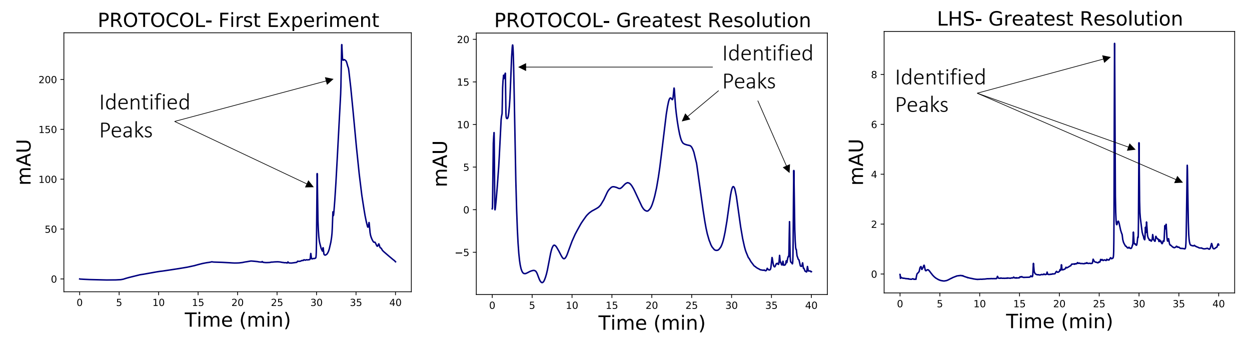


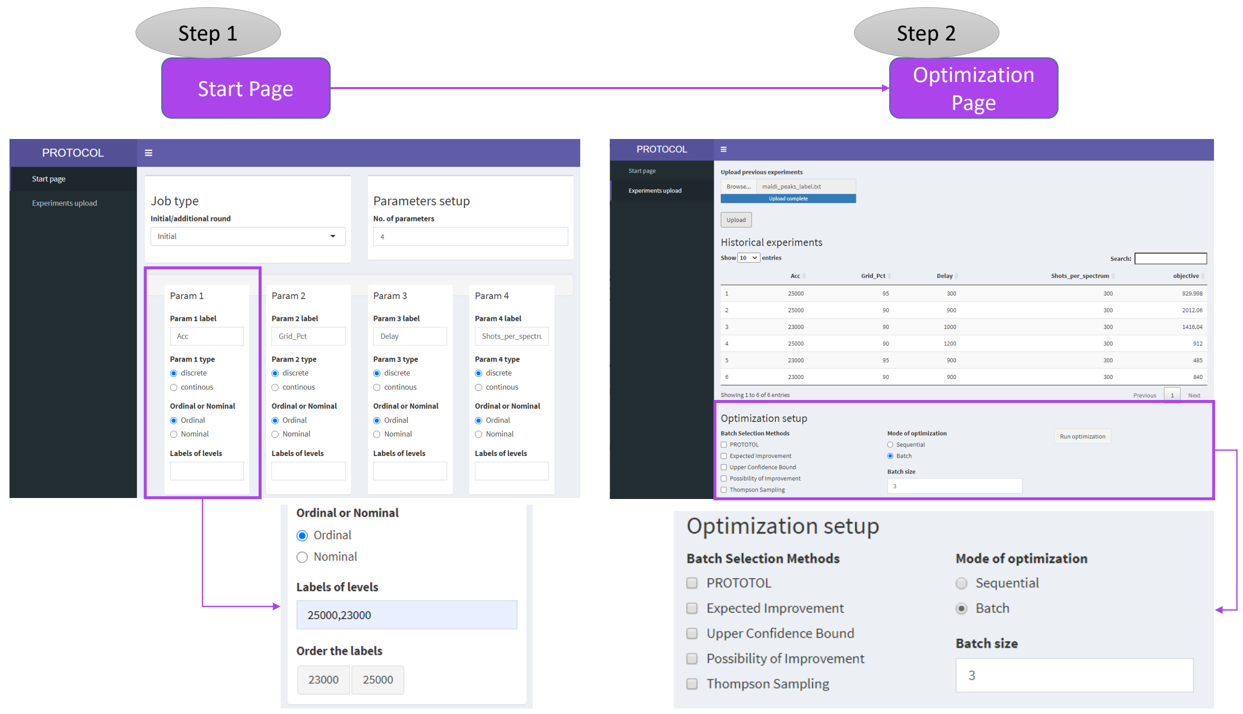
Figure 3.8: Chromatograms corresponding to the first experimental configuration chosen by PROTOCOL (left) as well as the experimental configuration that yielded the greatest resolution chosen by PROTOCOL (middle) and LHS (right).

purely serial procedures (such as IMGPO). While we only ran a modest number of experiments selected by PROTOCOL (18), it still took 26 days to complete. Had we had run each experiment sequentially one after another, we estimate from queue times and experiment execution times that it would have taken *at least* 41 days, meaning we saved greater than 15 days worth of work.

3.7 R Shiny application

We have implemented PROTOCOL as both a Python library, and as part of a stand-alone application written in R Shiny[59] [3]. The Shiny app lets the user initialize and run optimization jobs using PROTOCOL or several conventional BO methods (TS, EI, PI, UCB). The user defines the optimization problem via a start page by specifying the names and types of the parameters to optimize over (by hand, or by loading a configuration file (Figure 3.9-left). After confirming all parameters, the users can upload historical parameters combinations with their observed objective values (if available). The users then selects the optimization method (e.g. PROTOCOL) and the degree of parallelism (k). If desired, the user can select more than one optimization method to consider different selections (Figure 3.9-right). The application computes and displays the next parameter

[3]The R Shiny application was implemented by paper [21] co-author Zhiyun Gong.



Job type

Initial/additional round: Initial

Parameters setup

No. of parameters: 4

	Param 1	Param 2	Param 3	Param 4
label	Acc	Grid_Pct	Delay	Shots_per_spectru
type	[✓ discrete] continous	[✓ discrete] continous	[✓ discrete] continous	[✓ discrete] continous
Ordinal or Nominal	[✓ Ordinal] Nominal	[✓ Ordinal] Nominal	[✓ Ordinal] Nominal	[✓ Ordinal] Nominal
Labels of levels				

Ordinal or Nominal: [✓ Ordinal] Nominal

Labels of levels: 25000,23000

Order the labels: 23000 25000

Upload previous experiments: maldi_peaks_label.txt — Upload complete

Historical experiments

Show 10 entries

	Acc	Grid_Pct	Delay	Shots_per_spectrum	objective
1	25000	95	300	300	929.998
2	25000	90	900	300	2012.06
3	23000	90	1000	300	1416.04
4	25000	90	1200	300	912
5	23000	95	900	300	485
6	23000	90	900	300	840

Showing 1 to 6 of 6 entries

Optimization setup

Batch Selection Methods: PROTOTOL, Expected Improvement, Upper Confidence Bound, Possibility of Improvement, Thompson Sampling

Mode of optimization: Sequential [✓ Batch]

Batch size: 3

Figure 3.9: **Left**. The Shiny app start page, where the user can initialize an optimization problem by defining the parameters to optimize over. **Right**. The Shiny app data upload page, where the user can upload previously evaluated data and select the optimization algorithm to use.

combination(s) to run. Any existing data and the suggested combinations can be saved to a file, in order to save state, as the user waits for the experiments to run. When the results of the experiments are known, the user updates the file and then loads it into the application, which then suggests the next experiment(s) to run, and so forth.

3.8 Discussion

Cloud-based laboratories present an emerging and exciting new model for conducting scientific experiments. While they can provide access to sophisticated equipment and the ability to run experiments in parallel, performing experimental optimizations in a way that fully utilize these capabilities is an unexplored area. To this end, we have developed PROTOCOL, an algorithm that performs closed-loop optimization of experimental protocols within this setting. Built on the framework of recent bound-based optimization methods that come with convergence guarantees, we believe PROTOCOL to be the first such method that explores optimization of experimental designs within this environment.

In our MALDI-ToF experiments, we compared PROTOCOL to conventional BO approaches. We found that PROTOCOL more reliably identified the optimal configuration across five different endpoints and with two different samples. The ability of the conventional approaches to select desirable parameterizations is highly dependent on the initial data used to train the models. While there are ways to promote exploration of the search space with such GP-based approaches, this itself entails an auxiliary optimization routine, which could be computationally prohibitive and/or prone to over-fit limited data. We found that PROTOCOL's DIRECT-like division scheme over the input space was able to combat these issues.

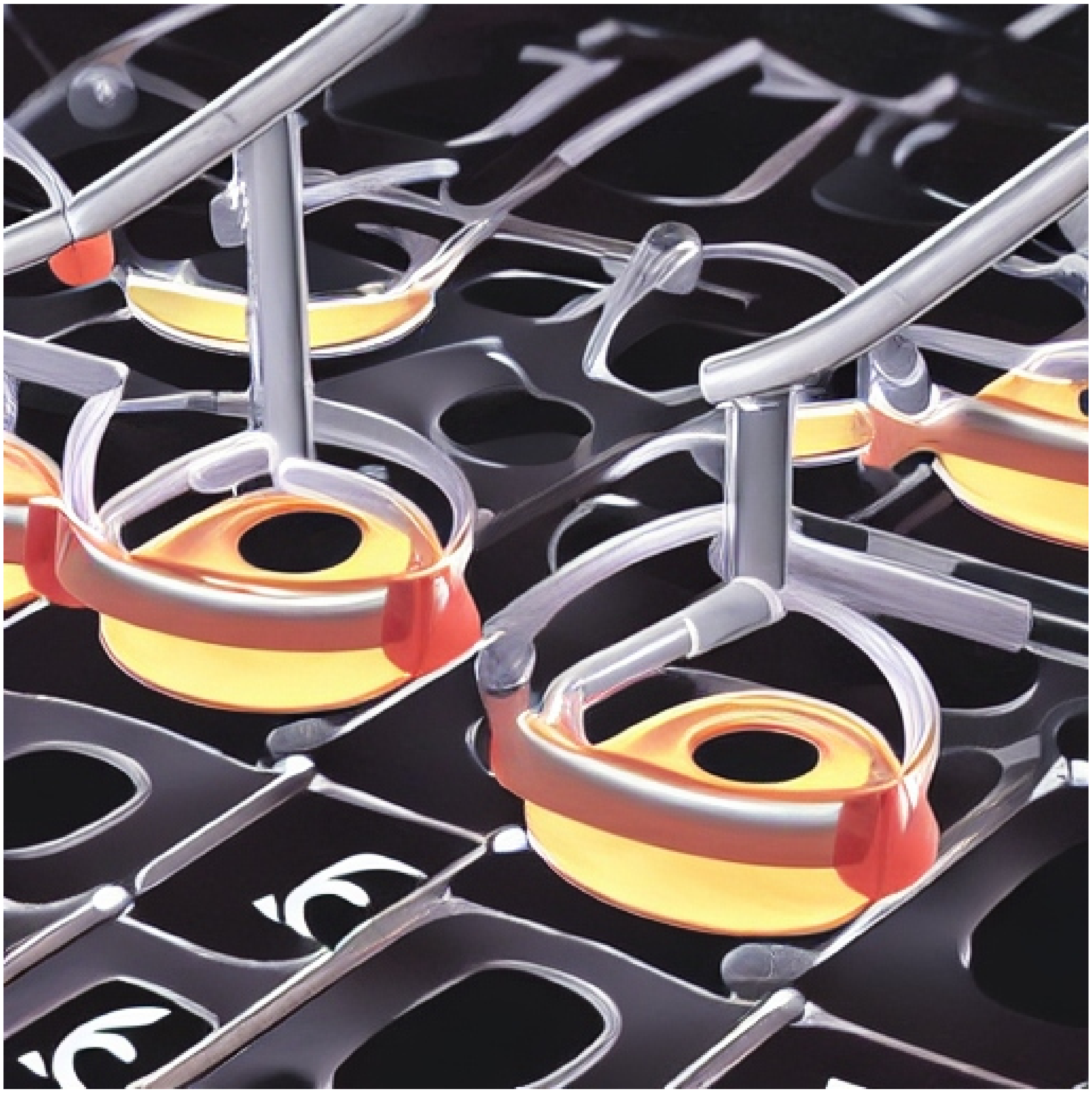
The deterministic nature of the dividing scheme ensures that the underlying GP model is exposed to instances that are representative of a wide range of the input space, so it naturally promotes exploration. Additionally, whereas the conventional approaches select parameterizations primarily based on what the model believes to be best at any given time, PROTOCOL's strategy is to present the model with parameterizations selected according to the division scheme, and requesting experiments the model is sufficiently uncertain about. This ensures that selections made by PROTOCOL are not as exclusively influenced by what the model has been exposed to previously, which leads to a regularization-like behavior. Another area for future work is to integrate the results of technical and/or biological replicates into the optimization logic.

While our work with PROTOCOL is a promising start towards experimental optimization in real Cloud Lab settings, there is still much room for improvement. In our HPLC experiments using ECL, we noted that PROTOCOL likely needed more than 18 selections to identify an experiment as well resolved as the anomalous LHS finding. Given that our search space over HPLC parameters was orders of magnitude larger than that of the MALDI-ToF parameters, it is not altogether surprising that a larger number of experiments could be necessary.

There are a number of algorithmic improvements worth pursuing. This includes identifying better strategies to initialize PROTOCOL's hierarchical tree. At present, the algorithm always starts by selecting the center of the hyperrectangle, no matter how much prior data are available. An alternative approach might initialize a tree that is already grown to some depth, and using some subset of leaf nodes in this tree as a starting point. The use of the frontier would provide a natural way to still select from nodes that were created in the tree's initiation. Since this tree could provide a larger initial pool of experiments to choose from, it could help provide a better initial search in larger dimensional settings.

In this chapter, we have shown how to adapt a principled bound-based BO routine to work in a parallel setting. We further demonstrate that such an approach is capable of being executed in a real cloud lab environment, and show that it performs favorably relative to conventional BO routines on real-world data. While we chose to adapt PROTOCOL from IMGPO due to its theoretical guarantees, there are of course other sophisticated BO algorithms that could be similarly adapted to work in our cloud-based setting (e.g. Kathuria, Deshpande, and Kohli [60]). We leave formal comparisons between the performance of such alternative approaches in the cloud lab to future work.

One final point is that PROTOCOL is broadly more applicable than the experimental design use case we have presented here. Since it shares similarities with other bound-based optimization algorithms (most directly IMGPO), it could be used to optimize most any black-box function. It would be interesting to apply PROTOCOL to tackle such problems in parallel, such as hyperparameter optimization for deep models, especially those applied to biological settings.



Chapter 4

Regularized Bayesian Optimization for Directed Protein Evolution

In this and the subsequent chapter, we shift our focus from automation within the Cloud Lab to that of protein design and engineering. As with the previous chapter, we continue to focus on dynamic, sequential experimental decision making algorithms. Here, we again build upon Bayesian optimization techniques, but now applied to selecting (*in silico*) mutagenesis experiments within the context of Directed Evolution.

4.1 Introduction

The field of protein engineering seeks to design molecules with novel or improved properties [61]. Many techniques used in protein engineering fall into two broad categories: *rational design* [62] and *directed evolution* (DE) [63]. Rational design uses model-driven *in silico* combinatorial searches to identify promising candidate designs, which are then synthesized and tested experimentally. Directed evolution, in contrast, involves iterative rounds of saturation mutagenesis at select residue positions, followed by *in vitro* or *in vivo* screening for desirable traits. The most promising sequences are then isolated and used to seed the next round of mutagenesis.

Traditionally, directed evolution is a model-free approach. That is, computational models are not used to guide or simulate mutagenesis. Recently, however, a technique for incorporating Machine Learning (ML) into the DE workflow was introduced [64]. Briefly, this ML-assisted form of DE uses the screening data from each round to update a model that predicts the effects of mutations on the property being optimized. The mutagenesis step in the next round of DE is then biased towards generating sequences with the desired property under the model, as opposed to generating a uniformly random sample. ML-assisted DE has been shown to reduce the number of rounds needed to find optimal sequences, relative to traditional (i.e. model-free) DE [64].

Significantly, the models learned in ML-assisted DE are *myopic* in the sense that they only consider the relationship between a limited set of residues (e.g. those in a binding interface) and the screened trait (e.g. binding affinity). Thus, DE may improve the measured trait at the expense of those that are unmeasured, but nevertheless important (e.g. thermostability, solubility, subcellular localization, etc.). The primary goal of this chapter is to introduce an enhanced version of ML-assisted DE that is biased towards native-like designs, while optimizing the desired trait. By 'native' we mean that the optimized design still has high probability under a generative model of protein sequences, or is predicted to be thermodynamically stable, according to a given energy function.

The intuition behind this approach is that any sequence with these properties is likely to respect factors that are not directly accounted for by the fitness model, such as epistatic interactions between the mutated residues and the rest of the protein [65], among others.

Our method performs Bayesian optimization [34] and incorporates a regularization factor derived from either a generative model of protein sequence or an *in silico* prediction of structural thermodynamic stability. In this chapter, we refer to these as “Evolutionary" or “Structure-based" regularization factors, respectively. Our method is agnostic with respect to the means by which the regularization factors are computed. For example, we evaluated three distinct generative models of protein sequences, including a contextual deep transformer language model [66], a Markov Random Field (MRF) generated by the GREMLIN algorithm [42], and profile Hidden Markov Model (HMMs) [67]. For structural thermodynamic stability, we use the FoldX protein design Suite [68] to calculate changes in Gibb’s free energy ($\Delta\Delta G$) associated with new designs.

We first demonstrate our method by re-designing the B1 domain of streptococcal protein G (GB1) at four residues to maximize binding affinity to the IgG Fc receptor. Next, using data obtained from deep-mutational scans, we use ML-assisted DE to investigate the factors governing the relationships between sequence and clinically relevant phenotypes. Specifically, we (i) identify variants of the RING domain of the BRCA1 protein for which the activity of tumor suppressor gene E3 ubiquitin ligase is maximal, and (ii) identify variants of the receptor binding domain of the SARS-CoV2 Spike protein that optimize binding affinity to the ACE2 receptor. Our results on these three targets demonstrate that a structure-based regularization term usually leads to better designs than the unregularized versions, and almost never hurts. The results using an evolutionary-based regularization are mixed; it leads to better designs for GB1, but worse designs for BRCA1. We also demonstrate that a Bayesian approach to ML-assisted DE outperforms the (non-Bayesian) approach introduced in Wu et al. [64]. Specifically, we show that our approach reduces the wet-lab burden to identify optimal GB1 designs by 67%, relative to the results presented in Wu et al. [64] on the same data.

4.1.1 Directed protein evolution

Directed evolution (DE) is an iterative technique for designing molecules. It has been used to create proteins with increased stability [69], improved binding affinity [70], to design new protein folds [71], to change an enzyme's substrate specificity [72] or ability to selectively synthesize enantiomeric products [64], and to study fitness landscapes [73], among others. Given an initial sequence, the primary steps in directed evolution are: (i) *random mutagenesis*, to create a library of variants; (ii) *screening*, to identify variants with the desired traits; and (iii) *amplification* of the best variants, to seed the next round. Each step can be performed in a variety of ways, giving rise to multiple options for performing DE. For example, the mutagenesis step can be performed one residue at a time, called a single mutation walk (Figure 4.1-top), or simultaneously at multiple positions, followed by genetic recombination (Figure 4.1-bottom). The key to the success of DE is that it performs what is in effect a parallel *in vitro* or *in vivo* search over designs that simultaneously *explores* the design space (via the mutagenesis step) while *exploiting* the information gained in previous rounds (via the amplification step). The exploratory aspect of DE is effectively a strategy for getting out of local optima on the underlying fitness landscape.

4.1.1.1 Machine learning-assisted directed protein evolution

While effective, the mutagenesis, screening, and amplification steps in DE are expensive and time-consuming, relative to *in silico* screens using statistical models or tools such as FoldX [68]. In an effort to reduce these experimental demands, a Machine Learning-assisted approach to DE was introduced recently [64]. This ML-assisted form of DE is summarized in Figure 4.2. The key difference between traditional and ML-Assisted DE is that the data generated during screening are used to (re)train a model that is capable of predicting the property of interest for a given sequence. The model, $\hat{f}$, may be a classifier or regression model and acts as a surrogate for the true function, f (i.e.

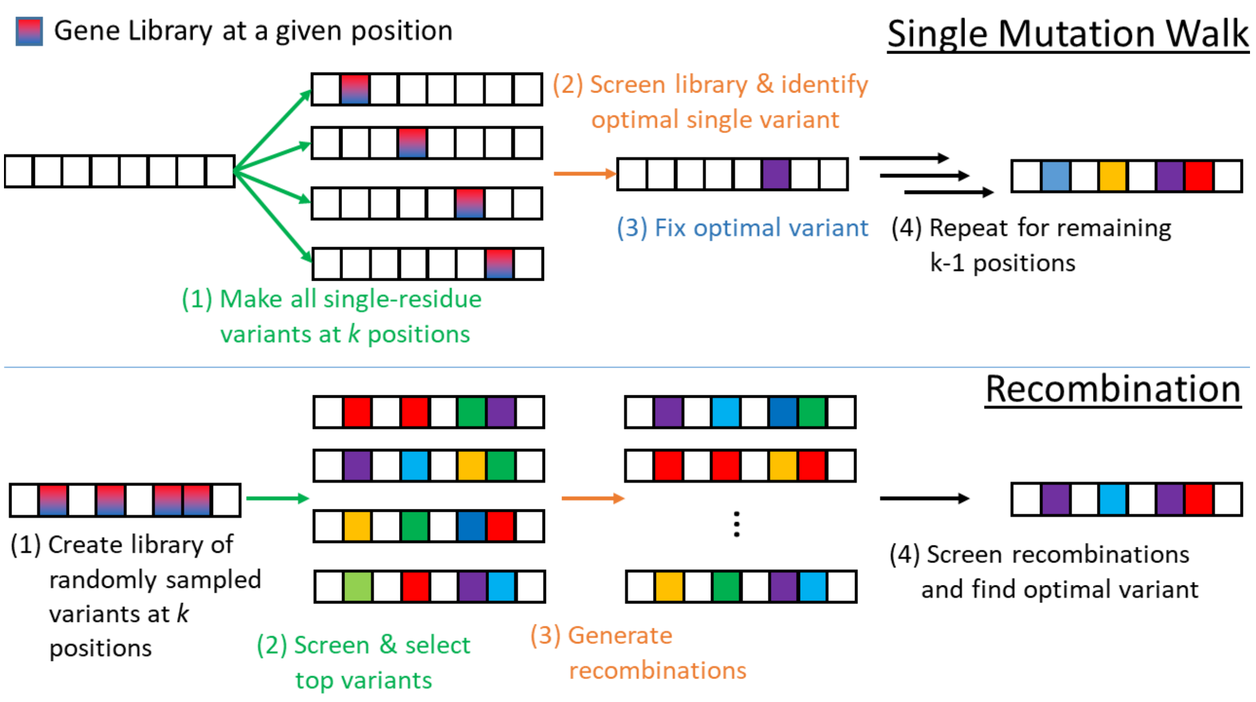

Gene Library at a given position
Single Mutation Walk
(2) Screen library & identify optimal single variant
(3) Fix optimal variant
(4) Repeat for remaining k-1 positions
(1) Make all single-residue variants at k positions
Recombination
(1) Create library of randomly sampled variants at k positions
(2) Screen & select top variants
(3) Generate recombinations
(4) Screen recombinations and find optimal variant

Figure 4.1: **Traditional, model-free approaches to directed evolution**: (*Top*) The 'single mutation walk' approach to directed evolution. The library of variants is the union of k libraries created by performing saturation mutagenesis at a single location. The resulting library, therefore, has $20k$ variants. The library is screened to find the single variant that optimizes the measured trait. That variant is fixed and the procedure is repeated for the remaining $k-1$ positions. (*Bottom*) The library of variants is created by performing saturation mutagenesis at k positions. The top variants are identified through screening. Those variants are randomly recombined to generate a second library, which is then screened to find the top design.

the one used by nature). The model can thus be used to perform an *in silico* screen over designs. Promising designs are then synthesized/cloned and screened in the lab. The key assumption made by ML-assisted DE is that the cost of performing an *in silico* screen using $\hat{f}$ is much lower than running wet-lab experiments (i.e. evaluating f). This assumption is almost always valid.

Like rational design, ML-assisted DE uses computational models, but the nature of those models is rather different. For one, the models used in ML-assisted DE make predictions corresponding to the quantity measured in the screening step, whereas the models used in rational design tend to be based on physical or statistical energy functions, and are therefore making predictions about the energetic favor of the design. Second,

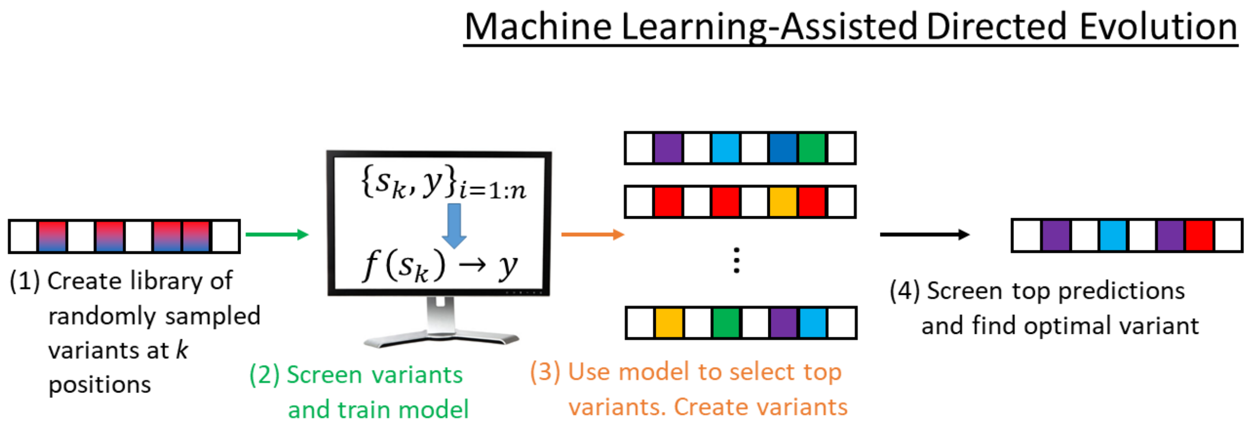

Machine Learning-Assisted Directed Evolution
$\{s_k, y\}_{i=1:n}$
$f(s_k) \rightarrow y$
(1) Create library of randomly sampled variants at k positions
(2) Screen variants and train model
(3) Use model to select top variants. Create variants
(4) Screen top predictions and find optimal variant

Figure 4.2: **Machine Learning-assisted directed evolution**: The first step in ML-assisted DE is the same as for traditional DE (see Fig 4.1). A library of variants is created via mutagenesis. Existing data, $\mathcal{S} = \{s_k, y\}_{i=1:n}$ are used to train a classifier or regression model, $f(s_k) \rightarrow y$, which is then used to rank variants via an *in silico* screen. The top variants are then synthesized/cloned and screened using *in vitro* or *in vivo* assays. The data from the i^{th} round is added to $\mathcal{S}$ and used in subsequent DE rounds.

the models used in ML-assisted DE are updated after each DE round to incorporate the new screening data, and thus adapt to protein-specific experimental observations. The models used in rational design, in contrast, are typically fixed. Finally, the models used in ML-assisted DE are myopic, in the sense that they only consider the relationship between a small subset of sequence positions (e.g. a binding site) and the measured quantity. The models used in rational design, in contrast, generally consider the *entire* sequence, and are thus better suited to filtering energetically unfavorable designs. The technique introduced in this chapter seeks to combine the strengths of both methods; our method uses a fitness model that adapts to the experimental data, but also considers the utility of the mutations across the entire sequence.

4.1.2 Bayesian optimization

Protein design is a quintessential black-box optimization problem with an expensive objective function, and so it is a natural candidate for Bayesian optimization, including in the context of ML-assisted DE (e.g. Yang, Wu, and Arnold [74]).

Recall that, in the Bayesian optimization framework, the objective function f is unknown, and so it is modeled as a random function with a suitably defined prior, $P(f)$. At the beginning of each iteration, an algorithm known as the *acquisition function* [30] selects one or more candidate designs to evaluate, based on $P(f)$. The resulting data are used to compute a posterior distribution over f, which becomes the prior for the next round. Typical choices for priors/posteriors include Gaussian Processes [38] and Tree-structured Parzen estimators [32]. In the context of this chapter, f is the function that maps protein sequences to an experimentally measured property (e.g. fitness or binding affinity/activity), and our goal is to find $s^* = \arg\max_{s \in S} f(s)$, where S is the space of sequences. Naturally, the evaluation of f is expensive, because it requires the previously described DE mutagenesis and screening steps, but a surrogate function, $\hat{f} \sim P(f)$ can be used to perform *in silico* screens.

Our proposed approach uses custom acquisition functions that consider whether a given sequence resembles proteins observed in nature [75, 76, 77], in addition to the usual considerations of exploration and exploitation. Computationally, this is implemented using a regularization term, as defined in Section 4.4.1. We evaluated two types of regularization terms: (i) an evolutionary factor calculated using generative models of protein sequences, and (ii) a structure-based factor calculated using the program FoldX.

4.1.3 Generative modeling of protein sequences

The statistical properties of protein sequences found in nature have been optimized through natural evolution to ensure that they have a full range of physical, chemical, and biological properties to function properly in a complex cellular environment. Therefore, one strategy for enforcing native-like properties in engineered proteins is to use a regularization term that penalizes designs that deviate significantly from the statistical patterns found in nature. To do this, we propose to use a generative model of protein sequences to calculate the regularization term.

The primary model we investigate here is the Evolutionary Scale Model [66], a deep contextual transformer protein language model (TPLM). Within the field of Natural Language Processing, transformer models have become state-of-the-art over Recurrent Neural Networks (RNN). This is because the attention-based mechanism [45] used by Transformers allows the model to contextualize its focus on elements of a sequence it believes are most important. Transformer models not only capture long-range dependencies, but do so without the need to memorize the sequence from beginning to end. On the other hand, an RNN layer requires $\mathcal{O}(n)$ many sequential operations to model such interactions, where n is the length of a sequence. When a sequence's length is less than the dimensionality of its representation learned by either a transformer or RNN (which is typically the case), the transformer's attention layer will be faster and have lower computational complexity compared to the RNN. Additionally, the computation within a transformer layer can be parallelized through the use of multi-head attention. Thus, on a per-layer basis, transformers enjoy higher efficiency compared to RNN's, which in practice can lead to the ability to train on greater amounts of data. The transformer model used in this chapter was trained on over 250 million protein sequences, and has been shown to learn representations for proteins that improve predictive performance over many tasks, including secondary structure and tertiary contact predictions.

In addition to the TPLM, we also evaluate two fold family-specific options— profile HMMs and Markov Random Fields (MRF), as generated by the GREMLIN algorithm [42] (see Appendix A). HMM and MRF models can be learned from known sequences from a given fold family. The primary difference between these models is that HMMs make strong assumptions about the conditional independencies between residues. In particular, GREMLIN identifies and models both sequential and long-range dependencies. Either way, the models encode a joint distribution over residue types at each position in the primary sequence, $P(s(1), ..., s(n))$, which can be used to compute the probability (or related quantities, like log-odds) of given designs. We assume that any design with a high probability or log-odds under the generative model is native-like.

Table 4.1: The wildtype sequences for each of the proteins used in our experiments.

GB1:	`MTYKLILNGKTLKGETTTEAVDAATAEKVFKQYANDNGVD` `GEWTYDDATKTFTVTE`

BRCA1:	`MDLSALRVEEVQNVINAMQKILECPICLELIKEPVSTKCD`
	`HIFCKFCMLKLLNQKKGPSQCPLCKNDITKRSLQESTRFS`
	`QLVEELLKIICAFQLDTGLEYAN`

Spike:	`NLCPFGEVFNATRFASVYAWNRKRISNCVADYSVLYNSAS`
	`FSTFKCYGVSPTKLNDLCFTNVYADSFVIRGDEVRQIAPG`
	`QTGKIADYNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLY`
	`RLFRKSNLKPFERDISTEIYQAGSTPCNGVEGFNCYFPLQ`
	`SYGFQPTNGVGYQPYRVVVLSFELLHAPATVCG`

4.2 Data

4.2.1 Sequences and notation

In our experiments, we use protein sequence data for proteins GB1, BRCA1, and Spike to identify variants that improve a targeted, experimentally measured quantity. In order to run our analyses, it is necessary to have a wildtype sequence for each of these proteins to use as a starting point. We use the wildtype sequences in Table 4.1 for each of these proteins.

When referring to a sequence, the coordinates we use are with respect to the wildtype sequences above. That is, the first amino acid of each sequence corresponds to position 1, and so on. When we refer to a variant sequence, we use the notation '`X#Y`,' where `X` is the wildtype amino acid, `#` is an integer denoting the position, and `Y` is the variant amino acid.

4.2.2 Protein G B1 domain

Protein G is an antibody-binding protein expressed in groups C and G *Streptococcus* bacteria. The B1 domain of protein G (GB1) interacts with the Fc domain of immunoglobulins. We performed our experiments on data generated by Wu et al. [78], who performed saturation mutagenesis at four carefully chosen sites in GB1 in order to investigate the protein's evolutionary landscape. The four chosen residues (V39, D40, G41, and V54) are collectively present in 12 of the protein's top 20 pairwise epistatic interactions, meaning these sites are not just expected to contain evolutionarily favorable variants [79], but also those that are involved in interactions with each other.

The fitness criterion for their study was binding affinity to IgG-Fc. Experimental measurements were obtained for 149,361 out of 160,000 (i.e. 20^4) possible variants at these four loci using mRNA display [80], followed by high-throughput Illumina sequencing. Briefly, this approach to measuring binding affinity works by first creating an input mRNA-protein fusion library from GB1 variants. This input library is then exposed to the GB1 binding target IgG-Fc. Any variant that binds to the target is subsequently sequenced for identification. By measuring the counts of each variant contained in the input library, c^{in}, and output "selected" library, c^{out}, the relative fitness w of the ith variant is calculated as follows:

$$w_i = \gamma \frac{c_i^{\text{in}}}{c_i^{\text{out}}} \tag{4.1}$$

Here, γ is a normalizing factor that ensures the wildtype sequence has fitness 1, and sequences with improved fitness are greater than 1. The range of fitness scores is from 0 to 8.76, with mean 0.08 (see Table 4.2). Only 3,643 sequences ($\approx$ 2.4%) have fitness greater than 1.

4.2.3 BRCA1 RING domain

BRCA1 is a multi-domain protein that belongs to a family of tumor-suppressor genes. It contributes to this function through involvement in homology directed DNA repair, which undoes the genetic instability that underlies cancer development by fixing broken DNA strands. The RING domain of BRCA1 plays a critical role in this process by forming a heterodimer with fellow tumor suppressor BARD1 to constitute an E3 ubiquitin ligase, whose activity is responsible for this tumor suppressing function [81].

Table 4.2: The mean, median, and variance for each scoring metric and each protein type. The final three columns show spearman correlations between each respective score.

Protein	Metric	Mean	Median	Variance	(1)	(2)	(3)
GB1	(1) Fitness	0.08	0.003	0.16	1.0	-0.09	-0.25
	(2) TPLM Log-odds	11.55	11.49	3.80	-0.09	1.0	-0.01
	(3) FoldX $\Delta\Delta G$	9.42	8.39	27.93	-0.25	-0.01	1.0
BRCA1	(1) E3 Ubiquitin Ligase Activity	0.63	0.60	0.17	1.0	0.31	-0.38
	(2) TPLM Log-odds	2.46	2.56	6.06	0.31	1.0	-0.26
	(3) FoldX $\Delta\Delta G$	2.03	0.76	15.83	-0.38	-0.26	1.0
Spike	(1) ACE2 Binding Affinity	0.74	0.87	0.07	1.0	-0.06	-0.63
	(2) TPLM Log-odds	2.62	2.73	0.42	-0.06	1.0	0.11
	(3) FoldX $\Delta\Delta G$	2.51	1.48	14.36	-0.63	0.11	1.0

Starita et al. [82] investigated the functional effect of single site point mutations and deletions at BRCA1 residues 2-304 on E3 ubiquitin ligase activity. Using a phage display assay [83], they determined an E3 ubiquitin ligase activity score for 5154 total variant sequences. The activity was determined by calculating a relative change in abundance of each variant allele as a result of the assay. The scores were normalized so that the wildtype sequence had a score of 1, and nonfunctional sequences 0. After filtering the sequences based on quality, they found scores that ranged from nonfunctional to 2.8.

We used our approach on the Starita data to find BRCA1 RING domain sequences that maximize ubiquitin ligase activity, and thus provide insights into the factors governing sequence-activity relationships. Our experiments were limited to the mutations in residues 2-103 corresponding to the RING domain coordinates that passed a quality filter specified by Starita. In total, this gave us 1388 variant sequences. Activity scores across these residues range from nonfunctional to 2.8, with an average score of 0.63 (see Table 4.2). Only 227 sequences ($\approx$ 15%) had activity scores greater than 1.

4.2.4 Spike protein receptor binding domain

The β-coronavirus SARS-CoV-2 is the virus responsible for the COVID-19 pandemic. The Spike glycoprotein is located on the virus' surface, and plays a critical role in viral transmission. More specifically, the Spike receptor binding domain binds the human cell surface protein ACE2. Binding with ACE2 facilitates entry of the virus into the cell's interior via membrane fusion or endocytosis.

Starr et al. [84] performed a deep mutational scan of the Spike glycoprotein using a yeast display platform to assess the effect of mutations in the receptor binding domain on ACE2 binding affinity. Their approach obtained measurements for each single site mutation within the protein binding domain [85]. In total, they tested 4020 variant sequences, and found binding affinities that ranged from -4.84 to 0.3, where wildtype was normalized to have value 0 and higher scores are better. We removed knock-out variants, leaving all single-site point mutations, and re-scaled their affinities to fall within the unit interval. The re-scaled wildtype had score 0.94, and the average score was 0.74 (see Table 4.2). In total, 398 sequences ($\approx$ 11%) had an affinity greater than wildtype. We used our approach on the Starr data to find Spike receptor binding domain sequences that maximize affinity to ACE2, and thus provide insights into one of the factors governing host-virus interactions.

4.3 Modeling

4.3.1 Evolutionary-based regularization factors

In order to obtain an evolutionary regularization term, we used pre-trained model ESM-1b made available through Rives et al. [66]. This transformer model is trained on all data available through the UniParc [86] database, and thus models *every* protein fold family represented therein. The model is trained using the following masked language model objective:

$$\mathcal{L}_{MLM} = \mathbb{E}_{s \sim S} \mathbb{E}_M \sum_{i \in M} -\log p(s(i) | s_{/M}) \tag{4.2}$$

Each training sequence s is masked at a set of indices M. The model is trained by minimizing the negative log likelihood of each true amino acid $s(i)$ given the masked sequence $s_{/M}$. In other words, the model must learn to use the surrounding sequence as context in order to predict the masked amino acids.

The model itself consists of 33 layers, and contains over 650 million parameters. While in principle the TPLM can be fine-tuned, we simply use it as-is. We use the model to obtain a log-odds score for a given protein design. To obtain this score, we provide as input a variant protein sequence, and the final layer of the TPLM outputs a logit for each possible amino acid at each position in the sequence. By summing over these logits for a given variant sequence, we obtain a TPLM-derived log-odds score. We calculated such log-odds scores for each variant in the available GB1, BRCA1, and Spike data.

In addition to using the TPLM, we also derive what we refer to as a fold-family specific regularization term. We evaluated two options for such models, (i) an MRF model learned using the GREMLIN algorithm [42], and (ii) a profile HMM. We downloaded the profile HMM [67] for the fold family to which GB1 belongs (Pfam id: PF01378) from the Pfam database [87]. We also downloaded the multiple sequence alignment that was used to train the HMM from Pfam, and then used the alignment to train the GREMLIN model. Thus, the GREMLIN and HMM models were trained from the same sequence data. We used these models to compute the log-odds of each design. These log-odds are used as a regularization factor in the Bayesian optimization. The two models make different assumptions about the conditional independencies among the residues in the distribution over GB1 sequences, and thus will output different log-odds scores for the same design, in general. Importantly, these are well-suited to training from limited sets of data, especially when compared to deep models. GREMLIN in particular learns a sparse model precisely to resist overfitting, and is thus better suited to learning from relatively small amounts of data.

4.3.2 Structure-based regularization factors

In order to obtain a structure-based regularization term, we used the FoldX protein modeling Suite. FoldX uses an empirical force field to calculate changes in energy associated with mutations to a protein's amino acid sequence. It contains terms that account for Van der Waal's interactions, solvation energies, energy due to Hydrogen bonding, energy due to water bridges, energy due to electrostatic interactions, and entropy due to dihedral angles and side chain conformations. The specific details of the FoldX force field are included in Schymkowitz et al. [68].

For each protein tested, we obtained expected changes in Gibbs free energy (relative to the wildtype sequence) for each variant sequence. We followed the below protocol to obtain these energy calculations for each set of variant sequences:

1. Download a protein structure from the Protein Data Bank [88]. Table 4.3 shows the structures used for each protein tested.

2. Repair the PDB structure using FoldX's `RepairPDB` command. This fixes structures that have bad torsion angles, Van der Waal's clashes, or total energy by rotating specific residues or side chains into more energetically favorable conformations.

3. Calculate the energy associated with introducing each mutation into the structure using FoldX's `BuildModel` command. Calculations for each mutated sequence are done three times.

The final energy associated with each mutation is given by the average over the three FoldX runs. Mutations that are predicted to improve folding energy will have negative values, whereas positive values indicate an energetically less favorable mutation.

Table 4.3: Protein structural data used for FoldX simulations. Binding partner refers to whether or not the structure includes the protein is in complex with its binding partner.

Protein	PDB ID	Experiment Type	Resolution (Å)	Binding partner?
GB1	2GB1	Solution NMR	NA	No
BRCA1 RING Domain	1JM7	Solution NMR	NA	Yes
Spike PBR Domain	6M0J	X-ray diffraction	2.45	Yes

4.4 Applying regularization terms to directed protein evolution

Bayesian optimization is performed using Gaussian Process (GP) regression as the prior over the unknown fitness or activity function, f. A GP requires a kernel, K, which describes the similarity between sequences s_i and s_j. In our models, we use the squared exponential kernel (also known as the "radial basis function") given by:

$$K_{s_i,s_j} = \exp\left(-\frac{d(s_i,s_j)^2}{2\ell^2} \right) \tag{4.3}$$

where $d(\cdot,\cdot)$ is the Euclidean distance and ℓ the scalar length scale. We used a one-hot encoding of the variants. That is, each residue was assigned a length 20 vector, where each position corresponds to a specific amino acid. The position that corresponds to the residue present in the sequence takes value 1, and all others 0. Hyperparameter θ is optimized while fitting the GP to data by maximizing the log marginal likelihood:

$$\log p(y|\theta) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log\det|K+\sigma^2 I| - \frac{1}{2}y^T(K+\sigma^2 I)^{-1}y \tag{4.4}$$

Since we use the squared exponential kernel in our experiments, length scale ℓ is the only hyperparameter. The term y is a vector of the given property (e.g. fitness) of N sequences, σ^2 the variance of observations, and I is the $N \times N$ identity matrix. Once fitted, the GP encodes a distribution, $\mathcal{P}$, which is used to obtain a posterior mean function $\mu_{\mathcal{P}}(s_i)$ and variance over the unknown function f:

$$\mu_{\mathcal{P}}(s_i) = \mathbb{E}[f(s_i)] = K_{s_i,s}(K+\sigma^2 I)^{-1}y \tag{4.5}$$

$$\mathrm{Var}[f(s_i)] = K_{s_i,s_i} - K_{s_i,s}(K+\sigma^2 I)^{-1}K_{s,s_i} \tag{4.6}$$

$K_{s_i,s}$ refers to the row vector of kernel function values between sequence s_i and all other sequences, denoted by subscript s. Additionally, $K_{s,s_i} = K_{s_i,s}^T$.

4.4.1 Regularized acquisition functions

The GP becomes the argument to an acquisition function, which is used to select sequences for wet-lab screening. The data produced via the screening step are used to update the GP for the next round. We performed experiments that used either the expected improvement (EI), probability of improvement (PI), or upper confidence bound (UCB) criteria as the acquisition function. Two versions of each acquisition function were considered: (i) the standard version, which is often used in Bayesian optimization, and (ii) a regularized form. The standard forms of EI, UCB, and PI are given by:

$$\mathrm{EI}(s_i;\mathcal{P}) = \mathbb{E}_{\mathcal{P}}\big[\max\big(0, f(s_i) - \mu_{\mathcal{P}}(s^+)\big)\big] \tag{4.7}$$

$$\mathrm{PI}(s_i;\mathcal{P}) = P\big(f(s_i) > \mu_{\mathcal{P}}(s^+)\big) \tag{4.8}$$

$$\mathrm{UCB}(s_i;\mathcal{P}) = \mu_{\mathcal{P}}(s_i) + \beta\sigma(s_i) \tag{4.9}$$

where s^+ is the location of the (estimated) optimal posterior mean, β a constant scaling factor (0.05 in our experiments), and $\sigma(\cdot)$ the standard deviation.

We also evaluated a *regularized* form of each acquisition function by scaling the standard version by a design-specific scaling factor, $\mathcal{F}(s;\mathcal{P})$. In our experiments, $\mathcal{F}$ refers to the evolution-based log-odds score obtained by either a TPLM, MRF, or profile HMM, or the structure-based $\Delta\Delta G$ calculated be FoldX, as described previously. Our regularized EI, PI, and UCB are defined as:

$$\mathrm{EI}_{\mathcal{F}}(s_i;\mathcal{P}) = \mathrm{EI}(s_i;\mathcal{P})\mathcal{F}(s_i) \tag{4.10}$$

$$\mathrm{PI}_{\mathcal{F}}(s_i;\mathcal{P}) = \mathrm{PI}(s_i;\mathcal{P})\mathcal{F}(s_i) \tag{4.11}$$

$$\mathrm{UCB}_{\mathcal{F}}(s_i;\mathcal{P}) = \mathrm{UCB}(s_i;\mathcal{P})\mathcal{F}(s_i) \tag{4.12}$$

We will demonstrate that this small modification to the acquisition function results in a substantial shift in the designs discovered via ML-assisted DE towards native-like designs, as expected.

4.4.2 Directed evolution with machine learning and *in silico* traditional approaches

Our experiments contrast the performance of 'standard' ML-assisted DE (i.e. non-regularized) to the regularized version. We also compare the results to simulated forms of 'traditional' DE (i.e. without ML), as was also done in Wu et al. [64]. Since the BRCA1 and Spike protein data only include single-site mutations, we limit this analysis to the more exhaustive GB1 data. Specifically, we simulated both the single mutation walk and recombination versions of DE (see Figure 4.1). We note that the single mutation walk approach is deterministic, given the starting sequence. With the single step, we start each trial with a randomly chosen sequence from the GB1 variant library. At each of positions 39, 40, 41, and 54, we observe the experimentally determined fitness values for all possible single-residue mutations. Having observed these mutations, we then fix in place the single-residue mutant which has the highest fitness. With this residue fixed, we then repeat this procedure for the remaining unfixed residue positions. Continuing in this manner, the trial ends when all residues have been fixed. All observed fitness values within a trial thus represent a DE determined fitness function approximation.

For the recombination method, we mimic saturation mutagenesis experiments by starting with n randomly chosen sequences from the GB1 variant library. From these, we identify the top three sequences that have highest fitness (as was done in Wu et al. [64]), and use these sequences to perform recombination. A recombinant library is simulated *in silico* by computing the Cartesian product $S_{39} \times S_{40} \times S_{41} \times S_{54}$, where the set S_m refers to the variant residues found at position m among the three highest fitness sequences in the initial random library. The resulting list of 4-tuples defines the recombinant library.

Here, the DE fitness function approximation is given by observing fitness values for the n starting sequences as well as the recombined sequences.

4.5 Results

In this section, we report the results of five approaches to performing DE:

(i) single mutation walk (see Figure 4.1-top)

(ii) recombination (see Figure 4.1-bottom)

(iii) Bayesian optimization using standard acquisition functions (Eq. 4.7-4.9), denoted by:

  a) GP+EI

  b) GP+PI

  c) GP+UCB

(iv) Bayesian optimization using evolution-based regularized acquisition functions with TPLM-derived log-odds, denoted by:

  a) GP+EI+TPLM

  b) GP+PI+TPLM

  c) GP+UCB+TPLM

(v) Bayesian optimization using structure-based regularized acquisition functions with FoldX-derived $\Delta\Delta G$ values, denoted by:

  a) GP+EI+FoldX

  b) GP+PI+FoldX

  c) GP+UCB+FoldX

The regularized versions of each acquisition function used in (iv) and (v) are given by Eq. 4.10-4.12. For simplicity, we focus our analysis on a subset of all experiments that we conducted. We include additional results briefly mentioned within the text in Appendix A.

Each method was allowed to screen (e.g. obtain fitness values for) a total of 191 variants. This number was chosen to be similar to the number of sequences screened by the deterministic single mutant walk so that each method had similar experimental burden. Each model was initially trained on 20 randomly selected sequences. The small number of initial sequences simulates the scenario where the available fitness data is limited, prior to DE. The Bayesian optimization methods selected the top 19 sequences during each acquisition round. Each model is then updated with the experimentally measured fitness values for the chosen batch of 19 sequences, and this process is repeated for 9 batches (i.e. 20 initial sequences plus 9 batches of 19 designs per batch, giving $20 + 19 \times 9 = 191$ variants selected). We refer to a complete set of variant selection batches as a *trial*. We performed 100 total trials with each selection strategy with different random initial starting sequences. 20% of the data were held out for testing purposes (see Appendix A).

4.5.1 ML-assisted DE outperforms traditional DE

Traditionally, DE techniques aim to identify sequences that score highly in one property. In Figure 4.3 we demonstrate that ML-assisted DE outperforms simulated traditional approaches (i.e. single-mutant walk and recombination) when optimizing GB1 with respect to fitness. We observe this trend across all three forms of acquisition function tested (EI, PI, or UCB). On average, ML-assisted techniques (regularized and unregularized) identify a variant with fitness 7.22 (EI), 7.27 (PI), and 7.08 (UCB), whereas simulated traditional approaches identify a variant with fitness 4.97. The single mutant walk procedure finds a variant with maximum fitness 5.22, whereas recombination yields a

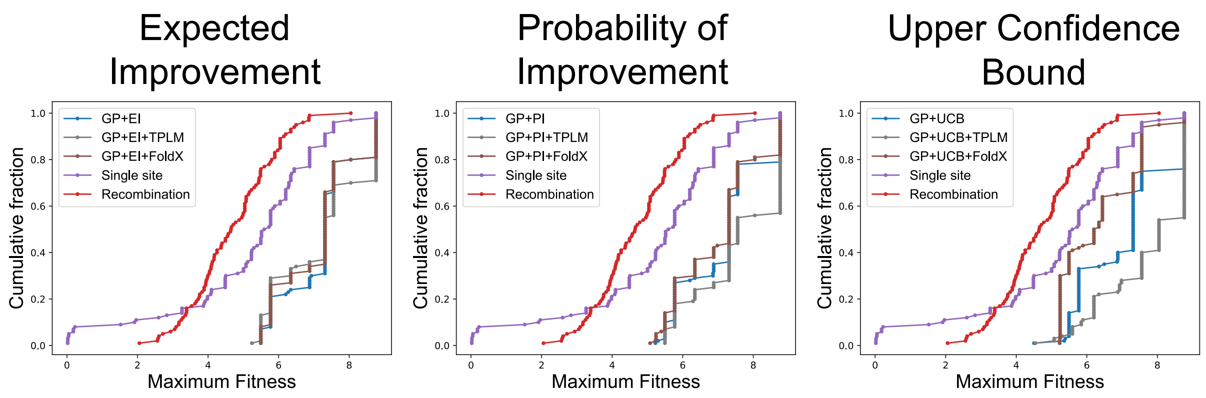


Figure 4.3: **ML-assisted Directed Evolution techniques identify high fitness GB1 variants more frequently than simulated traditional DE approaches.** Shown are the fraction of trials (y-axis) that reach less than or equal to a specified fitness (x-axis), where the selection criterion was either a simulated traditional DE approach, or standard or regularized EI, PI, and UCB was the acquisition function. **(Left) Expected Improvement**. The cumulative-weighted average fitness values are 7.25 for GP+EI+TPLM, 7.24 for GP+EI, and 7.16 for GP+EI+FoldX. **(Middle) Probability of improvement**. The cumulative-weighted average fitness values are 7.62 for GP+PI+TPLM, 7.17 for GP+PI, and 7.03 for GP+PI+FoldX. **(Right) Upper confidence bound**. The cumulative-weighted average fitness values are 7.76 for GP+UCB+TPLM, 7.10 for GP+UCB, and 6.38 for GP+UCB+FoldX. **(All)** The traditional single step and recombination approaches select variants with cumulative-weighted average fitness values of 5.22 and 4.71, respectively.

variant with maximum fitness 4.71. Overall, we find that ML-assisted DE thus improves upon traditional approaches in designing high fitness GB1 variants by an average of 45%. This result is consistent with the findings in Wu et al. [64]. In Figure A.1, we show that evolution-based regularization via GREMLIN and profile HMMs are also able to improve upon traditional DE techniques on the same GB1 variant selection task.

4.5.2 Structure-based regularization usually leads to better designs

Next, we investigated the effects of regularization and choice of acquisition function in the context of ML-assisted DE. Figure 4.4 shows the results of experiments on each protein using various regularized and unregularized acquisition functions. Overall, we find that structure-based regularization usually leads to better designs, and almost never hurts. The exceptions involve GB1. We note that the GB1 structure used in our experi-

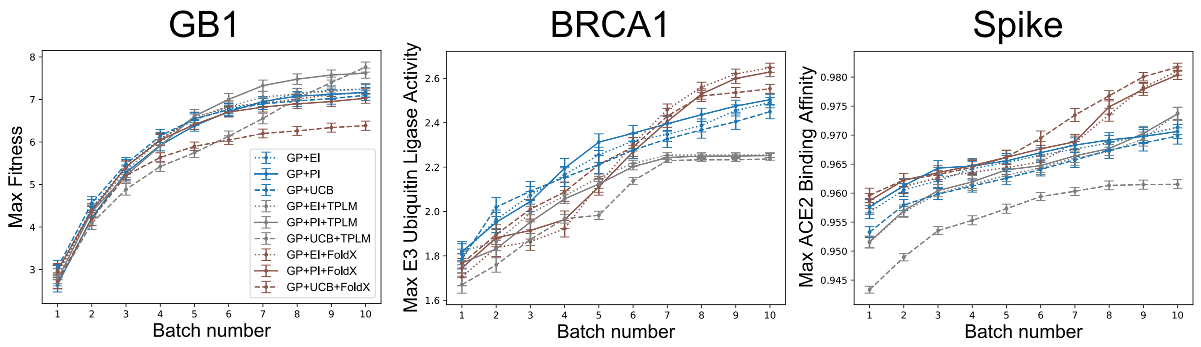


Figure 4.4: **Regularization leads to better designs.** Shown are the cumulative per batch scores for each protein averaged (± 1 SEM) over 100 trials. GP models were initialized with 20 randomly chosen sequences, and each batch consisted of 19 selected variants. **(Left)** GP+UCB+TPLM selected the GB1 variant with highest average fitness (7.76), **(Middle)** GP+EI+FoldX selected the BRCA1 variant with highest average E3 ubiquitin ligase activity (2.65), and **(Right)** GP+UCB+FoldX selected the Spike variant with highest average ACE2 binding affinity (0.98).

ments does not include the antibody, and so FoldX does not have a complete picture of the system being optimized.

Figure 4.4 also shows that the benefits of structure-based regularization vary according to the experimental budget. For example, if one were only able to perform 4 rounds, then the unregularized acquisition works better for BRCA1, but not for the other two proteins. Still, aside from the previously mentioned exception with GB1, structure-based regularization never hurts, given a sufficient number of rounds.

4.5.3 Evolutionary-based regularization is unreliable

If a structure model is not available, it is natural to consider an evolutionary-based regularization term. That is, one based on a sequence model, like a transformer. However, we find that evolutionary-based regularization via the as-is ESM-1b TPLM is unreliable. As seen in Figure 4.4, it does very well for GB1— outperforming both structure-based and the unregularized methods, but it underperforms for BRCA1 and Spike. This variability is perhaps expected, since a sequence model is obviously just an abstraction of a molecule. The TPLM apparently captures enough of the relevant information for the

GB1 design task, but does not for BRCA1 or Spike. We also ran experiments that used GREMLIN and HMM regularized methods, but found that they did not perform much differently than unregularized methods (Figure A.2-top). Together, these results suggest that one should obtain more consistent (in some cases, better) results using unregularized or structure-based ML-assisted DE compared to the evolutionary-based regularization methods we have tested.

4.5.4 Correlation among traits

Naturally, the inclusion of a regularization term will bias variant selection towards designs that score favorably according to the regularization criteria, in addition to the objective value. This is seen clearly in Figure 4.5, where the grey curves associated with the evolutionary bias achieve high log odds (top row), and the brown curves associated with the structural bias achieve low $\Delta\Delta G$ values (bottom row), as intended. The fact that the corresponding objective values are also high (see Figure 4.4) simply indicates that the regularization terms generally do no harm. What is unexpected, however, is the fact that the blue and brown curves in the top row *also* rise by varying amounts, and the blue and grey curves in the bottom row *also* fall by varying amounts, even though those curves correspond to acquisition strategies that do not consider the quantity plotted on the y axis. This behavior reveals that the objective and regularization values carry some information about unmeasured traits. GREMLIN and HMM regularized acquisition functions reveal similar patterns (Figure A.2-bottom)

4.5.5 Evolution and structure-based regularization promotes site-specific exploration of unexplored sequence space

Thus far, we have characterized variants selected by each ML-assisted DE technique in terms of their average fitness (GB1), measured activity (BRCA1), or binding affinity (Spike). We now consider the residue-specific behavior of choices made under each

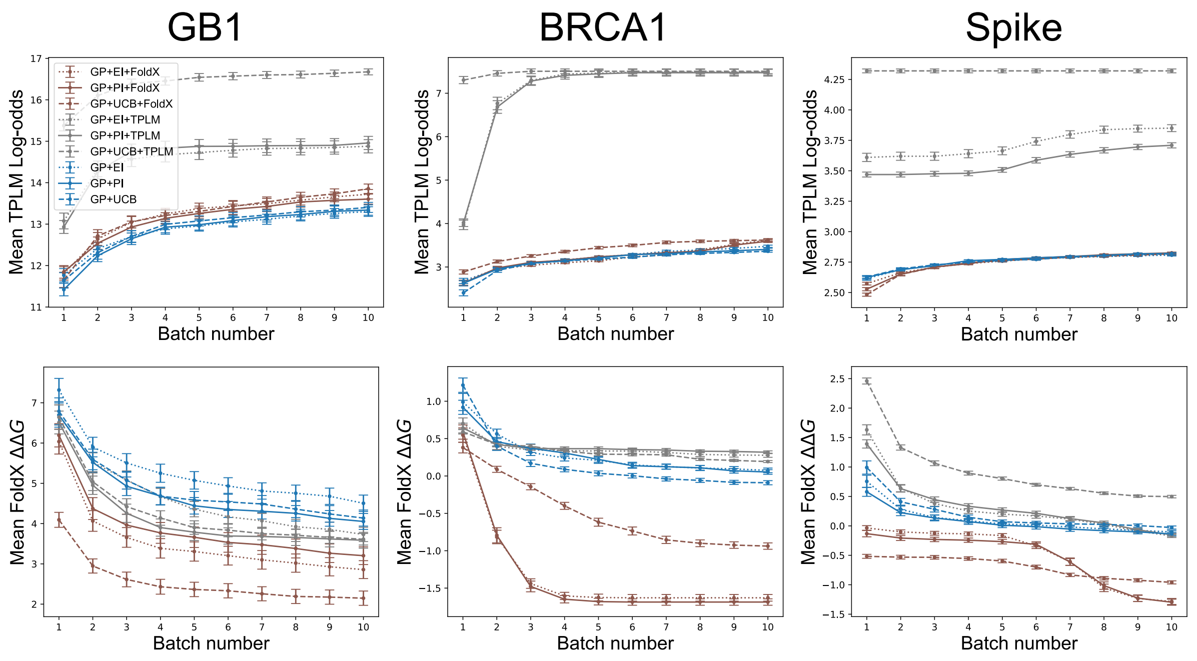


Figure 4.5: **Evolution and structure-based regularization biases variant selections towards those that score favorably under multiple criteria.** Shown are the regularization scores for variants selected for GB1 **(Left)**, BRCA1 **(Middle)**, and Spike **(Right)** under each selection criterion. As expected, variants selected by TPLM-regularized methods have higher log-odds under the TPLM than those selected from non-TPLM regularized methods **(Top)**. Similarly, variants selected by FoldX regularized methods have lower $\Delta\Delta G$ values than those selected by non-FoldX methods **(Bottom)**. The figures *also* show that TPLM-regularized methods tend to improve FoldX scores, and that FoldX-regularized methods tend to improve log-odds, indicating that there is some correlation between log-odds and thermodynamic stability.

model. Figure 4.6 shows residue-specific entropy of variant selections with the best performing model for each protein. A high entropy block (dark blue) indicates that the method selects many different residue types at that position within a given batch. Low entropy blocks (light blue) indicate that the method selects only few residue types. The relative entropy of selections thus provides a sense of how the model explores sequence space, as well as the confidence the model has that a particular variant residue is informative.

Qualitatively, we notice that regularized methods for each protein have darker shading than their unregularized counterpart. This indicates that regularized methods explore more variant types at specific positions in each protein. With GB1 (Figure 4.6-left),

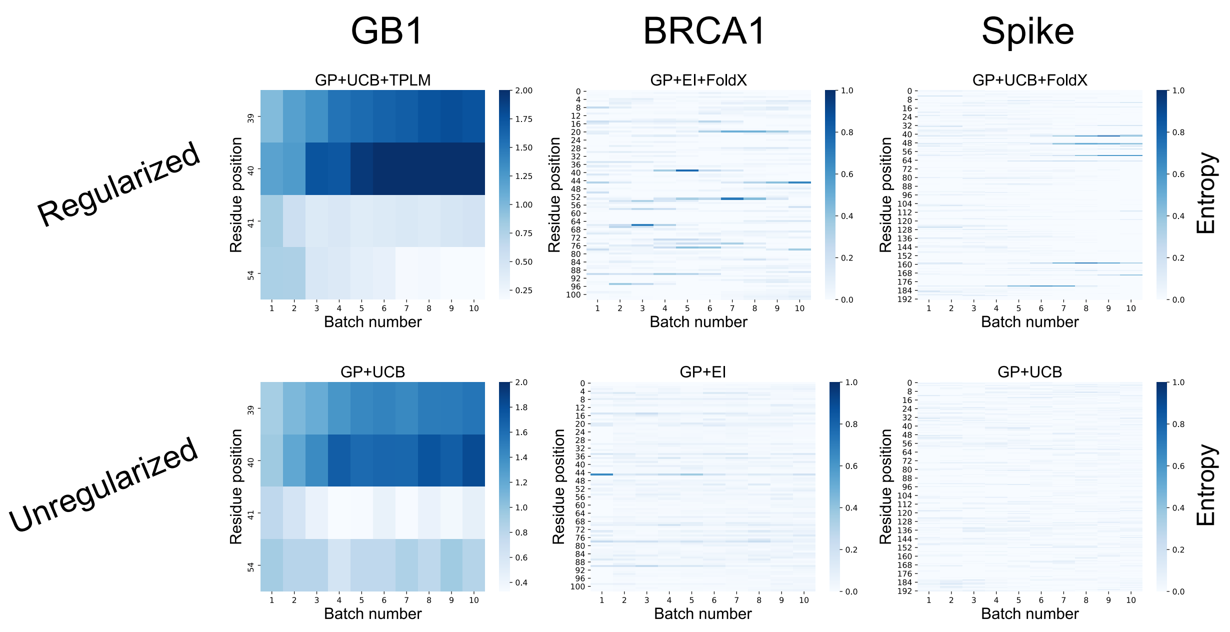


Figure 4.6: **Bayesian selection techniques quickly identifies informative sequence patterns.** Shown are the per-batch average position-specific entropy of variant selections under the top scoring model for each protein using a regularized **(Top)** or unregularized **(Bottom)** acquisition function. Lighter squares denote low entropy decisions, meaning the model selects among fewer residue types at that position in that batch.

positions 39 and 40 have highest entropy for regularized and unregularized methods. The unregularized method has the least entropy at position 41, indicating that it has highest certainty at this position, whereas the regularized method has highest certainty at position 54. With both BRCA1 and Spike (Figure 4.6-middle/right), there is generally much more light shading throughout the sequence, regardless of whether or not there is regularization. This is to be expected due simply to the larger number of positions that could be mutated using these protein data (recall that while the GB1 data includes all pairwise variants across 4 positions, these data include all single-site mutations across regions larger than 100 sequences each). Still, the regularized methods contain regions of darker shading, indicating a greater level of site-specific exploration. Given that the only difference between the methods shown for each protein is the presence or absence of regularization, it is the evolution or structure-based regularization that must drive this increased exploration at targeted positions.

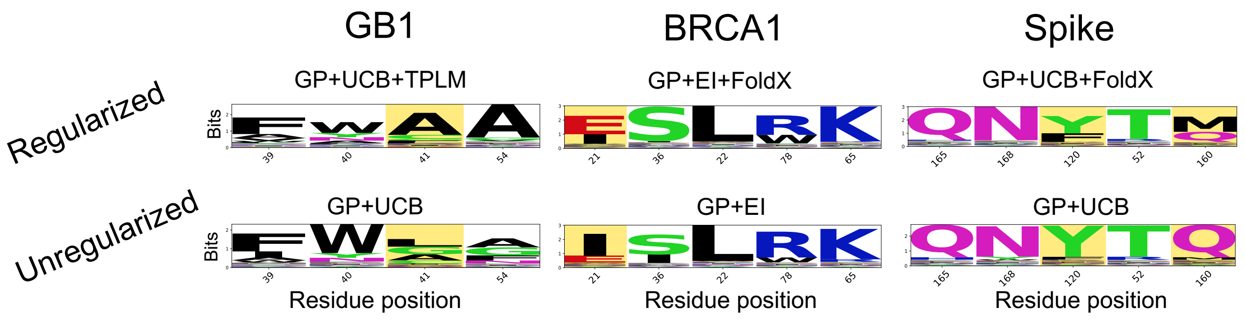


Figure 4.7: **Evolutionary and structure-based regularization biases variant selection towards sequences with desirable properties.** Shown are sequence logos for the best performing variant selection method along with their unregularized counterpart. All four variant residue positions are shown with the GB1 protein **(Left)**, whereas the positions that correspond to variants with the top five true activity/binding affinity scores are shown for BRCA1 **(Middle)** and Spike **(Right)**. Highlighted residues denote notable distinctions between the regularized and unregularized sequence selections.

In Figure 4.7, we show sequence logos obtained from the same models whose entropy are shown in Figure 4.6. With GB1 (Figure 4.7-left), the regularized method selected consensus sequence {V39F,D40W,G41A,V54A}, whereas the unregularized method selected {V39F,D40W,G41L,V54A}. While these sequences are similar, the single different amino acid selected at position 41 is meaningful in terms of the overall fitness of the design— {V39F,D40W,G41A,V54A} is the highest fitness variant in the data (8.76), whereas {V39F,D40W,G41L,V54A} ranks 443rd (3.66). With respect to BRCA1 (Figure 4.7-middle) and Spike (Figure 4.7-right), the sequence logos show amino acid selections for the positions where the top five scoring variants within each data set are located. For both of these proteins, the consensus sequences selected by the unregularized methods corresponds to the wildtype sequence. However, the best regularized method used for both proteins arrived at one consensus variant. With BRCA1, this corresponds to I21E, which is the highest scoring variant in the BRCA1 data. Additionally, the Spike variant Q120M selected by a FoldX regularized method has ACE2 binding affinity 0.18, which is the sixth highest scoring variant in the data set. Additionally, while Y is the consensus selection at position 120 for both models shown, we see that more trials that were regularized by FoldX selected variant Y120F, the third highest scoring variant, compared to the unregularized methods. Thus, regularized ML-assisted DE better identified the top scoring variant in the GB1 and BRCA1 data, and the third and sixth highest scoring variant in the Spike protein data compared to unregularized methods.

4.6 Discussion

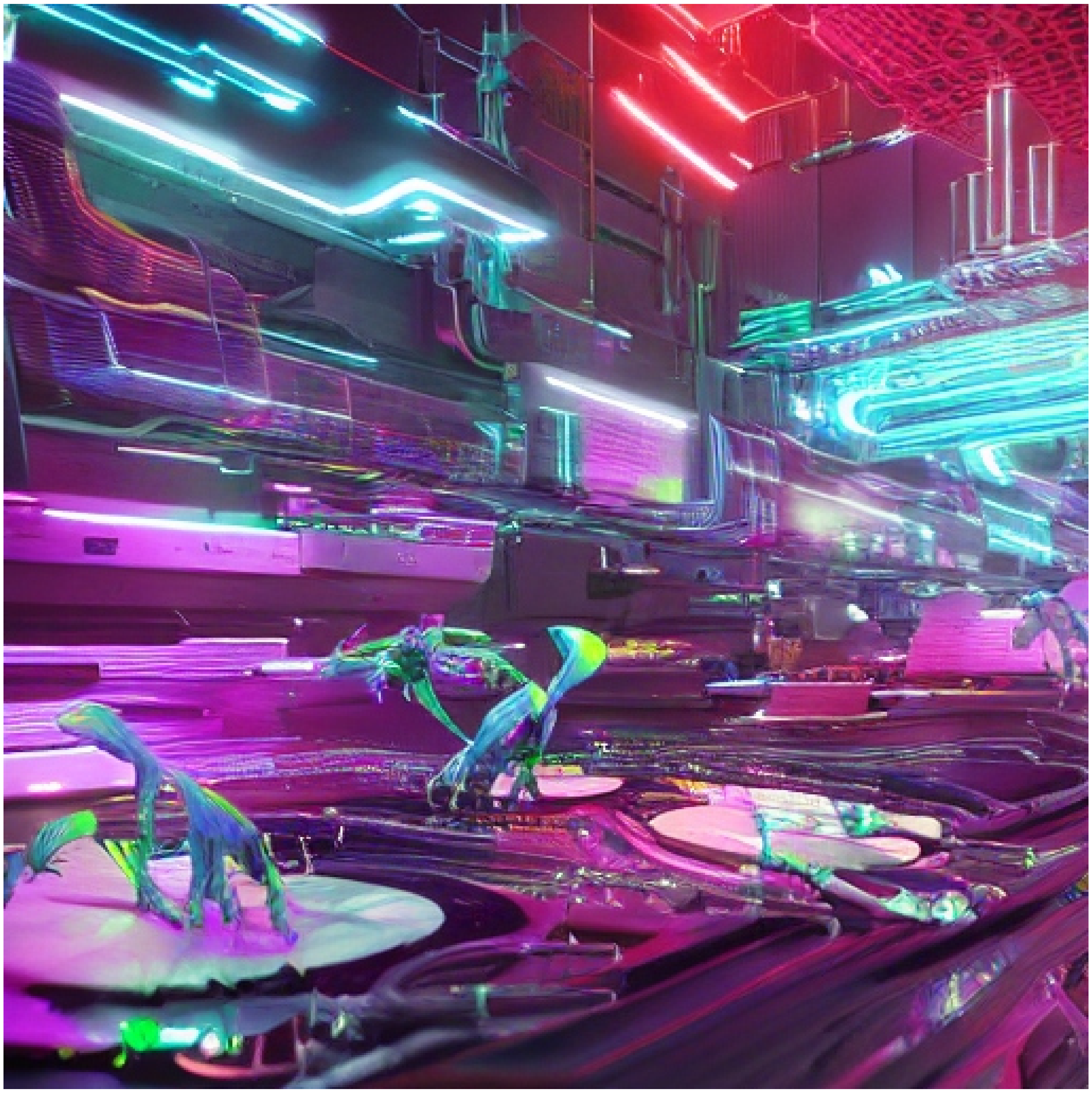
Our work extends ML-assisted DE via Bayesian optimization [74] by incorporating a regularization term into the acquisition function. The regularization term is intended to prevent the algorithm from optimizing the target property at the expense of unmeasured, but nevertheless important properties (e.g. solubility, thermostability, etc.). The results on the GB1 design task demonstrate that the inclusion of a regularization term can decrease the number of rounds needed to find high-fitness designs, relative to the unregularized version (Figure 4.3), but the difference in performance does depend on which acquisition criterion is used (EI, PI, or UCB), and which regularization term is used (evolutionary or structure-based).

When our method is applied to more proteins, a clearer picture emerges. Given a sufficient number of rounds, structure-based regularization usually produces better designs, and only did harm in one configuration (GP+UCB+FoldX on GB1). In contrast, evolutionary-based regularization terms were seen to be unreliable; it only helped in two configurations (GP+PI+TPLM and GP+UCB+TPLM on GB1), but did poorly on BRCA1 and one configuration of Spike (GP+UCB+TPLM). Taken together, these results suggest that structure-based regularization using either EI or PI is beneficial or, at worst, neutral.

The one protein for which structure-based regularization using either EI or PI does not produce better designs than the unregularized version was GB1. Here, we note that GB1 was the one protein where the structure did not include the binding partner (see Table 4.3). That is, FoldX was not given relevant information, and so its predictions are less helpful as a guide during optimization. The structures used for the experiments on BRCA1 and Spike, in contrast, include the binding partner. It makes sense in this circumstance that FoldX will better determine the stability induced by a variant when in the presence of the binding partner— it simply has more relevant information.

As previously noted, evolution-based regularization by the TPLM is less reliable than its structure-based counterpart. One consideration that may contribute to this observation are the differences between the three data sets. For one, GB1 has the shortest sequence length (56) compared to BRCA1 (103) or Spike (193). It may be that the TPLM better captures interactions between sequence elements within smaller protein regions. The GB1 data is also more exhaustive than that for BRCA1 and Spike in that it contains all pairwise variants from four specific GB1 residues, whereas the others are limited to single-site mutations. More specifically, the four GB1 sites that were varied were chosen because they were predicted to be among the most involved in epistatic interactions [79]. That is, these residues are expected to contribute to long-range interactions between residues in GB1. The TPLM model we used was chosen because it is effective at encoding long-range dependencies within a protein sequence [66]. Thus, it may be that the GB1 data set is particularly well-suited to demonstrate the strengths of the TPLM. Another consideration is the TPLM itself. Recall that the TPLM uses unsupervised training across over 250 million different proteins to learn an effective representation for protein sequences. The model is thus very general, and not customized for a particular fold-family. Others have recently shown that a TPLM can be fine-tuned for a given fold family, and that a fine-tuned model can perform better compared to the general model on predictive tasks related to the tuned protein family [89]. In Section 5.3.4, we will describe some of our own fine-tuning experiments within the context of a related protein engineering objective.

Finally, our results also demonstrate the benefits of a Bayesian approach to ML-assisted DE, as opposed to the approach introduced in Wu et al. [64]. When using the GB1 data (the only data set for which a head-to-head comparison is possible), we find that a Bayesian approach only required 191 fitness value acquisitions (20 initial observations plus 9 batches of 19) to identify designs with high fitness values. In contrast, the experiment in Wu et al. [64] required 470 initial observations plus a single batch of 100 to find similarly fit designs. This is a 67% reduction in the number of sequences tested, which demonstrates the merits of Bayesian optimization in this context. The acquisition functions used in Bayesian optimization make a trade-off between exploration and exploitation of the domain, much like DE itself. Thus, ML-assisted DE via Bayesian Optimization effectively uses two forms of exploration and exploitation (one computational, one experimental). In contrast, the computational approach used in Wu et al. [64] is effectively pure exploitation. It is known that optimal regret bounds require a combination of exploration and exploitation [52], which may explain the advantage of Bayesian optimization in this context. As shown in Figure 4.6, adding a regularization term to the acquisition function changes *where* the algorithm chooses to explore. This is seen by the differences in the entropy at select positions between the regularized and unregularized approaches. The regularized version tends to concentrate exploration at select residues, whereas unregularized methods select positions more uniformly. Notably, this trend is apparent in all regularized methods and all acquisition function types (Figures A.3-A.5). This difference in behavior leads to subtle changes in the final designs, as shown in Figure 4.7, but regularization tends to produce the better design (Figure 4.4).



Chapter 5

Identifying Promising Sequences for Protein Engineering using a Deep Transformer Protein Language Model

In the previous chapter, we found that incorporating protein structure information into an optimization routine helped us to guide Directed Evolution experiments towards designs with desirable properties. This suggests that further work aimed towards reducing experimental burden during protein engineering campaigns could benefit by leveraging information about physical protein interactions. In this chapter, we again focus on the ESM-1b transformer model. However, we now focus on a capability of ESM-1b that we had previously neglected that more directly accounts for molecular interactions. By treating ESM-1b's attention mechanism as a protein-protein interaction map, we show how to calculate a Promise Score that measures the promise of a sequence within the context of a protein engineering campaign.

5.1 Introduction

5.1.1 Protein engineering

As described in the previous chapter, protein engineering is a rapidly evolving field that aims to develop novel protein sequences with useful properties. Protein engineers can be found across many scientific domains, so these useful properties can have equally far-reaching applications, including those that are therapeutic [90, 91, 92, 93], industrial [94, 95, 96], and environmental [97, 98]. Regardless of application, a common issue among protein engineering pipelines is the sheer number of experiments required to identify these useful sequences. The combinatorics associated with altering the amino acid composition at different sites within a protein sequence quickly escalates beyond what is experimentally feasible. Developing general-purpose methods that can help protein engineers with experimental decision making is thus important and necessary.

Protein engineering pipelines do not fit within a single mold. Just as the applications are varied, so too are the experimental requirements and focal points. Whereas last chapter we focused specifically on direct evolution-based approaches, in this chapter we refer to protein engineering in a more general sense. While some workflows may focus specifically on a particular objective, many are multi-faceted and include different stages that each involve their own rounds of sequence selection or experimental decision making. A general workflow may be given as:

$$\texttt{Protein Discovery} \rightarrow \texttt{Protein Optimization} \rightarrow \texttt{Model Building}$$

As many protein engineering pipelines include at least one of these components, we will now briefly define each of these stages, and describe the types of experiments and decisions that are made within each.

Protein discovery. While the design of novel protein sequences lies at the heart of most protein engineering objectives, in many settings there is an initial discovery phase.

Perhaps most notably, this includes the design of antibody-based therapeutics [99, 100, 101]. Antibodies (Abs) play an important role in initiating an immune response by specifically targeting and binding targeted antigens. An Ab's specificity for and ability to bind antigens depends on the sequence identity at three complementarity-determining regions (CDR), as residues in these CDR will bind to the surface of an antigen. The Ab residues involved in antigen binding are known as paratopes, and those on the antigen as epitopes. Understanding the interactions between paratopes and epitopes is at the heart of many Ab therapeutics design problems [102, 103]. Recently, a similar class of proteins called nanobodies (Nbs) have also been used for their therapeutic properties [104, 105]. Nbs are shorter than Abs, which makes them easier and cheaper to produce, though are only found naturally in camelid species, which can make them harder to initially collect (whereas Abs can be obtained from common lab mice). Structurally, Abs and Nbs form Y-shaped proteins, where the CDR are located within hypervariable tips. The hypervariability means that sequences collected from an animal sample will have high diversity, and only a subset of the sequences collected will strongly bind a specified antigen target. Collecting this repertoire of diverse sequences is the crux of the protein discovery phase. These Abs or Nbs will undergo assays that quantify their ability to bind a specified antigen. From this, a panel of the best sequences, or *leads*, are identified and used in downstream engineering objectives that take advantage of the leads' strong therapeutic properties. Finding ways to identify strong leads while minimizing the required wet-lab experimentation is thus an important undertaking.

Protein optimization. Optimization is an integral component of all protein engineering campaigns. It is a stage where protein sequences are modified with the goal of identifying those that possess desirable properties. The optimization typically involves making a relatively small number of changes to a given parental protein sequence. For example, in *de novo* protein design [106, 107], computational models are used to propose novel sequences that are expected to have certain structural properties. These proposed sequences are typically experimentally refined (i.e. optimized) to ensure they best match the desired properties. More commonly, naturally obtained sequences are optimized directly, such as leads selected from a protein discovery campaign. In any case, there are many types of experimental technologies that may be used to perform the sequence optimization. Traditionally, these approaches involve random mutagenesis at one or more selected sites [63]. Site-specific mutagenesis [108, 109, 110, 111] and deep-mutational scanning [112] further allow greater throughput and specificity as to where in a protein's sequence edits should be made. Emerging approaches based on CRISPR/Cas9 [113, 114] continue this trend. Given the costs associated with these experiments as well as the exponential number of *potential* edits that can be made to any protein sequence, it is important to identify ways that efficiently and effectively select at which sites to perform these mutagenesis experiments.

Model building. In conjunction with the novel sequences that may be designed and synthesized during a protein engineering campaign, it is often desirable to also obtain a structural model of the system under investigation. The model serves as a tool that can be used to better understand physical properties of a protein, such as how it folds or interacts with other proteins. Traditionally, these models have been obtained through experimental means, such as x-ray crystallography [115]. While x-ray crystallography can provide a high-resolution structure of a protein or protein complex, it requires very specific conditions that may preclude its applicability to certain systems. It also only provides a static, or fixed model for a protein's structure, whereas most interactions worth modeling depend on many moving parts. *Computational* models built on machine learning and artificial intelligence bridge this gap between the generalizability and applicability of purely experimental model building approaches. With respect to protein structure prediction, AlphaFold [116] is able to predict any protein's 3D structure, and is shown to have highest accuracy on the benchmark CASP challenges [117]. More generally, computational models can be used to make predictions on any protein property.

In some settings, these models can even form a feedback loop, where predictions by a model are used to select experiments, and the experimental results are used to update the model [64, 22, 21]. Whether or not a model is used in-line with running protein engineering experiments, most protein engineering pipelines generate large amounts of data, which makes them well-suited for training machine learning models at the conclusion of all experimentation.

5.1.2 Attention and transformer protein language models

Protein engineers look to design novel proteins that have certain desired functions or structure. It is has been shown that these biological features are represented through the statistical dependencies between evolutionarily selected sequences that are found in nature [118, 119]. Finding ways to encode and exploit this evolutionary information could help to guide the sequence design choices made by protein engineers.

In the previous chapter, we introduced the deep transformer protein language model ESM-1b that was designed just for this purpose. We used the ESM-1b logits to calculate a log-odds score that served as a regularization factor within a Bayesian optimization-based directed evolution routine. In this chapter, we again utilize ESM-1b, but instead focus more squarely on the component of the model that is most responsible for its power— attention.

ESM-1b's attention mechanism [45] allows the model to capture the statistical dependencies across all sequences, as it accounts for all pairwise interactions between each position in a sequence. Attention has been shown to align strongly with protein contact maps, as well as target specific protein binding sites [120]. The resulting self-attention map, or matrix of all pairwise interactions, can thus be interpreted as a protein-protein interaction map.

Since ESM-1b is trained on many millions of protein sequences spanning practically all known protein families, it learns many general patterns that exist across all protein sequences. When model building, it is typical to want to learn how to predict a specific task (e.g. how proteins from a specific family binds a specific target). Transformer models can provide a good starting point when obtaining these specific models. Transformers have been shown to be effective *few-shot learners* [121], meaning they can be used to predict specific tasks when given only few labeled examples. Developing exact mechanisms to train such models is an active area of research within NLP, and includes methods such as knowledge distillation [122], fine-tuning [123], and transfer learning [124].

In this work, we show how to use the deep transformer protein language model ESM-1b to help select experiments associated with the protein discovery and protein optimization phases of protein engineering pipelines. We then demonstrate how to apply the computational model building techniques of fine-tuning and transfer learning using the Transformer in each of these settings.

5.2 Materials and methods

5.2.1 Encoding a sequence and its protein target as input for ESM-1b

We use the deep Transformer Protein Language Model ESM-1b [66] to jointly model interactions between a protein sequence, s_{binder}, and its protein binding target sequence, s_{target} (Figure 5.1). For our purposes, ESM-1b is a function $\texttt{ESM}(\cdot)$ that takes as input a length n (tokenized) protein sequence, s, and outputs a tuple $(\mathcal{R}, \mathcal{L}, \mathcal{A})$:

$$(\mathcal{R}, \mathcal{L}, \mathcal{A}) = \texttt{ESM}(s) \tag{5.1}$$

Here, $\mathcal{R} \in \mathbb{R}^{1280}$ is the sequence *representation*, or feature-rich learned encoding obtained from the final Transformer block. $\mathcal{L} \in \mathbb{R}^{n \times k}$ contains the *logits* returned by the Transformer model. These logits can be used to assign an evolutionary probability to each of k possible tokens (the 20 standard amino acids, nonstandard amino acids, and special tokens internal to ESM-1b) at each position of the input sequence. Finally,

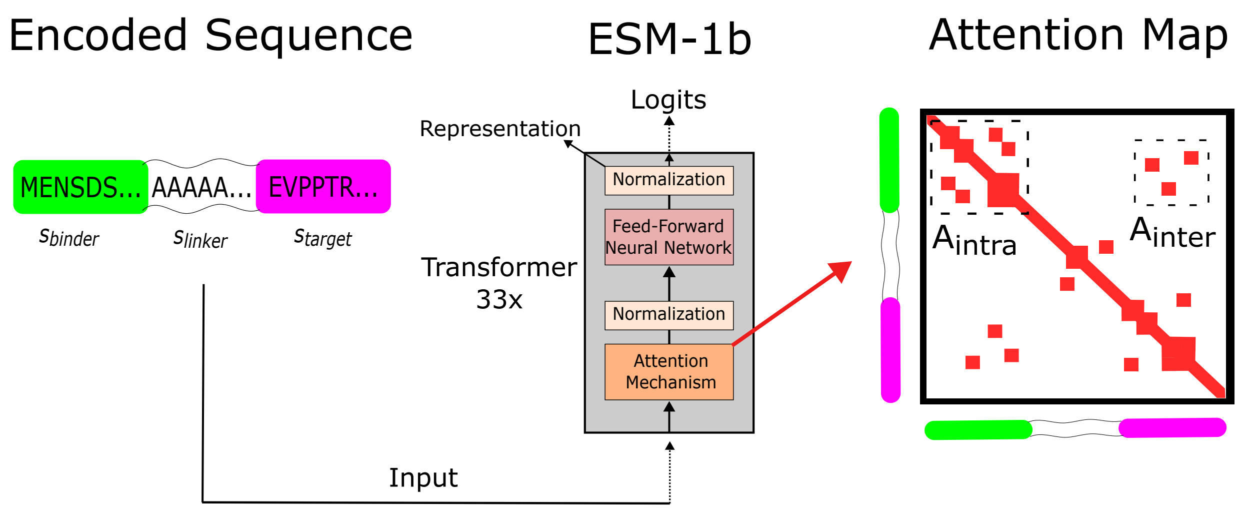


Figure 5.1: We use a deep Transformer Protein Language Model to identify *promising* sequences within protein engineering campaigns. Using Nb discovery and protein optimization as two test cases, we introduce the PROMISE SCORE, and show how it can help address the needle-in-the-haystack issue of identifying protein sequences with targeted properties. To calculate a PROMISE SCORE, we specifically encode a sequence and its binding target through concatenation with a linker, and use this adjoined sequence as input to the pre-trained ESM-1b model. This Transformer model uses an attention mechanism that captures all expected pairwise interactions between residues within the input sequence through an attention map. We use this map to quantify intermolecular and intramolecular interactions within each protein sequence. The PROMISE SCORE reflects these interactions, meaning stronger sequences should tend to have higher PROMISE SCORES than weaker sequences.

$\mathcal{A} \in \mathbb{R}^{n \times n}$ is the self-attention map returned by the final Transformer block's attention mechanism. This attention map is analogous to a predicted protein-protein interaction map, and is the principal output from ESM-1b that we use.

We want to use ESM-1b to identify interactions between s_{binder} and s_{target}. To do this, we construct input sequence s to encode both of these sequences jointly. In our experiments, we show that a simple concatenation scheme is effective. We encode each input sequence s as:

$$s = \texttt{concatenate}(s_{binder}, s_{linker}, s_{target}) \tag{5.2}$$

where s_{linker} is a sequence of one of the following:

1. **Alanine.** An amino acid. A poly-alanine linker represents a flexible loop in s.

2. **<mask>.** A special token used by ESM-1b.

3. **No linker.** s_{binder} and s_{target} are concatenated without a separating linker.

ESM-1b imposes the constraint that any input protein sequence must be no longer than 1024 residues. In our experiments, the combined lengths of s_{binder}, s_{linker} and s_{target} was always considerably less than 1024. We experimented with linker lengths 10, 50, and 100.

5.2.2 Formulating a *Promise Score*

In order to assess the promise of a given sequence s_{binder} as a focus for protein engineering, we use the attention map for s to calculate a "PROMISE SCORE". Treating the attention map as a predicted protein-protein interaction map, we use it to calculate a score that accounts for *intermolecular* interactions between s_{binder} and s_{target}, and, when appropriate, *intramolecular* interactions within s_{binder}. Let n, ℓ, and τ be the lengths of s_{binder}, s_{linker}, and s_{target}, respectfully, and let $N = n + \ell + \tau$. We accomplish this by first isolating sections of $\mathcal{A}$ that correspond to either intermolecular or intramolecular interactions:

$$\mathcal{A}_{inter} = \mathcal{A}_{1:n,n+\ell:N} \in \mathbb{R}^{n \times \tau} \qquad \mathcal{A}_{intra} = \mathcal{A}_{1:n,1:n} \in \mathbb{R}^{n \times n} \tag{5.3}$$

$\mathcal{A}_{inter}$ and $\mathcal{A}_{intra}$ represent intermolecular and intramolecular interaction profiles, which we use to calculate separate intermolecular and intramolecular scores. We obtain these profiles and scores for all sequences being considered for protein engineering. To standardize the numerical scaling of these profiles in order to ensure fair comparisons between different sequences, we apply min-max scaling to both as follows:

$$\mathcal{A}' = \frac{\mathcal{A} - \texttt{min}(\mathcal{A})}{\texttt{max}(\mathcal{A}) - \texttt{min}(\mathcal{A})} \tag{5.4}$$

The intermolecular interaction score S_{inter} is given as a sum over $\mathcal{A}_{inter}$ normalized by the length of s_{binder}[1]. For all $a_{ij} \in \mathcal{A}_{inter}$:

$$\mathcal{S}_{inter} = \frac{1}{n} \sum_{i,j} a_{ij} \tag{5.5}$$

In our protein discovery experiments, we simply use Promise Score $\mathcal{S} = \mathcal{S}_{inter}$. In many real-life protein engineering pipelines, a subsequent protein optimization stage applies site-specific mutagenesis experiments to a functional but suboptimal baseline protein sequence $s_{baseline}$. This could correspond to a lead sequence selected from a previous protein discovery campaign, or a wildtype sequence that has been identified previously. We obtain intramolecular interaction score $\mathcal{S}_{intra}$ by calculating a difference between the intramolecular interaction profiles of $s_{baseline}$ and s_{binder}. Where $\mathcal{B}_{intra}$ is the intramolecular interaction profile for $s_{baseline}$, we obtain such a score as:

$$\mathcal{S}_{intra} = \|\mathcal{B}_{intra} - \mathcal{A}_{intra}\|_2 \tag{5.6}$$

The larger $\mathcal{S}_{intra}$ is, the greater the difference between intramolecular interaction profiles. We assume that the functional property of $s_{baseline}$ is driven in part by the the intramolecular interactions reflected by $\mathcal{B}_{intra}$. This leads us to expect that a functional s_{binder} should have an intramolecular interaction profile that is more similar than dissimilar to $\mathcal{B}_{intra}$. This reasoning motivates our formulation of the Promise Score $\mathcal{S}$ during protein optimization— a linear combination of $\mathcal{S}_{inter}$ and $\mathcal{S}_{intra}$:

$$\mathcal{S} = \mathcal{S}_{inter} - \lambda \mathcal{S}_{intra} \tag{5.7}$$

A promising design will have many intermolecular interactions and an intramolecular interaction profile that is similar to a known functional baseline.

Hyperparameter $\lambda \geq 0$ governs the degree to which the intramolecular term impacts the overall score. In principle, any standard hyperparameter fitting method could be

[1]In our experiments, the length of s_{target} does not change. To generalize to settings where this length is variable, Eq. 5.5 should be further scaled by $1/m_t$, where m_t refers to the length of target sequence t.

applied to set λ. We have devised a means to automatically select λ that takes advantage of the fact that S_{inter} and S_{intra} can be pre-computed for any sequence without the need for an external label. As $\mathcal{S}_{inter}$ and $\mathcal{S}_{intra}$ are typically considerably different in magnitude, our approach is to treat λ as a scaling factor between the two terms so that one value does not dominate the other. Let $\boldsymbol{\mathcal{S}}_{inter}$ and $\boldsymbol{\mathcal{S}}_{intra}$ be the set of all $\mathcal{S}_{inter}$ and $\mathcal{S}_{intra}$ computed for each sequence that is being evaluated. Let $\overline{\boldsymbol{\mathcal{S}}_{inter}}$ refer to the mean of $\boldsymbol{\mathcal{S}}_{inter}$, and $\overline{\boldsymbol{\mathcal{S}}_{intra}}$ the mean of $\boldsymbol{\mathcal{S}}_{intra}$. We identify the scaling factor by taking a simple ratio, and use this approach to select λ in all of our experiments:

$$\lambda = \frac{\overline{\boldsymbol{\mathcal{S}}_{inter}}}{\overline{\boldsymbol{\mathcal{S}}_{intra}}} \tag{5.8}$$

5.2.3 Fine-tuning and knowledge transfer with ESM-1b

While we have so far used the pre-trained ESM-1b model as-is in order to calculate Promise Scores, we now describe how it can be *fine-tuned* in order to learn a model that predicts the binding strength between s_{binder} and s_{target}. In order to imbue ESM-1b with the ability to provide such a prediction, we attach a three-layered neural network to the head of the model. In other words, ESM-1b's learned representation serves as input for the neural network head. Fine-tuning occurs during model training. During backpropogation, we fine-tune by updating the parameters of ESM-1b's final Transformer block in conjuction with the parameters of the neural network head. All other ESM-1b parameters are left fixed at their pre-trained state. In our experiments, the neural network head consists of three linear layers connected by rectified linear unit (ReLU) activation functions. The full model is trained using cross-entropy loss for 30 epochs using PyTorch's AdamW optimizer [125, 126] with default parameter settings (learning rate $\gamma = 1e{-}6$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay coefficient $\lambda = 0.01$).

We expect that it would be beneficial to be able to take knowledge gained in one engineering campaign and apply it to that of another (related) campaign. To this end, we show the ability to transfer knowledge learned from one protein engineering objective to another within the discovery and optimization stages. Within protein discovery, we use models for protein binding learned from Nano-HSA data and show their ability to predict binding on Nano-GST data, and vice-versa. Similarly within protein optimization, we take models learned from BRCA1-BARD1 binding data and show their ability to predict binding between Spike and ACE2, and vice-versa.

5.2.4 Data

We use the PROMISE SCORE at two phases of the protein engineering pipeline— protein discovery and protein optimization. Table 4.2 provides an overview of the data we use in our experiments.

At the protein discovery phase, we show how to use the PROMISE SCORE to select lead sequences from two Nb repertoire data sets [127]. The repertoires were previously collected from an immunized llama using protein targets human serum albumin (Nano-HSA) and glutathione S-transferase (Nano-GST). The relative binding strengths of each Nb sequence to its target sequence was originally provided as a categorical label, where one of the three possible labels corresponds to strong binding sequences. We re-labeled each sequence as "strong binding" or "not strong binding" to obtain a binary descriptor for relative binding strength, which we use in our experiments. In total, the Nano-HSA data set contains 20,670 Nb sequences and the Nano-GST data set 51,330. The percentage of strong binders in Nano-HSA and Nano-GST was 47.5% and 31.1%, respectively.

When computing PROMISE SCORE $\mathcal{S}$ with these data, we let s_{binder} be the full Nb sequence. However, when obtaining $\mathcal{A}_{inter}$, we focused our attention on the CDR3 region of each Nb by only considering the region of attention map $\mathcal{A}$ that corresponds to intermolecular interactions between the CDR3 and target sequence. Figure B.1 shows the distributions of CDR3 sequence lengths for both repertoires.

At the protein optimization stage, we show how to use the PROMISE SCORE to select mutagenesis experiments for two different proteins— BRCA1 and Spike. We obtained

Table 5.1: An overview of the data we used to evaluate the PROMISE SCORE. The sequence IDs are given as the DataBase:ID indicating where the sequence can be accessed, or are NA (not applicable).

Dataset	Baseline $s_{sequence}$	s_{target}	Total sequences	"Strong" percentage
Nano-HSA	NA	UniProt:P02768	20,670	47.5
Nano-GST	NA	UniProt:P08515	51,330	31.1
BRCA1	PDB:1JM7	PDB:1JM7	1389	10
Spike	PDB:6M0J	PDB:6M0J	3651	10

BRCA1 sequences from single site mutational scans [82] that measured the downstream activity caused by interaction between BRCA1 and its binding target protein BARD1. Spike sequences were similarly obtained from single site mutational scans [84] that measured the binding affinity between Spike and its binding target protein ACE2. These are the same data previously described in Sections 4.2.3 and 4.2.4, respectively. We obtained binary labels for both datasets by binning the raw experimental values, where the top 10% were labeled as "strong" sequences, and the rest "not strong".

5.3 Experiments and results

5.3.1 The *Promise Scores* of strong and weak sequences are different

In order for the PROMISE SCORE to provide an effective means for selecting sequences for protein engineering, the scores of known strong binders should be distinguishable from those of weak binders. According to how the PROMISE SCORE is formulated, we further expect that the scores of strong binders should be generally higher than those of weak binders. For both Nb discovery and protein optimization, we computed PROMISE SCORES for all sequences using different linker types and lengths, and compare how strong binders were scored relative to weak ones. The raw scores are only comparable within a given linker type and length— the relative ability to discern strong from weak binders is what can be compared.

During Nb discovery (Figure 5.2-top), there is a significant difference in the distribu-

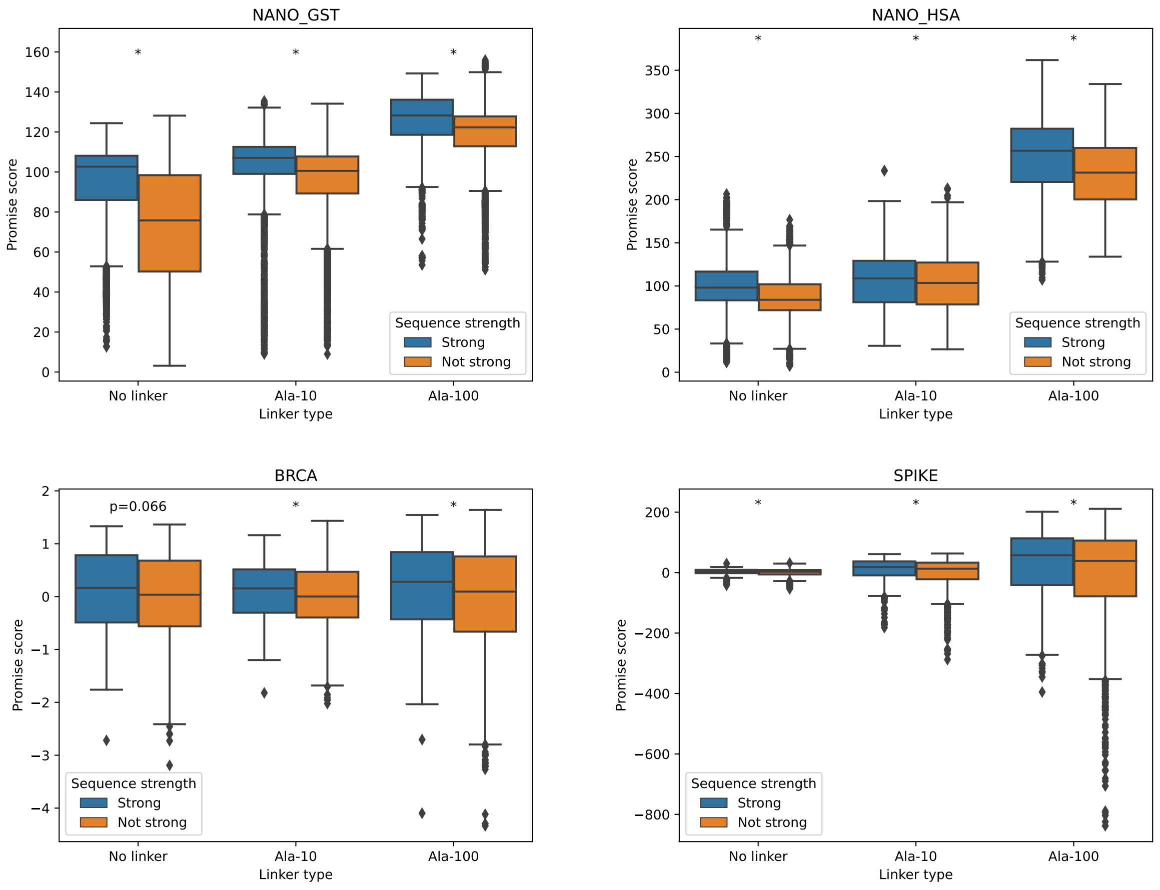


Figure 5.2: We calculated Promise Scores for each sequence identified within two Nb discovery campaigns (Top) and two protein optimization campaigns (Bottom), and compare how "strong" sequences scored compared to those that were "not strong". In each case, strong binders had an average Promise Score greater than sequences that were not strong. These differences were either statistically significant ($p < 0.05$ denoted with *, Mann-Whitney U-test), or had a p-value just above the 0.05 threshold (BRCA with no linker— $p = 0.066$).

tion of Promise Scores of strong and weak binders for each linker type and length, and the average score of strong scoring sequences is higher than that of weak binders. For Nano-GST, using no linker obtains Promise Scores that have greatest ability to discern strong from weak binding Nb sequences on average. For Nano-HSA, a length 100 alanine linker provides the greatest ability to discern. For both Nano-HSA and Nano-GST, calculating a Promise Score using a mask linker of any length or a length 50 alanine also confers the ability to discern strong from weak binders (Figure B.2-top).

During protein optimization (Figure 5.2-bottom), in all but two cases, there is again a

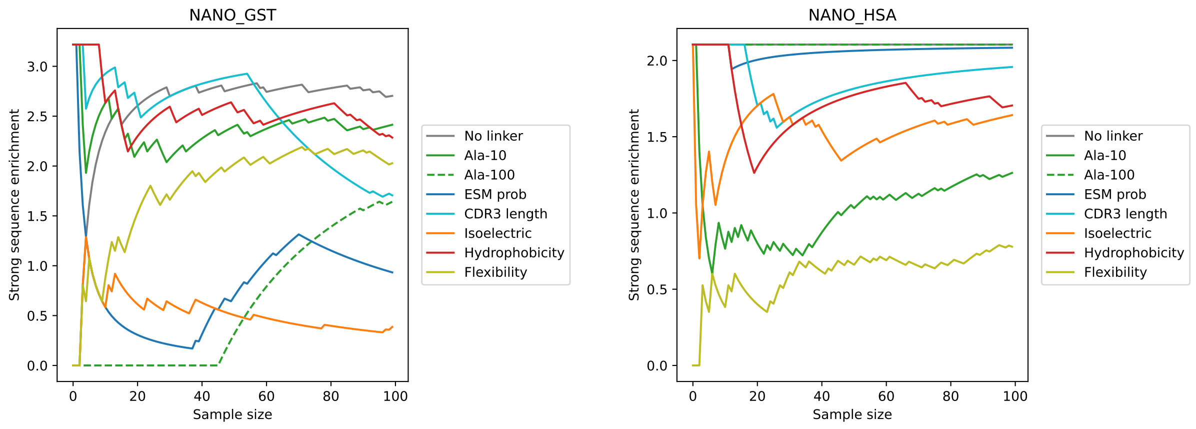


Figure 5.3: We used PROMISE SCORES to help identify potential lead sequences from two Nb discovery campaigns. We found that PROMISE SCORE derived methods identified sequences that were enriched with strong binders, and often identified a higher frequency of strong binders compared to other sequence-based calculation methods. In both Nb discovery campaigns, the best overall strategy used the PROMISE SCORE.

significant difference in the distribution of PROMISE SCORES of strong and weak binders for each linker type and length, where the stronger sequences have a higher PROMISE SCORE on average than their weak counterparts. The exceptions are with BRCA1 using no linker and a length 10 mask linker, though they just barely misses the threshold for significance at $p = 0.066$ and $p = 0.054$, respectively. In general, the PROMISE SCORES for each linker type and length in protein discovery and protein optimization provides a narrow (but still real) ability to discern strong from weak binders. This also includes using a mask linker or length 50 alanine (Figure B.2-bottom). We found that including the S_{intra} term in Equation 5.2.2 ensured this ability to discern strong from weak sequences (Figure B.3).

5.3.2 The *Promise Score* identifies beneficial experiments

Since strong binders tend to have higher PROMISE SCORES than weak binders, we can use the score to rank the sequences in a repertoire and expect that a set of top-ranked sequences will be enriched with strong binders. Figure 5.3 demonstrates this for different sized subsets obtained by taking the top $1, 2, \ldots, 100$ sequences ranked by the PROMISE

Score. Thus, the Promise Score may be useful as a means to identify binders.

During protein discovery, we compare the cumulative strong binder enrichment for the top 100 sequences obtained using the Promise Score calculated with no linker, 10 length alanine, and 100 length alanine linkers. The enrichment is given as the ratio of the observed strong binder frequency at a given sample size to the expected strong binder frequency (i.e. the underlying true strong binder frequency). We compare to sequence-derived calculations commonly used to assess Nb sequences when identifying potential lead sequences. Though none of these metrics directly measure a protein's ability to bind a target, they measure qualities that are reasonably expected to correlate with this property. These metrics include:

1. **CDR3 Length.** The number of residues within the CDR3. While an optimal CDR3 length depends on the particular target sequence, previous analysis [127] of the Nb sequences used in our experiments determined that sequences with shorter CDR3 tended to be stronger binders.

2. **Nb Isoelectric point (pI).** [128, 129] The pH where the Nb carries no net electrical charge. Low pI Nbs are more likely to aggregate, which can lead to inactivation and induces immunogenicity [130].

3. **CDR3 Hydrophobicity.** The frequency of hydrophobic residues within the CDR3. The prevalence of hydrophobic residues can increase the propensity for aggregation [131].

4. **Nb Flexibility.** [132] The relative mobility of Nb residues. An Nb with greater flexibility may have greater ability to bind its target.

We ranked all sequences according to these metrics and report the cumulative strong binder enrichment of the top 100 sequences. CDR3 length and hydrophobicity are ranked in ascending order, whereas Nb isoelectric point and flexibility are ranked in descending order. We also used the logits produced by ESM-1b to calculate a probability under the ESM-1b model for the CDR3 of each sequence. We do this by computing the softmax over the logits associated with each position of the CDR3. The probability is then given as the product over the probabilities assigned to each residue present in the sequence. We only used the Nb sequences as input for ESM-1b when calculating a probability, rather than the sequence concatenated to the target sequence, and report the cumulative strong binder enrichment of the top 100 most probable sequences.

With Nano-GST (Figure 5.3-left), the Promise Score computed with no linker and a length 10 alanine linker have a cumulative strong binder enrichment greater than two at almost every sample size up to 100. At sample size 100, they produce two of the highest cumulative strong binder enrichment values, along with a length 50 alanine linker (Figure B.4-left). Using the Promise Score computed with a length 100 alanine linker struggles to identify strong binders for the top 50 scoring sequences, but by sample size 100 produces a cumulative enrichment greater than 1.5. Calculating a Promise Score using a mask linker tends to have a strong binder enrichment less than 2 (Figure B.4-left). Among the sequence-based calculations, hydrophobicity tends to produce the highest cumulative enrichment, and typically ranks between no linker and alanine 10. The shortest CDR3 are enriched with strong binders, but the enrichment fades by sample size 100. The isoelectric point of Nb sequences is generally not enriched for strong binders at these sample sizes, and neither are the probable sequences under the ESM-1b model.

With Nano-HSA (Figure 5.3-right), the Promise Score computed with no linker and a length 100 alanine linker are consistently the two strategies that yield the greatest strong binder enrichment. Both have cumulative enrichment greater than two across all sample sizes up to 100. Using any length mask linker or 50 length alanine yields an enrichment greater than 1.5 for each sample size up to 100 (Figure B.4-right). The most probable sequences under the ESM-1b model also fare well, with a strong binder enrichment consistently right around 2. The Promise Score calculated with a length

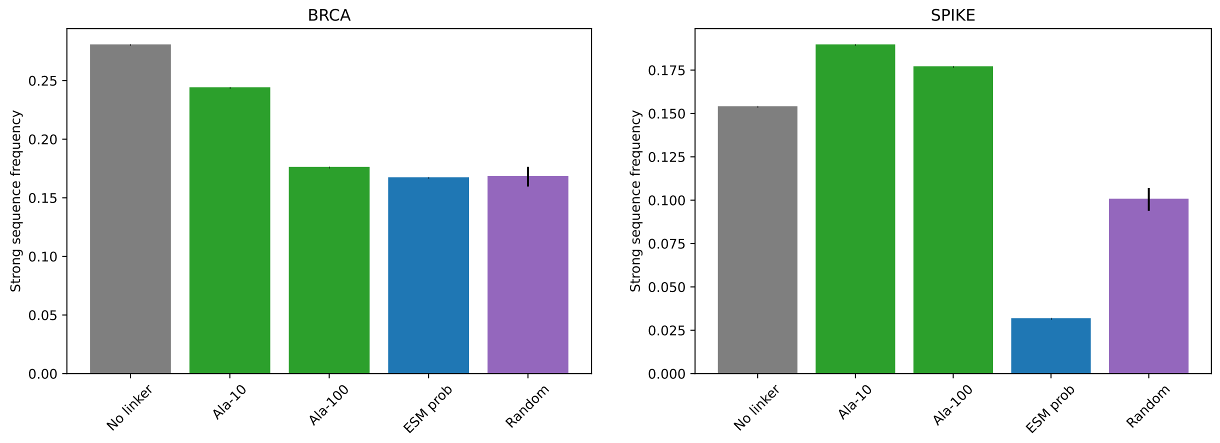


Figure 5.4: We used Promise Scores to select 10 site-specific mutagenesis experiments on each of BRCA1 and Spike. For most linker types and lengths, the Promise Scores identified sites that had a higher strong sequence frequency than selecting randomly or using evolutionary sequence probabilities.

10 alanine linker has the lowest strong binder enrichment among the ESM-1b derived metrics. Each of CDR3 length, CDR3 hydrophobicity, and Nb isoelectric points tend to have cumulative strong binder enrichment that is lower than using no linker or a length 100 linker, but higher than the length 10 linker. Sequences predicted to have high flexibility are not enriched for strong binders.

To summarize across both Nb discovery campaigns, the no linker Promise Scores most consistently identified sequences enriched with strong binders. Among the comparison metrics, each of the isoelectric point, protein flexibility, and ESM-1b were inconsistent–sequences selected according to each were enriched with strong binders in one campaign and not in the other. While both hydrophobicity and CDR3 length consistently identified sequences enriched with strong binders, at most sample sizes, the enrichment was lower than that obtained by the no linker Promise Score.

We also show how the Promise Score can be used to select site-specific mutagenesis experiments during protein optimization. Rather than selecting from a set of discovered sequences, the task here is to select specific positions from a baseline sequence at which to conduct single-site mutagenesis. The intent is to select the positions that will cumulatively yield the highest percentage of strong binders. To do this, we first obtain the

Promise Score for each individual (linked) variant sequence. We then take the top 500 scoring sequences, and identify at which position each sequence is variant relative to the baseline sequence. We treat this tally as a "vote", and the selected sites correspond to the positions that obtain the most votes. In our experiments, we identified the top 10 sites. Figure 5.4 shows the strong binder frequency obtained by this procedure on the BRCA1 and Spike protein optimization sequences.

With BRCA1 (Figure 5.4-left), the Promise Score calculated with no linker identified sites that yielded the highest percentage of strong binders. Alanine with length 10 linker also performed well and had the second highest strong binder frequency. The length 100 alanine linker had the third highest strong binder frequency. We generally found that short mask linkers were also effective (Figure B.5-left). To better quantify how enriched for strong binders these experiments were, we simulated randomly selecting mutagenesis sites 50 times for comparison. Both no linker and a length 10 alanine linker have greater strong frequency enrichment relative to the random sampling. The length 100 alanine linker performed comparably to random sampling. We also calculated the probability of each variant sequence under the ESM-1b model, and used those probabilities to select single sites for mutagenesis. We used the same voting scheme as with the Promise Score, but instead used the sequence probabilities. The sites with most probable sequences yielded strong binders at frequency similar to random sampling.

With Spike (Figure 5.4-right), the Promise Score calculated with alanine linkers yield the highest strong binder frequency. The length 10 linker was highest and the length 100 linker second highest. Using no linker yielded the third highest frequency. All three of these approaches performed similarly to each other and better than random sampling. We found that using a mask linker was generally less effective than using an alanine linker (Figure B.5-right). The sites selected by the most probable sequences under the ESM-1b model were not enriched for strong binders, and had a lower strong binder frequency than random sampling.

5.3.3 The attention map provides insights into protein-target interactions

In addition to identifying strong binders for the purposes of protein engineering, Promise Scores can also provide biological insights about the modes of interaction between a protein sequence and its target protein. To calculate any Promise Score, we use the attention map to quantify intermolecular interactions between a protein sequence s_{binder} and its target s_{target}, given by $\mathcal{A}_{inter}$. The rows of $\mathcal{A}_{inter}$ correspond to each position of s_{binder}, whereas the columns correspond to each position of s_{target}. While calculating the Promise Score involves summing over all of $\mathcal{A}_{inter}$, we can sum over the columns to instead obtain a per-residue score for the target sequence. We use these per-residue scores to identify specific residues on s_{target} that are expected to interact the most with a given s_{binder}. To identify the target residues predicted to be most involved in intermolecular interactions, we took the top 10% of protein sequences according to Promise Score, and used these sequences to obtain an average per-residue score for the target residue. We then identified target residues that scored in the top 20 percentile as those most strongly believed to be involved in intermolecular interactions. Figures 5.5 and 5.6 show these per-residue scores overlaid on 3D structures of each tested protein (Figures B.6 and B.7 shows each of these structures rotated 180°).

In a Nb discovery campaign, identifying target residues involved in intermolecular interaction is akin to identifying Nb epitopes. With both Nano-GST and Nano-HSA, we compared the strongest scoring residues on target proteins to experimentally validated epitope regions. With Nano-GST (Figure 5.5-top left), we identified 42 GST residues predicted to be involved in intermolecular interactions. Of these identified residues, 12 of them are located within the experimentally validated E3 epitope. E3 was the strongest validated epitope, as 50% of the tested Nbs bound to this region. Seven of the 12 residues are singletons within the E3 region located at residues 158-200, and the other five residues are at the contiguous E3 region at residues 213-217. Of the remaining

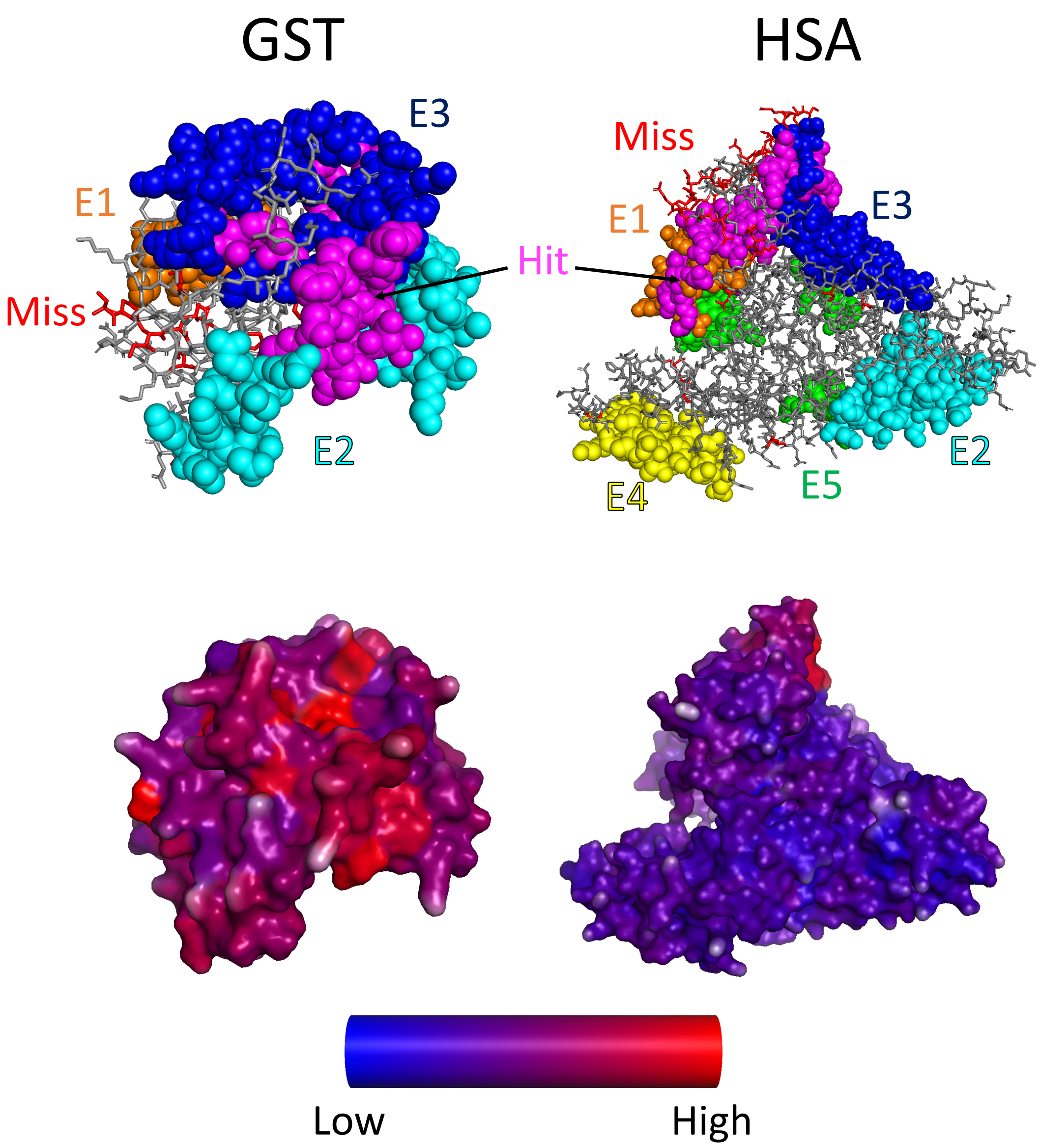


Figure 5.5: We use intermolecular contact map $\mathcal{A}_{inter}$ to calculate residue-specific scores to identify residues that contribute the most to intermolecular interactions. With Nb discovery, we highlight overlap between our top scoring residues and epitopes previously validated through cross-linking mass spectrometry (Top). The magenta spheres indicate this overlap. Each other colored sphere indicates a residue within a validated epitope region. The red sticks correspond to top scoring sites that did not fall in any validated region (denoted “Miss"), and the gray sticks indicate all other residues within the protein sequence. (Bottom) The heatmaps show the relative values of the residue-specific scores across the entire target protein. The coloring scale is normalized for each individual protein to show the relative scores of residues on a given protein (i.e. red shaded residues on GST are the highest scoring residues on that protein, but may have different Promise Scores than the red shaded HSA residues). The structures we used are given by PDB IDs 1DUG and 1AO6.

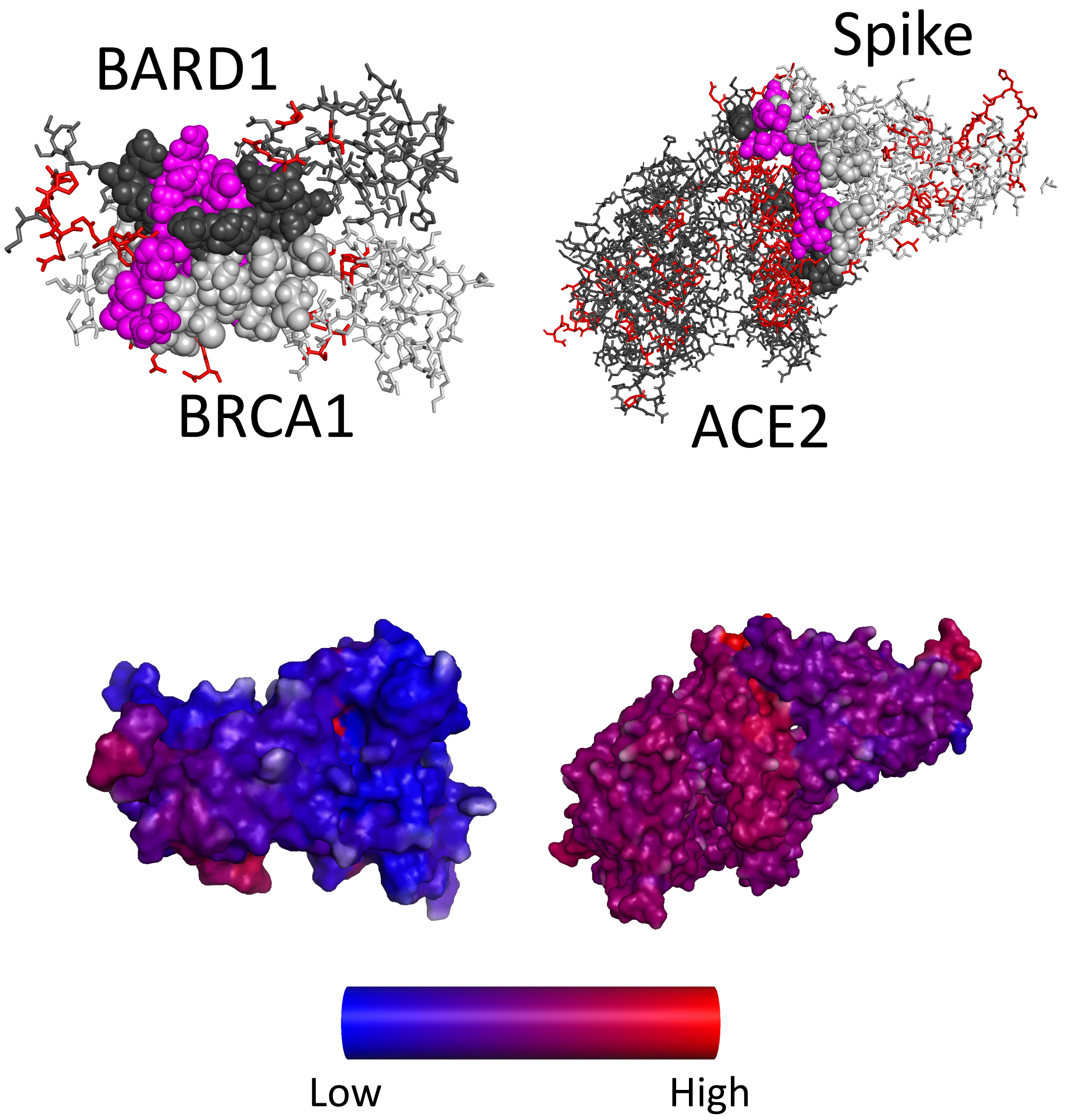


Figure 5.6: We use intermolecular contact map $\mathcal{A}_{inter}$ to calculate residue-specific scores to identify residues that contribute the most to intermolecular interactions. With protein optimization, we show structures of each protein bound to its target. (Top) The spheres indicate residues known to play a role in protein-target binding. Those shaded magenta were identified by the per-residue scoring. Red sticks indicate residues outside this binding region that were also identified. Dark gray sticks indicate s_{target} residues, and light gray s_{binder} residues. (Bottom) The heatmaps show the relative values of the residue-specific scores across the entire s_{binder}-s_{target} complex. The coloring scale is normalized for each individual protein to show the relative scores of residues on a given protein (i.e. red shaded residues on BRCA1 are the highest scoring residues on that protein, but may have different PROMISE SCORES than the red shaded BARD1, Spike, or ACE2 residues). The structures we used are given by PDB IDs 1JM7 and 6M0J.

identified residues, one fell within the E2 epitope region (residue 125), and the rest did not fall within one of the validated epitope regions.

With Nano-HSA (Figure 5.5-top right), we identified 98 HSA residues predicted to be involved in intermolecular interactions. These predicted interacting residues overlap most strongly with validated epitopes E1 and E3. For E1, these include the contiguous region at residues 23-25, 113-114, and 126-127, as well as the singleton residues 28, 111, 116, 118, 123, and 138. For E3, these include the contiguous regions at residues 5-13 and 93-94 as well as singletons at residues 96 and 98. 5% of the validated Nbs bound to epitope E1 and 20% at epitope E3. There were two sporadic singleton residues identified in other validated epitopes, residue 172 in E5, and residue 301 in E2. The remaining selected residues did not fall in one of the validated epitope regions, though blanketed the protein region surrounding epitopes E1, E3, and E5.

We performed similar analyses with both of the protein optimization protein sequences (Figure 5.6). In addition to computing per-residue scores for s_{target}, we also computed per-residue scores for s_{binder} by summing over the rows of $\mathcal{A}_{inter}$. As with the s_{target} per-residue scores, we identified the top 20% of residue scores as those most involved in intermolecular interactions. We found 3D structures for both the BRCA1-BARD1 (Figure 5.6-left) and Spike-ACE2 complexes (Figure 5.6-right), and highlight the residues on s_{binder} and s_{target} that had highest scores indicative of intermolecular interactions. In general, we found that the largest contiguous cluster of identified residues were located at the protein-target binding interface.

Binding between BRCA1 and BARD1 is known to be driven by interactions between BRCA1 residues 8-22 and 81-96 and BARD1 residues 36-48 and 101-116 [133]. Of the 23 high scoring BRCA1 residues we identified, 10 of them fell within these regions (residues 14-15, 18-19, 21-22, 86, 93, and 95-96), and the remaining high scoring residues flanked each of these regions (residues 24, 26-27, 41-42, 79, and 97-103). Similarly, 10 out of the 24 BARD1 residues we identified fell within these ranges (residues 36-38, 42, 45, 102, 105, 108-109, and 112), and the rest also flanked these regions (residues 27-35, 49, 51, 54, 99, and 134).

Spike-ACE2 binding is facilitated through specific interactions between 17 different Spike residues and 20 different ACE2 residues [134]. We identified 39 Spike residues expected to be most involved in intermolecular interaction, two of which overlap with the known interacting residues. While some of the non-overlapping residues are in the vicinity of these binding regions, we found that identified residues were widely spread across the entire Spike protein. With ACE2, we identified 121 residues, including 13 of the known interacting residues. While many of the non-overlapping residues were again in the vicinity of the known binding region, many were also spread throughout the ACE2 protein.

In order to demonstrate that the per-residue patterns we have described above did not arise by chance, we simulated *randomly* selecting residues for each protein system. In each case, we randomly selected a number of residues equal to the amount identified by the per-residue PROMISE SCORE. Over 50 replicates, we identified the frequency with which these randomly selected residues fell within a known binding area, and compared these frequencies to those obtained with the PROMISE SCORE selected residues. Indeed, for each of the regions that we identified substantial overlap between the residues selected by the per-residue PROMISE SCORES and a known binding region, we found that the Promise score selected residues within that region at a rate higher than if selected at random (Figure B.8).

We investigated PROMISE SCORES' ability to predict whether particular residues fall within the previously determined epitopes or binding regions. With respect to the two Nb repertoires (Figure B.9), we found that PROMISE SCORES were able to identify residues within select epitopes better than a random model (i.e. those where we previously identified overlap), but were ineffective at predicting, in general, whether a given s_{target} residue fell within *any* epitope. With protein optimization (Figure B.10), PROMISE SCORES are able to identify residues that fall within the BRCA1, BARD1, and ACE2 binding regions better than a random model, but are ineffective at identifying residues within the Spike binding region.

Finally, to better understand how ESM-1b interprets the alanine linkers that we used to separate s_{binder} and s_{target}, we computed per-residue Promise Scores using the rows of attention map $\mathcal{A}$ that correspond to s_{linker} (Figures B.11 and B.12). This shows how ESM-1b believes the linkers interact with components of the actual protein sequences s_{binder} or s_{target}, as well as with s_{linker} itself. In all the protein systems that we tested, we found that the Promise Scores tend to be highest in the regions of the sequence that correspond to the linker interacting with itself. While there is always a degree of predicted interaction between the linker and s_{binder} or s_{target}, these interactions tend to be fairly uniform, with the exception that sometimes the residues that are sequentially adjacent to the linker region will have increased interaction.

5.3.4 Using ESM-1b to learn models for protein binding

Having used ESM-1b to prioritize sequences expected to strongly bind their target, we now show to what extent we can use ESM-1b to learn predictive classification models for this targeted property. With Nb discovery, we randomly selected 5000 sequences from Nano-GST and Nano-HSA to serve as two separate training sets, and another 15,000 sequences from each as testing sets. Within a protein engineering context, the training sequences are analogous to a preliminary round of experimentation carried out for the purpose of collecting initial data. We then used fine-tuning to train two classifiers, one that was trained with Nano-GST, and one that was trained with Nano-HSA. We then used these models in a traditional (i.e. used to predict binding strength to the same target sequence used during training) and transfer learning manner. As a point for comparison, we also trained classifiers without fine-tuning. That is, during training, we only updated the parameters of the neural network head, and left all ESM-1b parameters fixed. This demonstrates the ability to simply use the pre-trained representations when learning a model for protein binding. We also used these models in both the transfer

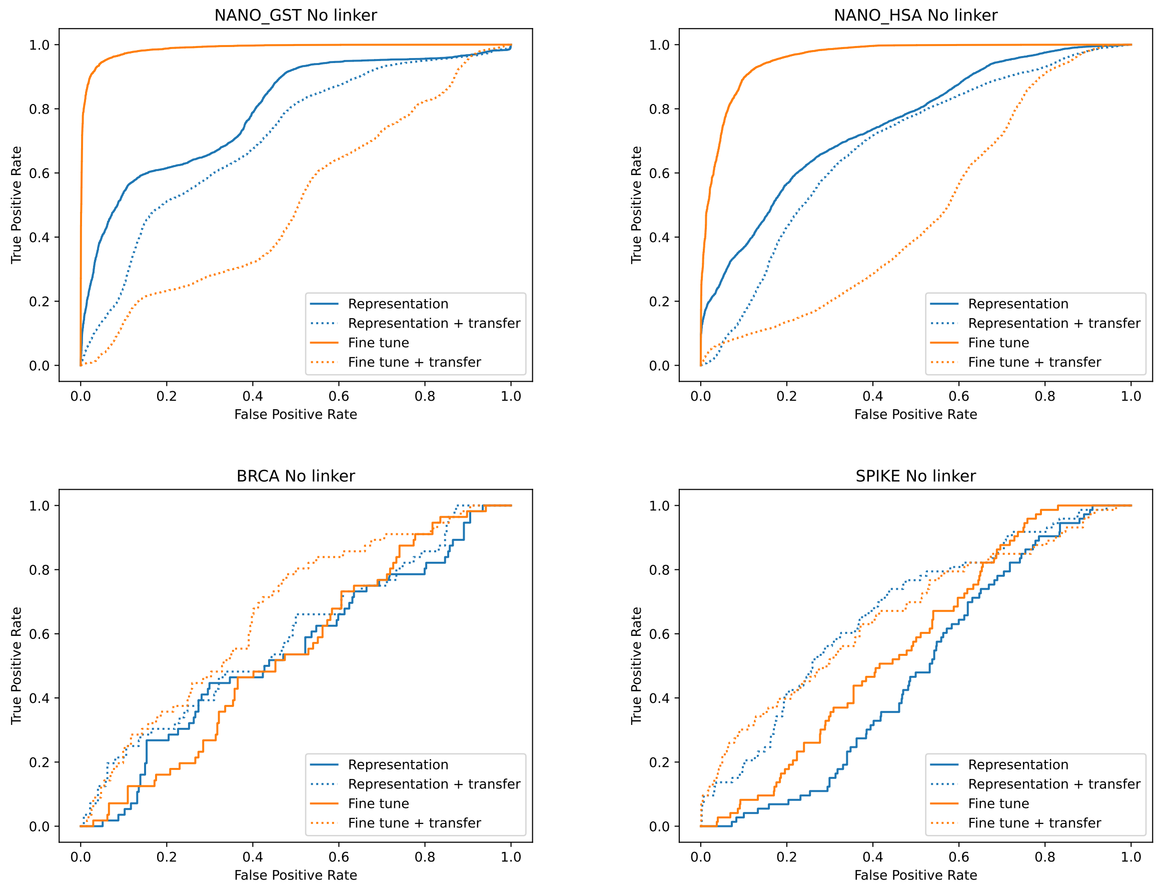


Figure 5.7: ROC curves for models trained to classify strong and weak sequences for both Nb discovery (top) and protein optimization (bottom). Orange curves correspond to models that used fine-tuning, and blue those that did not. Solid curves correspond to models applied in a traditional machine learning paradigm, whereas dashed lines used transfer learning. With Nb discovery, fine-tuned models used in the traditional setting were very strong. While all models learned in protein optimization were of lower quality, we found that fine-tuning with and without transfer performed better than a random classifier with Spike (Bootstrapping with $n = 1000$, $p < 0.05$).

and traditional learning settings.

We obtained comparable results with both Nano-GST and Nano-HSA (Figure 5.7-top). Fine-tuned models used traditionally were far and away the best at predicting binding strength. The model used to predict Nb binding strength with GST had an AUC of 0.988, and the model used to predict Nb binding strength to HSA an AUC of 0.958. The non-fine-tuned models that used pre-trained representations were second best, though clearly less effective than their fine-tuned counterparts. The AUC obtained were 0.795 with GST and 0.753 with HSA. We found that knowledge transfer hurt the predictive capability of both the fine-tuned and non-fine-tuned models. The non-fine-tuned models were hurt less, and still achieved an AUC of 0.712 with GST and 0.683 with HSA. Fine tuning combined with transfer learning essentially yielded random classifiers, as the models has AUC of 0.514 and 0.474 with GST and HSA, respectively. We found similar results for each linker type used in the input sequence encoding (Figure B.13).

We next used ESM-1b to train classifiers using the two sets of protein optimization sequences. For both BRCA1 and Spike, we randomly selected 75% of the single residue sites in each protein. We used the sequences previously obtained via single-site mutagenesis at these selected positions as training data. The mutatations at all other sites were then used as test data. We again trained models with and without fine-tuning, and used both in a traditional and transfer setting.

While we were able to obtain at least one reliable classification model for protein binding using fine-tuning with the Nb discovery data, we found that it was much more difficult to do so within the context of protein optimization (Figure 5.7-bottom). To better quantify the performance of the models trained using BRCA1 and Spike protein sequences, we used bootstrapping ($n = 1000$) to collect a sample of AUC values, and used this sample to calculate p-values from z-scores to ascertain whether each model's AUC was different than a random model (i.e. an AUC of 0.5). With BRCA1, we failed to reject the null hypothesis in each case ($p > 0.05$). With Spike, we were able to reject the null hypothesis with both fine-tuned models ($p < 0.05$). The model using fine-tuning with transfer had an AUC of 0.671, while the model with fine-tuning and no transfer had an AUC of 0.567. The traditional learning models performed no better than random classifiers ($p > 0.05$). As with Nb discovery, we found that using different linkers led to similar trends (Figure B.14).

5.4 Discussion

We have shown how to use the deep Transformer Protein Language Model ESM-1b to calculate a Promise Score that can prioritize specific sequences and be used to guide protein engineers towards designs that are more likely to bind to a given target. We believe this can lead to more efficient and successful protein engineering campaigns.

5.4.1 The role of linkers when computing *Promise Scores*

In our experiments, we hypothesized that self-self attention maps would reveal, in a coarse-grained fashion, the interactions between a sequence s_{binder}, and a binding partner s_{target}. We tested this hypothesis by first concatenating s_{binder} and s_{target} with a separating linker, and use this as input to the ESM-1b model. Strictly speaking, ESM-1b was not designed with this use case in mind. Rather, the model was trained using millions of *monomer* sequences, and has been used to describe secondary structures or make downstream predictions for *singular* protein sequences. And yet, when we use the attention map generated with our concatenated protein-target sequence, we are able to calculate Promise Scores that tend to favor strong binders. From a biological perspective, this suggests that the types of interactions present in the monomers used to train ESM-1b are informative enough to identify the intermolecular interaction between a protein and its binding partner. From a modeling perspective, this suggests that representation learning approaches trained on monomers are strong enough to identify the ways that multiple sequences interact with each other.

Where we used linkers, we experimented with two types— a polyalanine, and the special <mask> character that is internal to ESM-1b. Alanine's simple methyl side chain makes it a flexible amino acid. The idea was that the model may interpret a chain of alanines as a "tether" connecting the protein sequence and its target protein sequence. The flexibility would allow the two protein sequences to interact as if they were two unconnected proteins in close proximity. The <mask> character is used when training ESM-1b. Random residues are replaced with the <mask> character, and the model learns to predict which residue belongs where the <mask> is placed. In our use, the <mask> character denotes space between sequences without specifically committing to any particular residue type that may bias the interactions between residues.

Generally speaking, we found that using no linker always produced a high degree of strong sequence enrichment. We found that using a linker could occasionally perform better than no linker, but not overwhelmingly so. When comparing the two linker types, we found that the natural amino acid alanine produced more consistent results, whereas there was much greater variance in the performance of PROMISE SCORES calculated with <mask> linkers. For future work, we would recommend using no linker or a short alanine linker.

When we investigated how ESM-1b may interpret the linker, we found that the attention map indicates the model places highest attention on interactions between the linker and itself as opposed to the linker and s_{binder} or s_{target}. Since the linker is ultimately an unnatural sequence, perhaps this suggests that ESM-1b has a general ability to detect unnatural sequences. More pertinent here, it seems that this effect increases as does the linker length. This perhaps explains our observation that longer alanine linkers lead to less reliable PROMISE SCORES— ESM-1b's attention is preoccupied by these unnatural sequences to the detriment of identifying real interactions between s_{binder} and s_{target}.

5.4.2 The generalizability of *Promise Scores*

We first showed how to use Promise Scores to select lead sequences using Nbs obtained from two different Nb discovery campaigns. Notably, we showed that Promise Score selected sequences are often stronger than sequences selected by a number of sequence-based scoring metrics. This again shows the power of representation-learning approaches like ESM-1b. Despite not being trained to know anything about the biophysical properties of a given protein sequence, we show that we can use the model to make inferences about the relative strength of a protein with greater effect than using biophysical calculations.

We also showed how to use Promise Scores to select single site mutagenesis experiments during protein optimization. The primary task associated with protein optimization is considerably different than that of Nb discovery. Rather than identifying a subset of promising sequences from a naturally large and diverse repertoire, the goal is to identify specific sites on a baseline sequence to perform single-site saturation mutagenesis. As a result, the set of sequences under consideration will be smaller and more uniform than those encountered during Nb discovery, both of which add to the challenge of this task. We overcame this challenge by adding an extra term ($\mathcal{S}_{intra}$) to the calculation of Promise Score $\mathcal{S}$ that accounted for intramolecular interactions. The formulation of $\mathcal{S}_{intra}$ is consistent with protein optimization— we want to identify sequences that enhance an already existing feature of a baseline sequence. So while we have shown the effectiveness of computing particular Promise Scores for Nb discovery and protein optimization, the more general point is that we have shown an ability to formulate scoring functions using the attention map of a Transformer Protein Language Model that are tailored to specific protein engineering objectives. We believe that our work can serve as a baseline for different protein engineering objectives that have different constraints and assumptions.

While we have demonstrated how the Promise Score may be used in isolation to prioritize protein sequences during Nb discovery and protein optimization, we know that most protein engineering pipelines are multi-faceted. They often include different types of analyses that have somewhat orthogonal yet complementary focuses. These may include genomic assays that identify DNA-protein interactions [135], molecular modeling that predict molecular dynamics and structure [136], or phylogenetic analyses that uncover genomic relationships between related species [137]. We want to emphasize that using Promise Scores is similarly complementary, as they can be used to prioritize sequences at any and all stages of complex protein engineering pipelines.

Finally, we believe it is important to note that Promise Scores can be computed using any Transformer Protein Language Model, not just ESM-1b. While we found ESM-1b to be an effective and convenient model with which to showcase this ability, any model that uses self-attention could be used in its place. Our intent is not to champion one Transformer model over the other, but to demonstrate that this class of model can be used to prioritize protein sequences within the context of protein engineering. We see this as a strength of the approach, as it can be easy to implement using new and improved models and they become available. Indeed, even the authors of ESM-1b have released other versions of Transformer Protein Language Models with differing designs and specifications since we began working with ESM-1b [138, 139, 140]. We leave formal comparisons between Transformer Protein Language Models applied towards this goal to future work.

5.4.3 ESM-1b as a coarse-grained model for intermolecular interactions

While the Promise Score is a singular numeric value that reflects the promise of a sequence in its entirety, we showed that we can also easily obtain per-residue scores for s_{binder} or s_{target}. Using these per-residue scores, we show that we can identify specific residues that are known to be involved in intermolecular interactions. More generally, we believe these results demonstrate that ESM-1b can be viewed as a coarse-grained model for intermolecular interactions. With each set of sequences that we used, we observed that the “hits” and “misses” tended to congregate in regions of each protein that have highest concentration of known interacting residues.

In the case of Nb discovery, we used these per-residue scores to identify potential epitopes, and compared these findings to experimentally validated epitope sites. It is important to note that the experimental validation consisted of cross-linking models at sites chosen by computational molecular docking. The validation thus does not provide any evidence for or against potential epitopes beyond the sites that were tested. Additionally, the experimental validation only used 24 Nb sequences, meaning the reported binding percentages are noisy estimates of the true underlying values. We believe these contributed to our finding that Promise Scores did not effectively predict whether a given s_{target} residue fell within one of these epitope regions— it is likely that these regions do not actually correspond to ground truth binding regions for many of the epitopes within the repertoire, including those that are known to be strong binders.

Still, across two sets of Nb discovery sequences, the per-residue scores identified sites that overlapped three out of eight total validated epitope regions, as well as identifying many sites outside these regions. We believe that this is an encouraging sign that Protein Language Models like ESM-1b are able to identify specific residues that are expected to contribute to intermolecular interactions. Such an ability could allow protein engineers to better pinpoint the regions in a protein of interest that should be investigated within an protein optimization campaign. Of course, we also made simplifying assumptions in our work that contribute to disagreement with the validation analyses. For one, we only considered interactions involving CDR3, so any effect caused by CDR1 or CDR2 are unaccounted for entirely. As such, we see these analyses as a jumping off point for future work that identify epitope, or even paratope sequences.

Applied to protein optimization, we investigated not only finding per-residue scores for s_{target}, but also doing so for s_{binder}. For the BRCA1-BARD1 complex, nearly half of the residues that we identified fell within the known binding interface. This strongly suggests that high per-residue scores obtained from $\mathcal{A}_{inter}$ do indeed identify residues involved in intermolecular interactions. In the case of Spike-ACE2, we identified more than half of the known interacting ACE2 residues, though only two out of the 17 known interacting Spike residues. This highlights an important distinction behind how we compute scores for each s_{target} (ACE2) and s_{binder} (Spike). For each sequence used, s_{target} is unchanged, whereas each s_{binder} differs from all others at exactly one residue (due to each Spike sequence having been obtained from a single site mutation scan). Thus, the distribution of high scores throughout the Spike sequence indicates that the model believes different mutations affect the residues that interact at the binding interface. This capability could be used to generate hypotheses about how mutations affect binding patterns between a protein and its partner. Still, that the vast majority of known ACE2 interacting residues were identified indicates that the Transformer accurately identifies interacting residues within the Spike-ACE2 complex.

5.4.4 Model learning within the context of protein engineering

Having a predictive model for the targeted property in a protein engineering campaign could serve as an invaluable asset, as it can serve as a tool that helps with sequence design decisions. For this reason, we investigated how well we could learn such models using ESM-1b. First, we compared fine-tuning ESM-1b to using the sequence representations from the pre-trained model, and found that fine-tuning worked very well with the Nb discovery setting. With protein optimization, we found it difficult to learn reliable models with or without fine-tuning. We believe this ability to learn reliable models with Nb discovery but not with protein optimization can be explained by differences between these data. As discussed in Section 5.4.2, the Nb discovery data is both more diverse and contains more sequences than the protein optimization data. Diversity and data quantity are well-known factors that influence the ability to train machine learning models [141]. This highlights an important consideration for the experimental design choices within protein engineering campaigns. Our work suggests that single-site mutagenesis experiments across relatively short spans of a protein sequence may not yield informative enough data for accurate model learning. If obtaining a reliable predictive model is imperative or desirable when using single-site mutagenesis, protein engineers may want to consider ways to inject greater diversity into their experiments.

In addition to fine-tuning, we also investigated the viability of knowledge transfer when training models. The ability to use knowledge transfer would allow protein engineers to take what they learn during one engineering campaign and apply it to another. With the Nb discovery data, we found that knowledge transfer was less harmful compared to not using knowledge transfer if using the pre-trained representations rather than a fine-tuned model. This makes sense— a fine-tuned model is adapted to work with a specific s_{binder}-s_{target} pair, so using it on different data would be counter productive. Interestingly, with the Spike protein optimization data, we found that using knowledge transfer with fine-tuning yielded a model with higher AUC relative to using knowledge transfer with no fine-tuning. One thought as to why we may see an improvement in this case again relates back to the issue of diversity. Perhaps being trained on a set of sequences that were different entirely than those being tested on actually provided a greater degree of diversity that led to slightly better performance. As a final point, we only tried a very basic form of transfer learning in each of these cases, where we use one data set to train a model (e.g. Nano-HSA), and make prediction on a separate data set (e.g. Nano-GST). It is conceivable that having access to many different data sets to use during training could improve the model's performance.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

In this dissertation, we addressed sequential decision making problems in two biological domains— general laboratory experimentation via a Cloud Lab and protein engineering.

We introduced our algorithm PROTOCOL in Chapter 3, which used the unique capabilities of the Cloud Lab to automate the optimization of experimental parameterizations. PROTOCOL is a bound-based Bayesian optimization algorithm that builds off of the theoretically-grounded IMGPO. Our innovation is to make this optimization parallelizable by calculating a Pareto-optimal frontier that selects multiple experimental parameterizations. These experiments can be executed asynchronously in the Cloud Lab, and the chosen parameterizations represent a balance between exploration and exploitation.

We used PROTOCOL in both a simulated and real Cloud Lab environment. In the simulated lab, it outperforms alternative approaches to Bayesian optimization in terms of its ability to find optimal configurations, and the number of experiments required to find the optimum. In the real-world lab, the algorithm makes progress towards the optimal setting. We believe this work marks the first such instance of academic research based on the Cloud Lab environment, and hope that it can act a springboard for future advances using this technology.

In Chapter 4, we introduced a regularized approach to ML-assisted DE via Bayesian optimization. We evaluated two approaches, one based on structure and the other based on evolutionary constraints. Our results suggest that structure-based regularization using an EI or PI acquisition function usually leads to designs with higher fitness compared to unregularized approaches. In the absence of a structure model, it is natural to consider the use of a sequence-based regularization term. However, our results demonstrate that such terms do not lead to reliably better designs, at least for the specific proteins we considered.

Previous research had demonstrated that ML-assisted DE can reduce the experimental burden, relative to traditional DE. Our results demonstrate that ML-assisted DE via Bayesian optimization decreases the experimental burden further, compared to the method in Wu et al. [64]. We also demonstrated that integrating a regularization term into the acquisition function can lead to better designs, and does so by concentrating exploration at select residues.

Chapter 5 described how to use the transformer protein language model ESM-1b to identify promising sequences during protein engineering campaigns. By jointly encoding a protein sequence s_{binder} and its binding target s_{target}, we use ESM-1b's self-attention mechanism to identify intermolecular and intramolecular interactions between these sequences. We use these interactions to formulate what we refer to as the PROMISE SCORE, and show how this score can be tailored to prioritize protein sequences in two distinct protein engineering domains— protein discovery and protein optimization.

With protein discovery, we show how to use PROMISE SCORES to effectively select lead sequences from two separate Nb repertoires. And with protein optimization, we show how to use PROMISE SCORES to identify single-site mutagenesis experiments that successfully identify strong binders. In both cases, we show how to also compute per-residue scores that indicate those expected to undergo intermolecular interactions. We showcase that high scoring residues on Nb target proteins correspond to known epitopes, and those within the BRCA1-BARD1 and Spike-ACE2 complexes correspond to known interacting residues.

Finally, we demonstrate that ESM-1b can be fine-tuned to learn accurate models for Nb binding strength, and discuss the limitations of model learning in single-site mutagenesis protein engineering campaigns. We believe this work is highly adaptable and directly applicable to many protein engineering pipelines, so can help to make protein engineering more efficient and effective.

6.2 Future directions

6.2.1 Experimental science in the Cloud Lab

Given the relative novelty of applying the Cloud Lab towards academic research pursuits, the space of future directions in this area seems nearly limitless. In Section 3.8, we briefly described a potential modification to the PROTOCOL algorithm, which we will now expand upon.

In our experiments, we operated as first time Cloud Lab users with no previous data to help us make decisions on how to parameterize the laboratory equipment we used. In practice, this may be a somewhat unrealistic scenario. For one, Cloud Lab users will quickly accumulate data that they can refer to in subsequent experimental tasks. Additionally, the Cloud Lab is a shared resource, meaning the work of others may also be available for use. Clearly, it would be beneficial to be able to effectively use any and all existing data to help make experimental decisions.

PROTOCOL is a deterministic procedure that sequentially grows a hierarchical partitioning tree. A limitation of this hierarchical tree structure is that it is not influenced by existing data, only by decisions made using an underlying Gaussian process model. While this underlying Gaussian process model may be trained on all available data, the hierarchical tree is still the scaffold on which PROTOCOL's sequential decision making takes place. If this scaffold could also reflect any level of available prior knowledge, then the entire PROTOCOL optimization routine should stand to benefit. For this reason, we believe that identifying different ways to initialize PROTOCOL's hierarchical tree would be a prudent future direction.

There are two primary considerations to account for in proposing new ways to initialize PROTOCOL's hierarchical tree. First is the actual structure of this initialized tree. Perhaps the most straightforward strategy would be to simply pre-compute a tree similar to the one already used by PROTOCOL to a specified depth. Another approach may be to use an entirely different data structure altogether, such as a *k*-d tree [142]. With a *k*-d tree, the volumes of the hyperrectangles in the initialized tree would depend upon the density of the training instances over the feature space. This could potentially help with the exploration-exploitation tradeoff when computing each frontier, as hyperrectangles corresponding to those most likely in greatest need of exploration would have larger volumes.

In either case, the effectiveness of these strategies will depend upon the second consideration— how to associate the prior knowledge to the pre-initialized tree. Since most any lab equipment will be used to address many varied experimental objectives, most available data would likely not directly correspond to the experimental task at hand. Still, finding ways to address these issues could help make PROTOCOL an even stronger Cloud Lab resource.

Of course, there are many directions beyond the scope of PROTOCOL that can be addressed to further facilitate the advancement of the Cloud Lab within academia. We see PROTOCOL as a first step towards this goal that addresses a practical yet important issue that anyone using the Cloud Lab may face, and believe that future applications of machine learning to this setting will only make the Cloud Lab that much more vital to future academic laboratory-based experimentation.

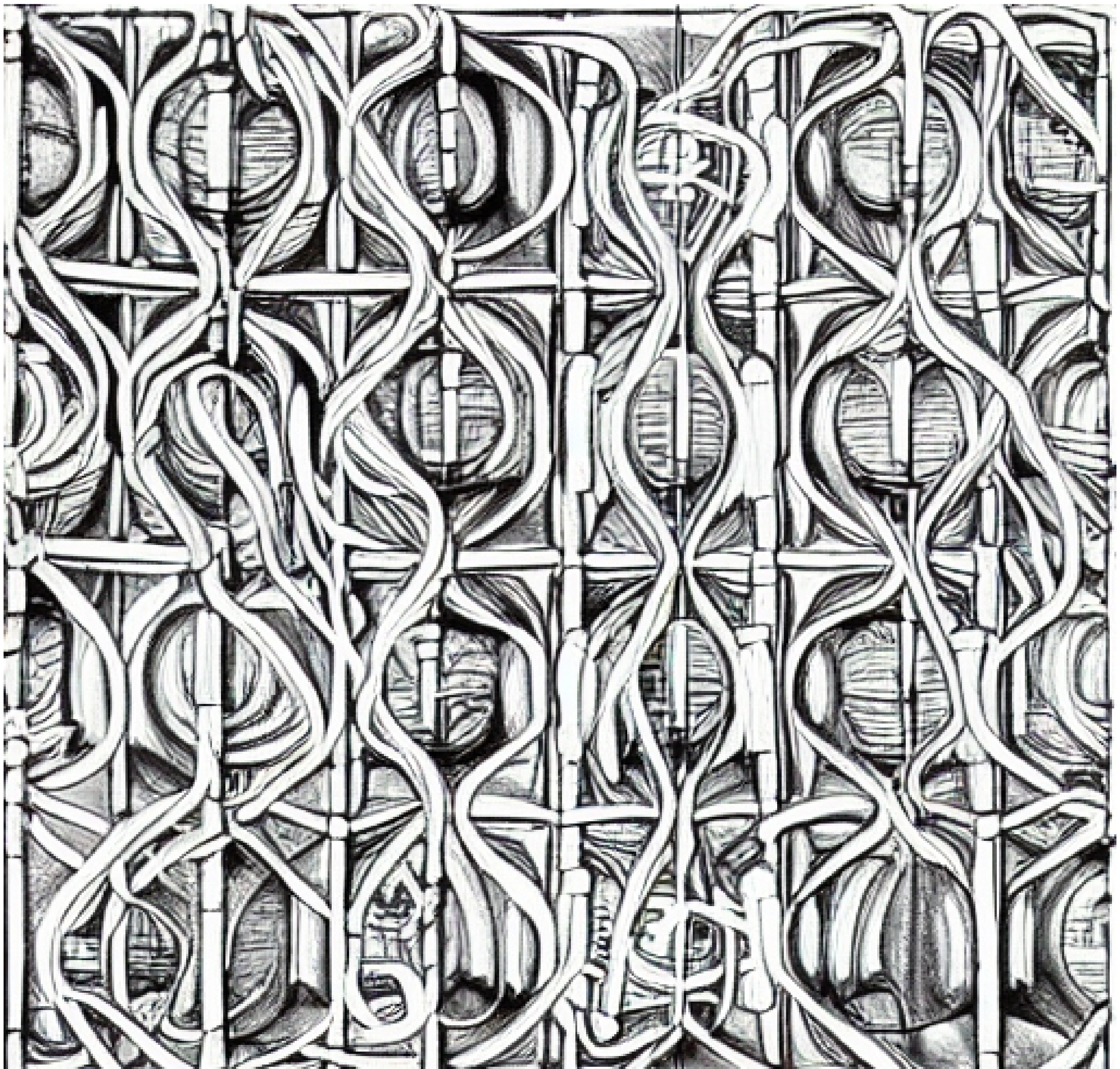
6.2.2 Protein design and engineering

Since we started our own work using transformers for protein engineering, there has been a huge (independent) surge in the popularity of the transformer applied to many different tasks. Towards NLP objectives, transformer-based large language models have shown great promise towards text-based generative capabilities. ChatGPT [121, 143] can generate human-like passages when provided colloquial commands or prompts. Similar models can even generate complex images from text input, like those at the beginning of each chapter of this dissertation [144]. Novel approaches have also been applied to protein sequence design [145]. Many of these applications use the underlying representation learned by a transformer to make predictions about some property of given protein sequences, such as secondary structure [66, 146, 147], homology [148], mutational effects [66, 149, 150], protein-protein interactions [151, 152], and drug-target interactions [153, 154].

Other approaches have more directly used the generative capabilities of the transformer as a means to design new sequences. Madani et al. [89] introduce the conditional transformer ProGen as a means for controllable protein design. Their model "grows" a seed sequence one amino acid at a time by conditioning on a set of keyword tags that describe qualities of the desired protein sequence, such as its host organism, function, and cellular location. Castro et al. [155] introduce Regularized Latent Space Optimization. The method learns a latent space embedding that is constrained to be smooth with respect to protein fitness, continuous with respect to training data points, and pseudo-convex with respect to non-training data. This allows for gradient or non-gradient based optimization within the latent space directly, which will yield novel representations that can be decoded into novel protein sequences. Certainly transformer models will continue to play a large role in the design and engineering of novel protein sequences, and further tapping into the generative capabilities of these models should lie at the heart of most future work.

Before concluding this dissertation, it would be a great oversight to not touch on the potential union between this dissertation's two focal points— Cloud Lab experimentation and protein engineering. In Chapters 4 and 5, we describe automatable approaches to sequential protein design problems, but actual real-world automation is yet to be seen. A Cloud Lab equipped with the right instrumentation to carry out protein engineering experiments could fundamentally change the capabilities of AI-driven protein engineering.

A limitation with many existing deep generative protein design approaches is that they are beholden to existing labeled protein sequence data. The Cloud Lab removes this limitation, as any necessary sequences could be experimentally measured on demand. Perhaps most importantly, this capability could open the possibility for algorithmic paradigms that take advantage of open-ended exploration, such as reinforcement learning [156]. As an example, imagine using a transformer architecture in order to obtain a latent space similar to that described by Castro et al. [155]. Then, use reinforcement learning to train a policy that traverses the latent space to find regions that encode high fitness sequences. At each iteration, the specified sequences are generated in the Cloud Lab, and allow the policy and transformer-based model to be updated in real time as more and more real data becomes available. This is of course a very high-level description of a potential protein engineering algorithm, and there are certainly practical considerations that would need to be hammered out, but these are the types of possibilities that are offered by combining a Cloud Lab with protein engineering. This is exactly why the future of these two domains is so exciting. 🧬



Appendix A

Supplementary Material: Regularized Bayesian Optimization for Directed Protein Evolution

A.1 Evolutionary-based regularization of GB1 with *Gremlin* and profile-HMM log-odds

Our experiments using GREMLIN and HMM regularized approaches follow the same sequential strategy used by experiments outlined in Section 4.5. In Figure A.1, we show how they compare to simulated traditional approaches, as well as those regularized by TPLM and FoldX. When EI or PI is used as the acquisition function, GREMLIN and HMM-based regularization typically identify variants with higher fitness relative to traditional approaches. Compared to the other ML-assisted methods, these approaches tend to identify variants with slightly lower fitness. When UCB is the acquisition function, we find that GREMLIN and HMM-based regularization outperform traditional approaches in roughly half of trials, but is outperformed by other ML-assisted approaches in most trials.

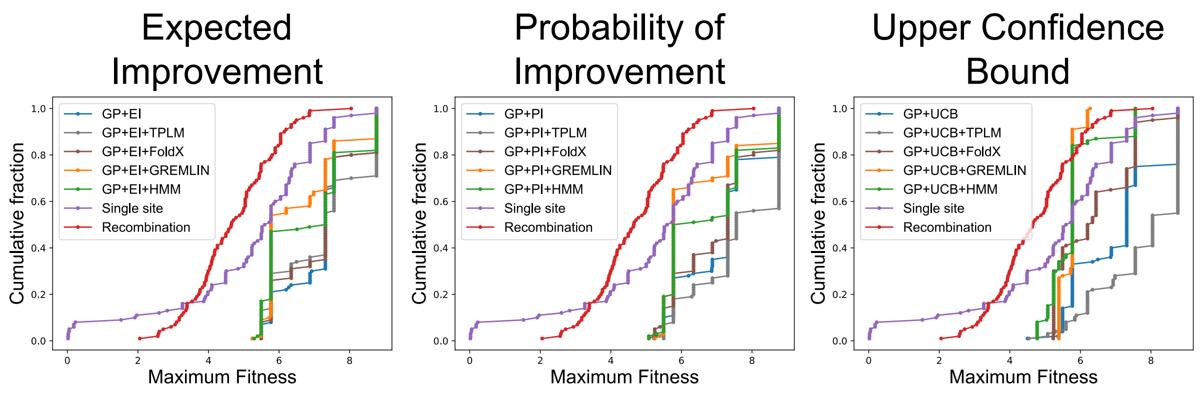


Figure A.1: **ML-assisted Directed Evolution techniques identify high fitness GB1 variants more frequently than simulated traditional DE approaches.** Shown are the fraction of trials (y-axis) that reach less than or equal to a specified fitness (x-axis), where the selection criterion was either a simulated traditional DE approach, or standard or regularized EI **(Left)**, PI **(Middle)**, and UCB **(Right)** was the acquisition function. Experiments with methods regularized by gremlin and profile-HMM log-odds scores are shown alongside results depicted in Figure 4.3.

In Figure A.2-top, we show the same results with GB1 from Figure 4.2 with the addition of GREMLIN and HMM regularized trials. While these additions do greatly improve upon wildtype GB1 fitness, with the exception of using structure-based regularization and UCB acquisition, other ML-assisted DE approaches, regularized or not, identify higher fitness GB1 variants. In Figure A.2-bottom, we show that GREMLIN and HMM-based regularization has the intended effect of biasing variant selections towards those that have high log odds under each model. Putting together these results, when performing evolutionary-based regularization, TPLM is the best option for generative model compared to GREMLIN or profile-HMMs.

A.2 Additional sequence-space exploration experiments

In Section 4.5.5, we describe how regularization induces site-specific exploration of unexplored sequence space. In Section 4.6, we note that this behavior occurs for all regularization types, acquisition functions, and protein types that we investigated. Figures A.3-A.5 show these results for each of GB1, BRCA1, and Spike, respectively. Columns from left

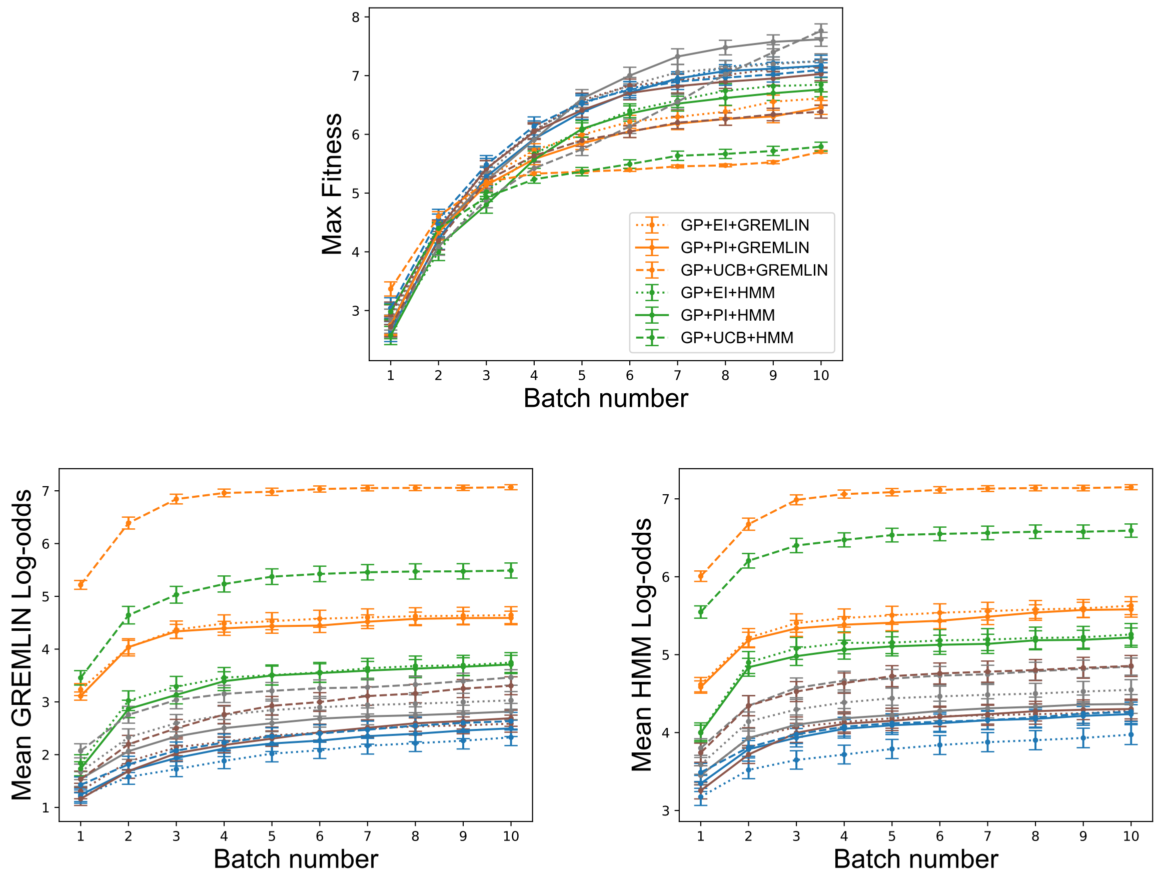


Figure A.2: **(Top) Regularization leads to better designs.** Shown are the cumulative per batch scores for GB1 averaged (± 1 SEM) over 100 trials. GP models were initialized with 20 randomly chosen sequences, and each batch consisted of 19 selected variants. Experiments with methods regularized by gremlin and profile-HMM log-odds scores are shown alongside results depicted in Figure 4.2-left. **(Bottom) Evolution and structure-based regularization biases variant selections towards those that score favorably under the regularization criterion.** Shown are the GREMLIN and HMM log-odds scores for variants selected from the GB1 ML-assisted DE experiments. Variants selected by methods regularized by both of these terms have high log-odds.

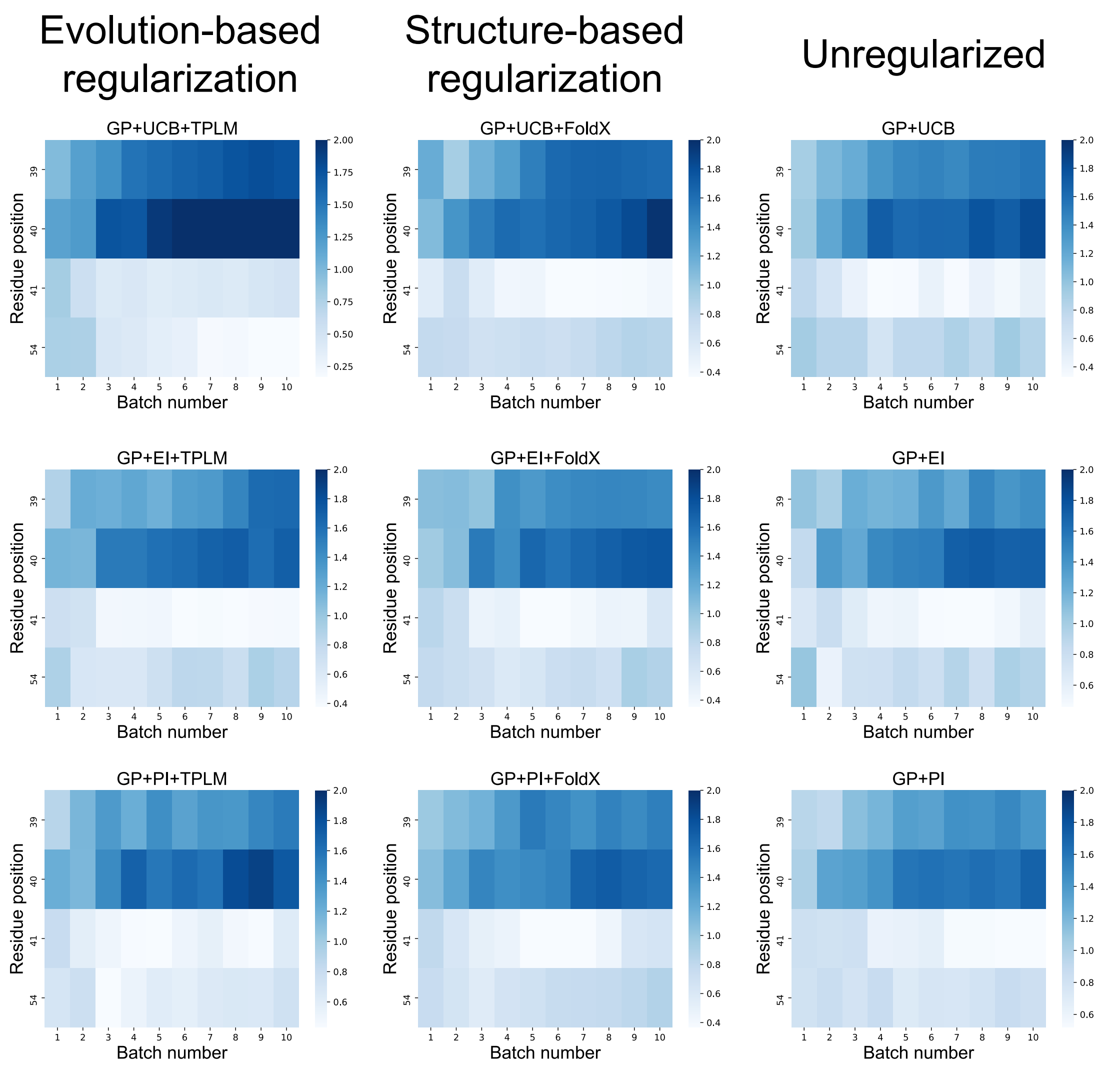


Figure A.3: **Bayesian selection techniques quickly identify informative patterns (GB1).** Shown are the per-batch average position-specific entropy of GB1 variant selections under each (un)regularized method.

to right show TPLM-based regularization, structure-based regularization, and unregularized approaches, and rows from top to bottom show experiments with UCB, EI, then PI acquisition functions. As described previously, we observe localized shading with greater intensity in regularized approaches compared to the unregularized ones. Even with GB1 where there is clearly more exploration at residues 40 and 39 compared to 41 and 54 regardless of regularization, there tends to be darker shading in the regularized approaches.

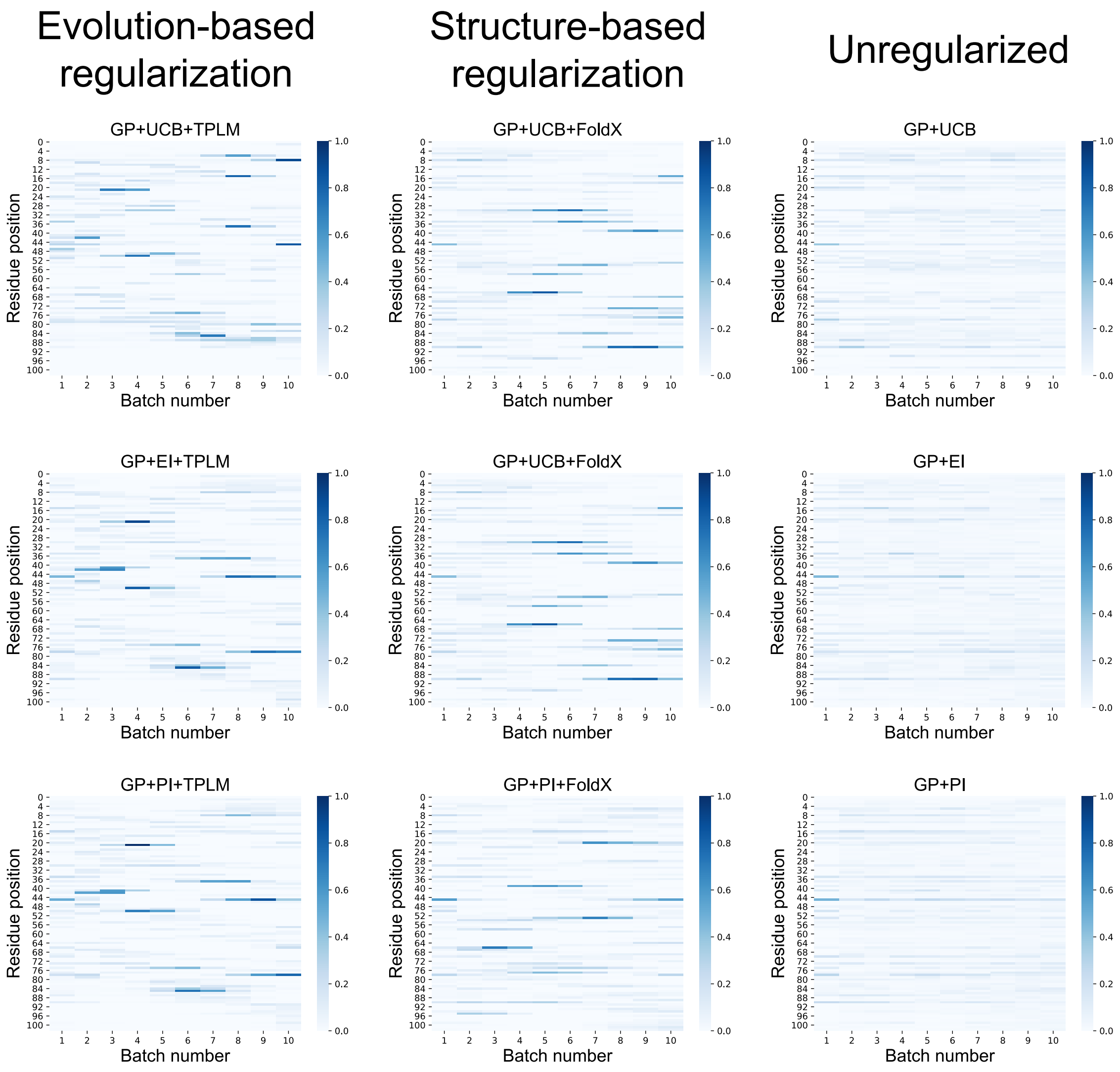


Figure A.4: **Bayesian selection techniques quickly identify informative sequence patterns (BRCA1).** Shown are the per-batch average position-specific entropy of BRCA1 variant selections under each (un)regularized method.

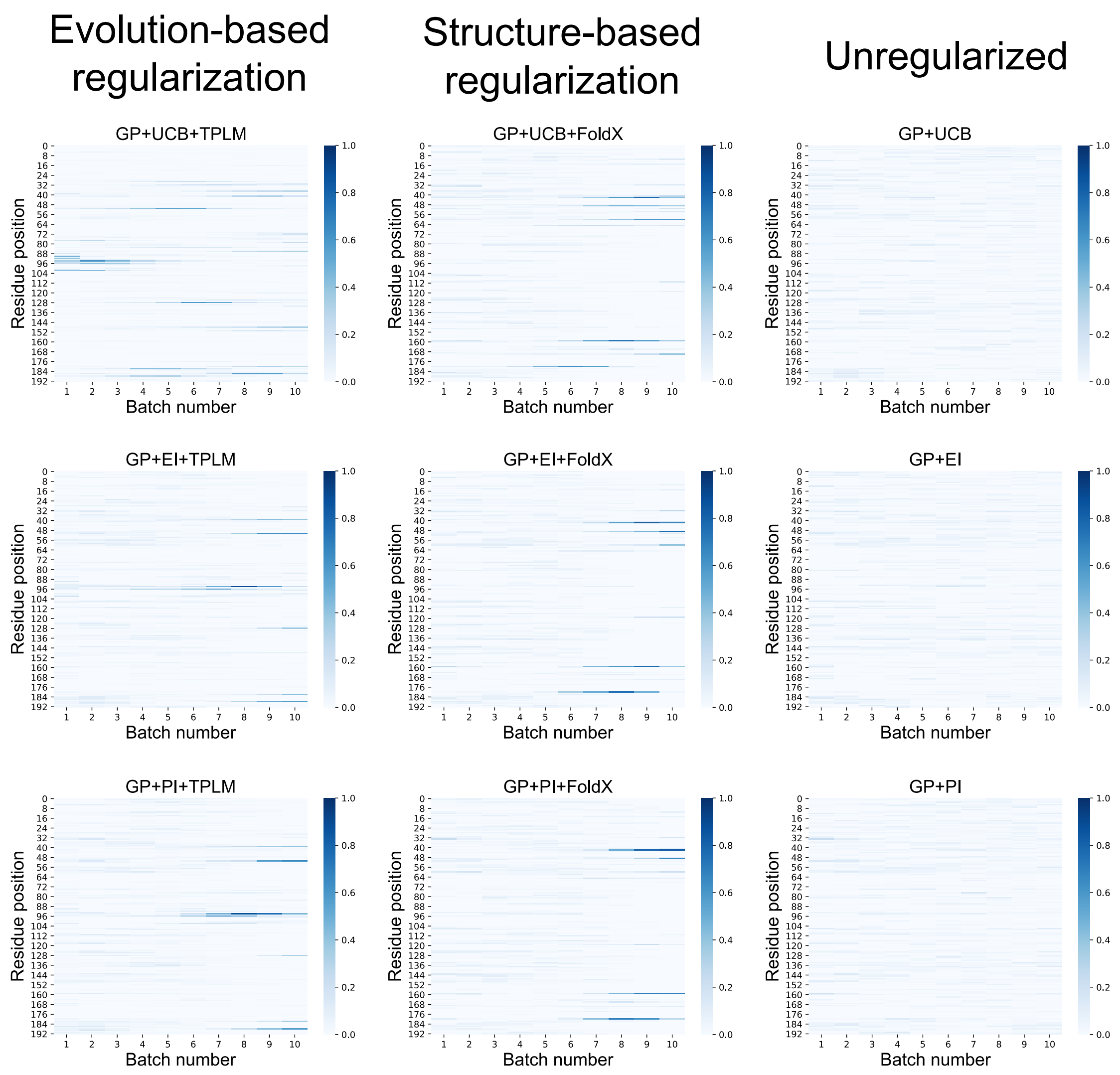


Figure A.5: **Bayesian selection techniques quickly identify informative sequence patterns (Spike).** Shown are the per-batch average position-specific entropy of Spike variant selections under each (un)regularized method.

A.3 Predictions on unseen data

In Section 4.5, we characterize the batched variant sequence selections made by ML-assisted DE techniques with and without evolution or structure-based regularization. These selections allowed us to iteratively update GP models using sequences that each model expected to be informative. To demonstrate the continued predictive capability of these models, we used them to predict the respective objectives of a held out test set for each protein. We emphasize that these sequences were never seen by the models

Table A.1: Predictions on held out data for each protein type, averaged across all regularization types. MSE refers to the mean square error over all predictions. Fitness/Activity/Affinity refers to the average true value for the predicted top sequence obtained for each method.

	GB1		BRCA1		Spike	
Regularization	MSE	Fitness	MSE	Activity	MSE	Affinity
Unregularized	1.96	4.99	0.14	1.47	0.04	0.93
TPLM	2.22	5.05	0.15	1.46	0.02	0.93
FoldX	1.89	5.04	0.14	1.35	0.04	0.92

during the iterative variant selection stage of each trial, and that they constitute a random subsample (20%) of the data for each protein.

For each protein, we used the models from the end of each trial to identify what they believe to be the best variant from the held out testing data. Table A.1 shows the average mean squared error (MSE) averaged across all trials and acquisition types for each form of regularization. Additionally, it shows the average true value of this predicted best sequence. With GB1, we find that all models are high error, but do a good job at identifying a high fitness variant. With BRCA1, the models have better accuracy, and consistently identify a variant that improves upon wildtype E3 ubiquitin ligase activity. With Spike, all model types have good accuracy, and the top predicted sequence is generally comparable to wildtype ACE2 binding affinity. Thus, even when the models have relatively low accuracy, they are able to identify sequences that are comparable to or better than the wildtype sequence, similar to previous results [64].

Appendix B

Supplementary Material: Identifying Promising Sequences for Protein Engineering using a Deep Transformer Protein Language Model

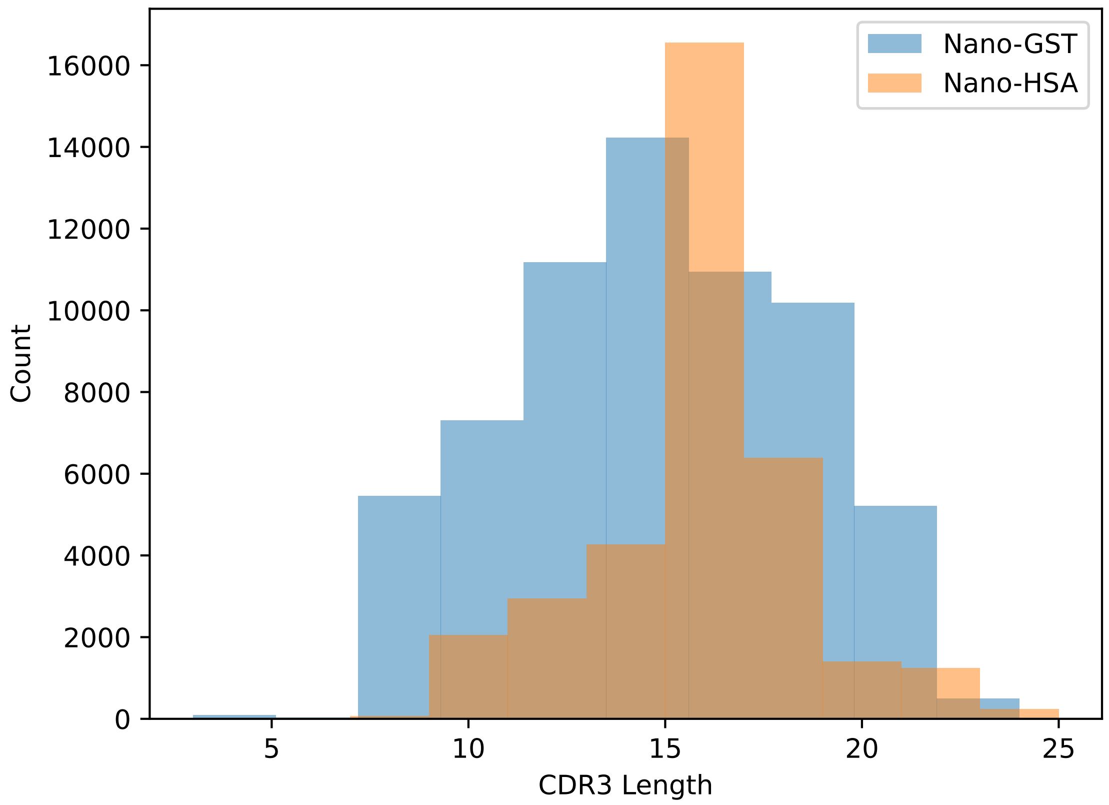


Figure B.1: The distribution of CDR3 lengths from the Nano-GST and Nano-HSA repertoires.

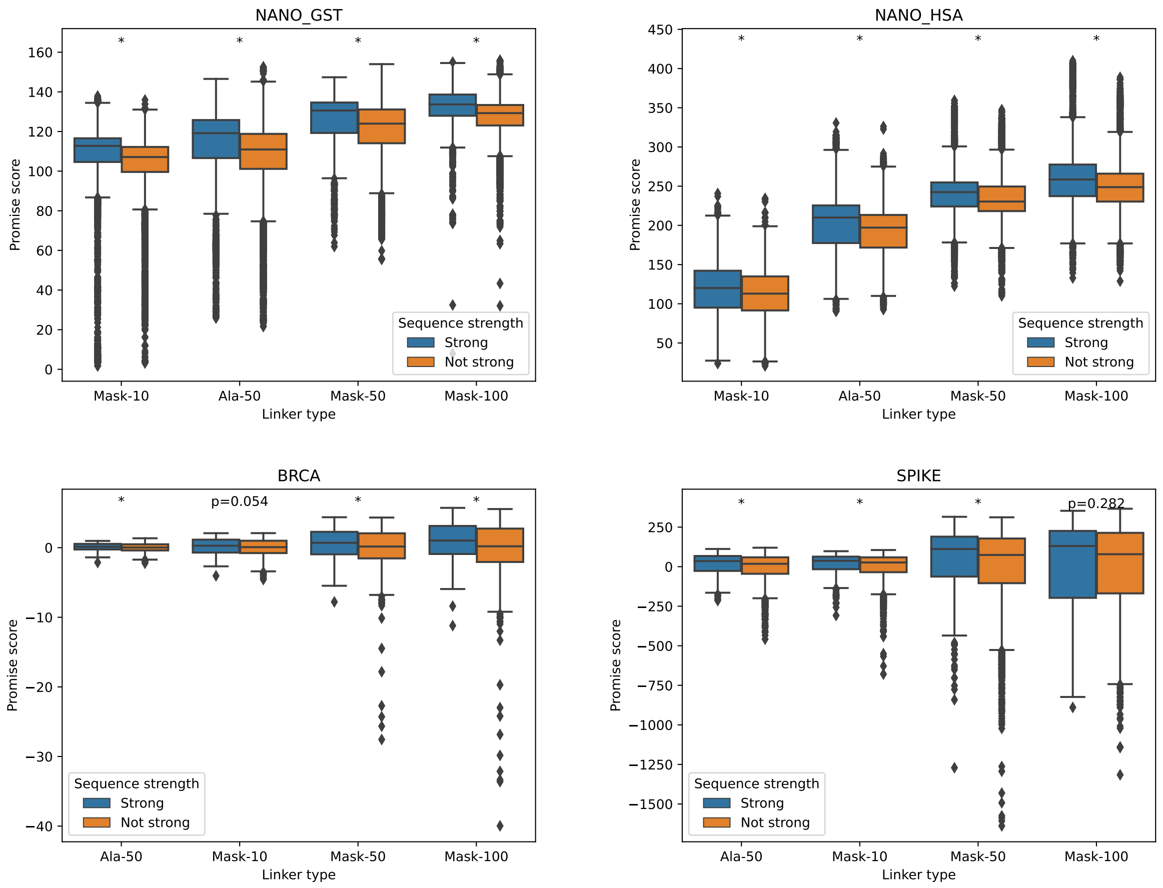


Figure B.2: We calculated PROMISE SCORES for each sequence identified within two Nb discovery campaigns (Top) and two protein optimization campaigns (Bottom), and compare how “strong” sequences scored compared to those that were “not strong”. In each case, strong sequences had an average PROMISE SCORE greater than sequences that were not strong. These differences were either statistically significant ($p < 0.05$, Mann-Whitney U-test, denoted with *) or had a p-value just above the 0.05 threshold (BRCA mask-10— $p = 0.054$), with the only exception when using a 100 length mask linker with Spike ($p = 0.282$).

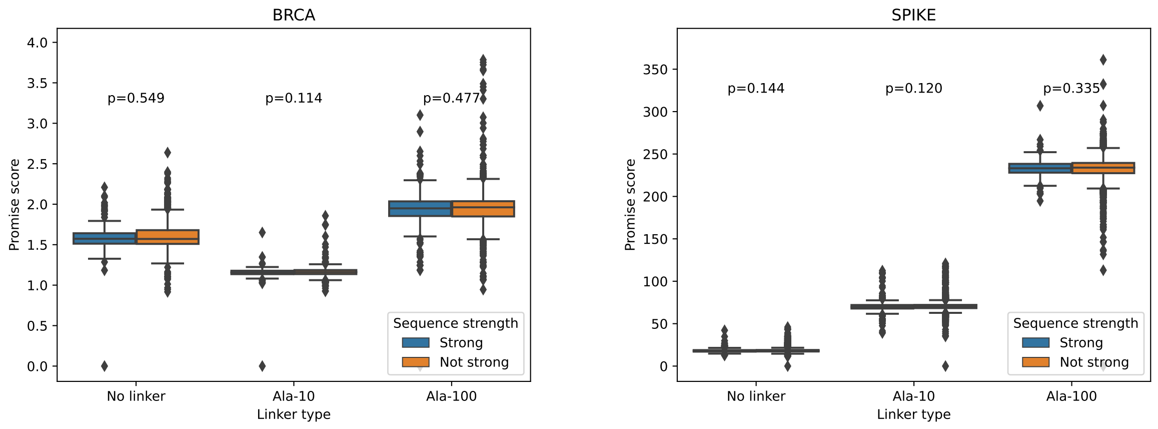


Figure B.3: Promise Scores do not differentiate "strong" and "not strong" sequences ($p \geq 0.05$, Mann-Whitney U-test) during protein optimization when the S_{intra} term is excluded from the objective function (Equation 5.2.2).

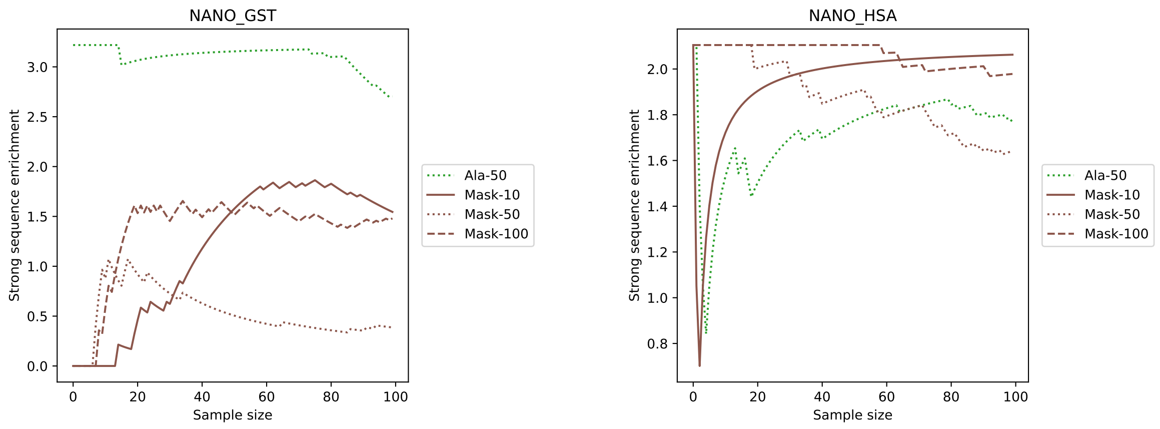


Figure B.4: We used Promise scores to help identify potential lead sequences from two Nb discovery campaigns. Here, we show the strong sequence enrichment when using Promise scores obtained with mask linkers of length 10, 50, and 100, as well as a length 50 alanine linker. With Nano-GST, we found that alanine linkers yielded higher strong sequence enrichment than mask linkers. With Nano-HSA, we found that all linker types yielded comparable strong sequence enrichments.

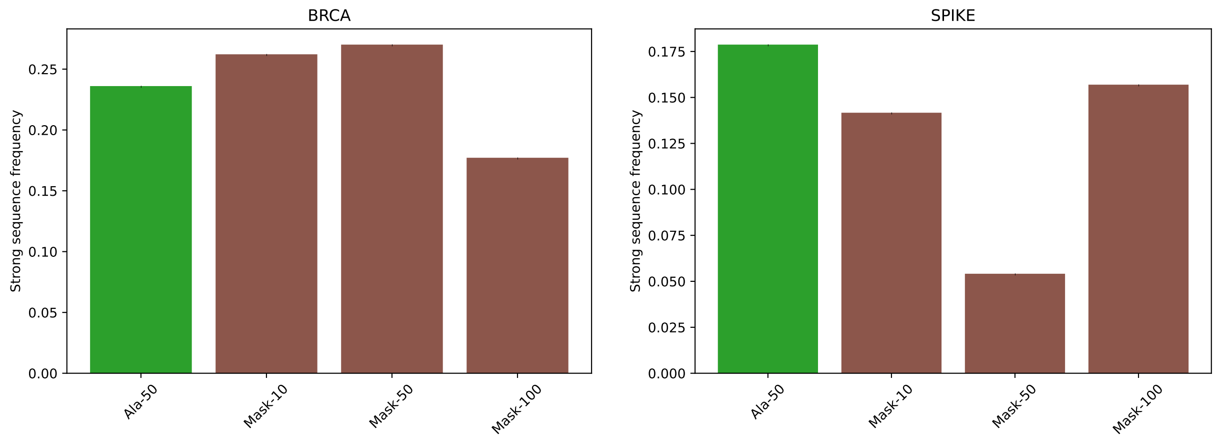


Figure B.5: We used PROMISE SCORES to select 10 site-specific mutagenesis experiments on each of BRCA1 and Spike. With BRCA1, using length 10 or 50 mask linkers yield strong sequence frequencies comparable to those when using alanine linkers. With Spike, the mask linker yields more variable results compared to alanine linkers at different linker lengths.

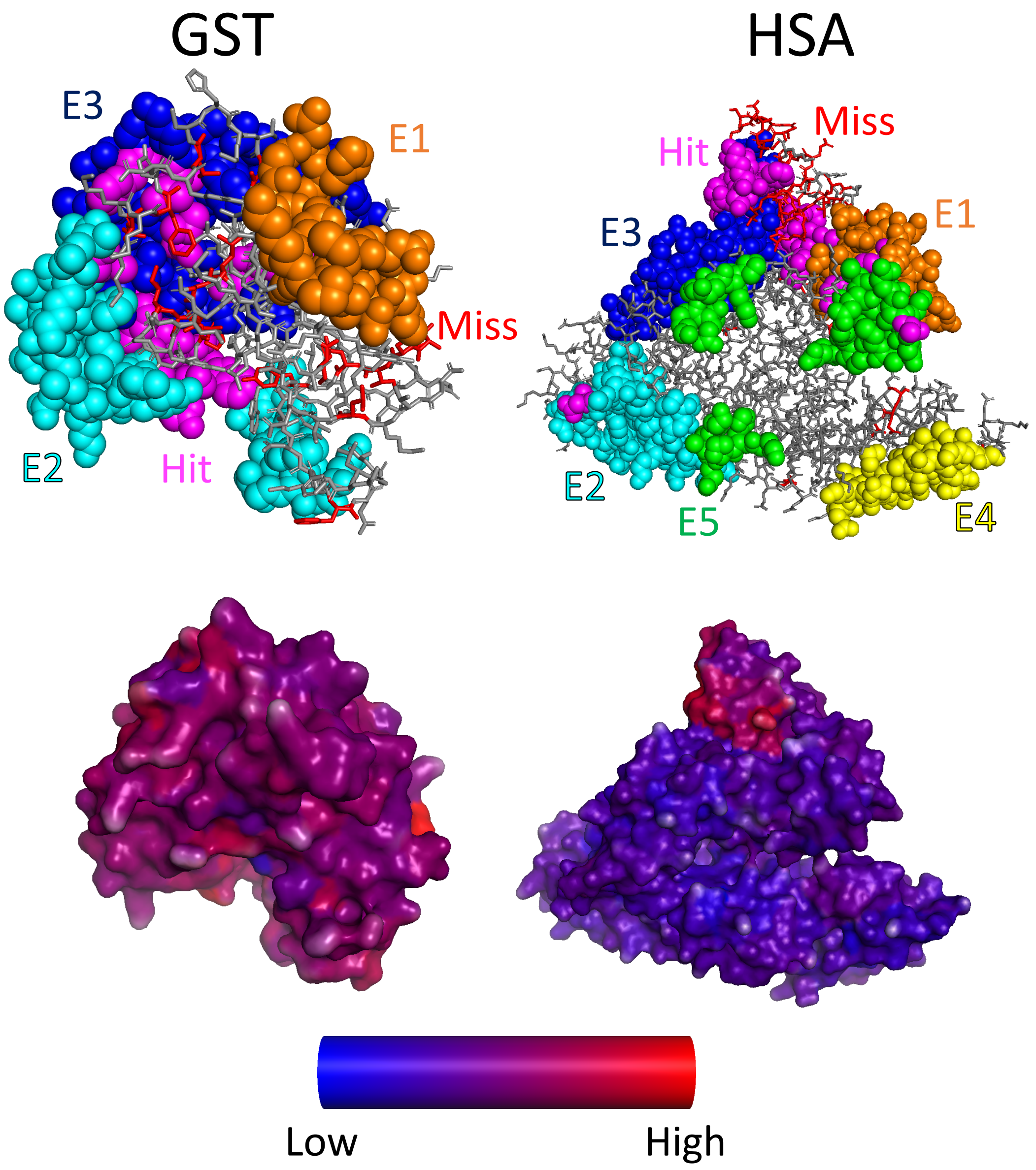


Figure B.6: These are the same structures shown in Figure 5.5, but rotated 180° about the y-axis. We use intermolecular contact map $\mathcal{A}_{inter}$ to calculate residue-specific scores to identify residues that contribute the most to intermolecular interactions. With Nb discovery, we highlight overlap between our top scoring residues and epitopes previously validated through cross-linking mass spectrometry (Top). The magenta spheres indicate this overlap. Each other colored sphere indicates a residue within a validated epitope region. The red sticks correspond to top scoring sites that did not fall in any validated region (denoted "Miss"), and the gray sticks indicate all other residues within the protein sequence. (Bottom) The heatmaps show the relative values of the residue-specific scores across the entire target protein. The coloring scale is normalized for each individual protein to show the relative scores of residues on a given protein (i.e. red shaded residues on GST are the highest scoring residues on that protein, but may have different PROMISE SCORES than the red shaded HSA residues). The structures we used are given by PDB IDs 1DUG and 1AO6.

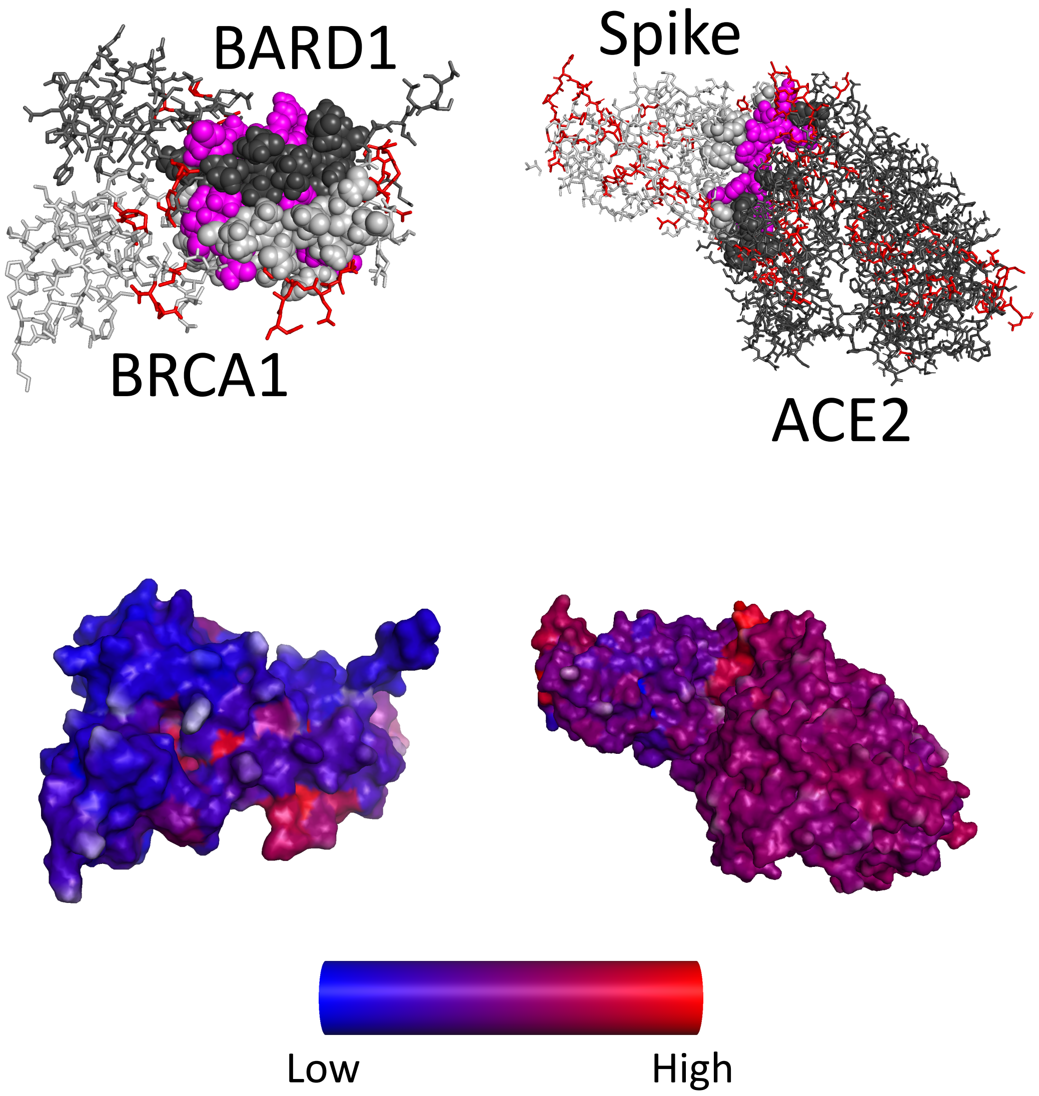


Figure B.7: These are the same structures shown in Figure 5.6, but rotated 180° about the y-axis. We use intermolecular contact map $\mathcal{A}_{inter}$ to calculate residue-specific scores to identify residues that contribute the most to intermolecular interactions. With protein optimization, we show structures of each protein bound to its target. (Top) The spheres indicate residues known to play a role in protein-target binding. Those shaded magenta were identified by the per-residue scoring. Red sticks indicate residues outside this binding region that were also identified. Dark gray sticks indicate s_{target} residues, and light gray s_{binder} residues. (Bottom) The heatmaps show the relative values of the residue-specific scores across the entire s_{binder}-s_{target} complex. The coloring scale is normalized for each individual protein to show the relative scores of residues on a given protein (i.e. red shaded residues on BRCA1 are the highest scoring residues on that protein, but may have different PROMISE SCORES than the red shaded BARD1, Spike, or ACE2 residues). The structures we used are given by PDB IDs 1JM7 and 6M0J.

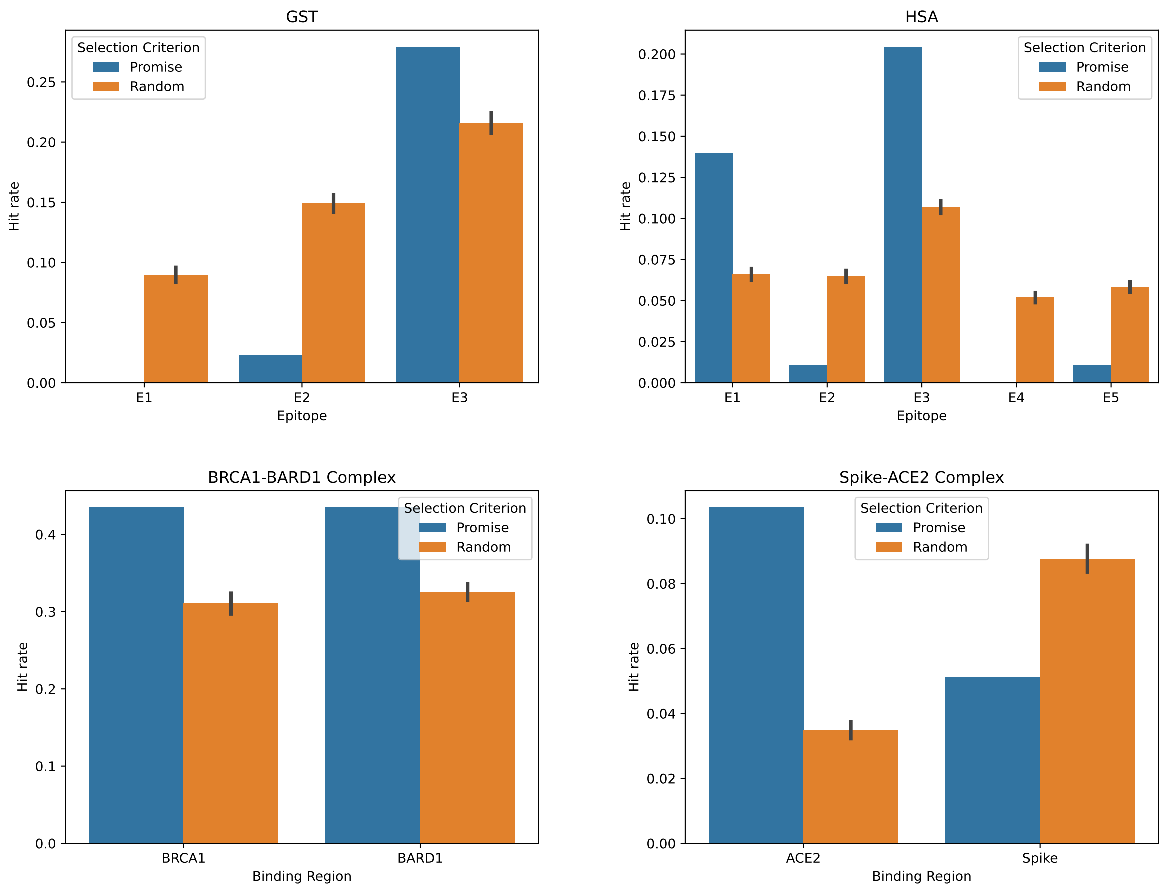


Figure B.8: The frequency (i.e. "Hit rate") at which PROMISE SCORE selected and randomly selected residues fall within a known binding region for a given protein system. The hit rate obtained via random selection is given as a mean over 50 replicates ± 1 SEM.

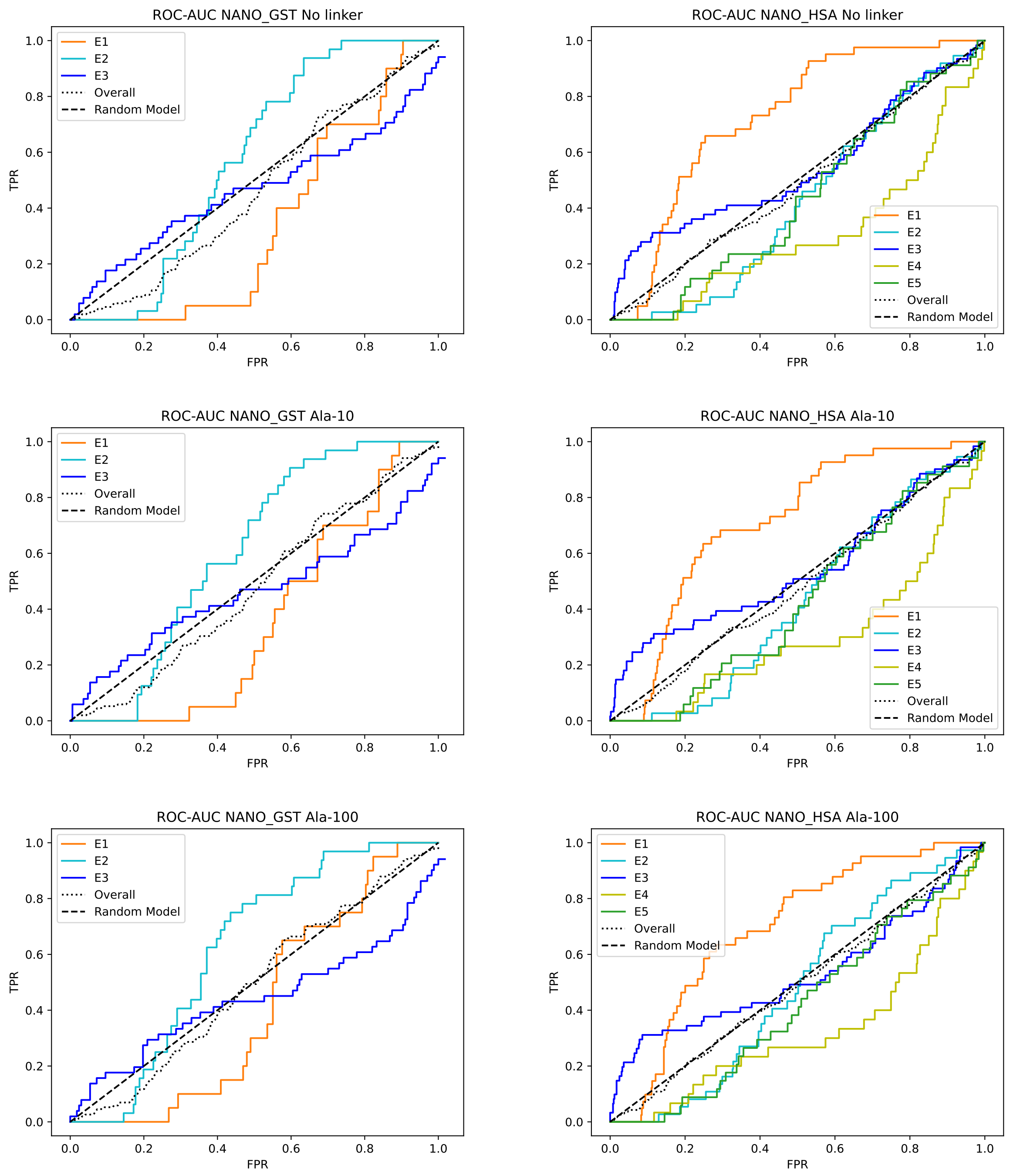


Figure B.9: ROC-AUC curves showing the PROMISE SCORE's ability to predict residues involved in epitope binding using the Nano-GST (left) and Nano-HSA (right) repertoires. From top to bottom, we show the effect of using either no linker, a length 10 Alanine linker, or a length 100 Alanine linker. While the PROMISE SCORE has some predictive ability for select epitope regions, it is a poor predictor of *general* epitope binding.

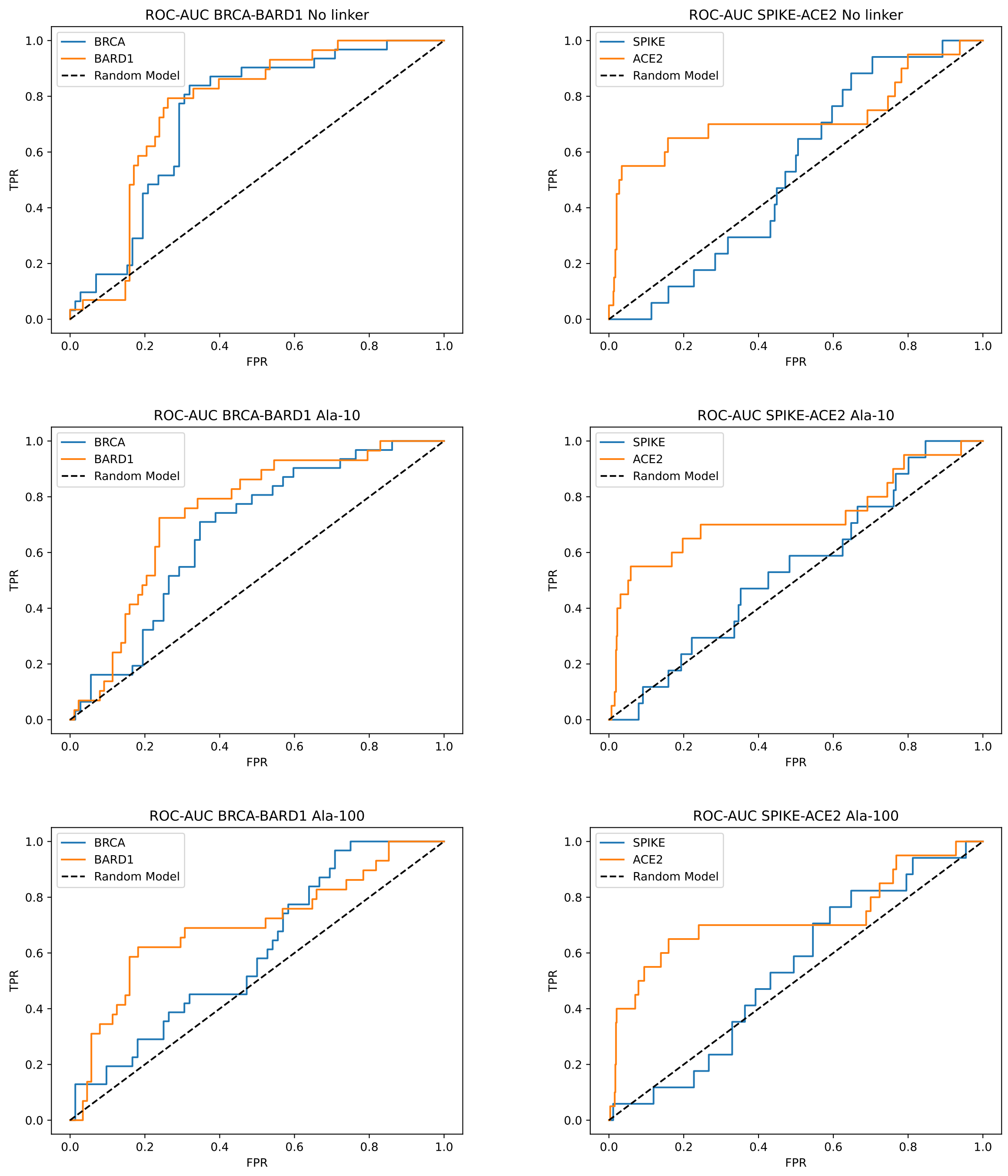


Figure B.10: ROC-AUC curves showing the Promise Score's ability to predict residues involved in protein binding using the BRCA1-BARD1 (left) and Spike-ACE2 (right) protein complexes. Promise Scores are generally predictive within the BRCA1-BARD1 complex (especially with no linker or a length 10 Alanine linker), as well as with ACE2. Promise Scores are generally not predictive with Spike.

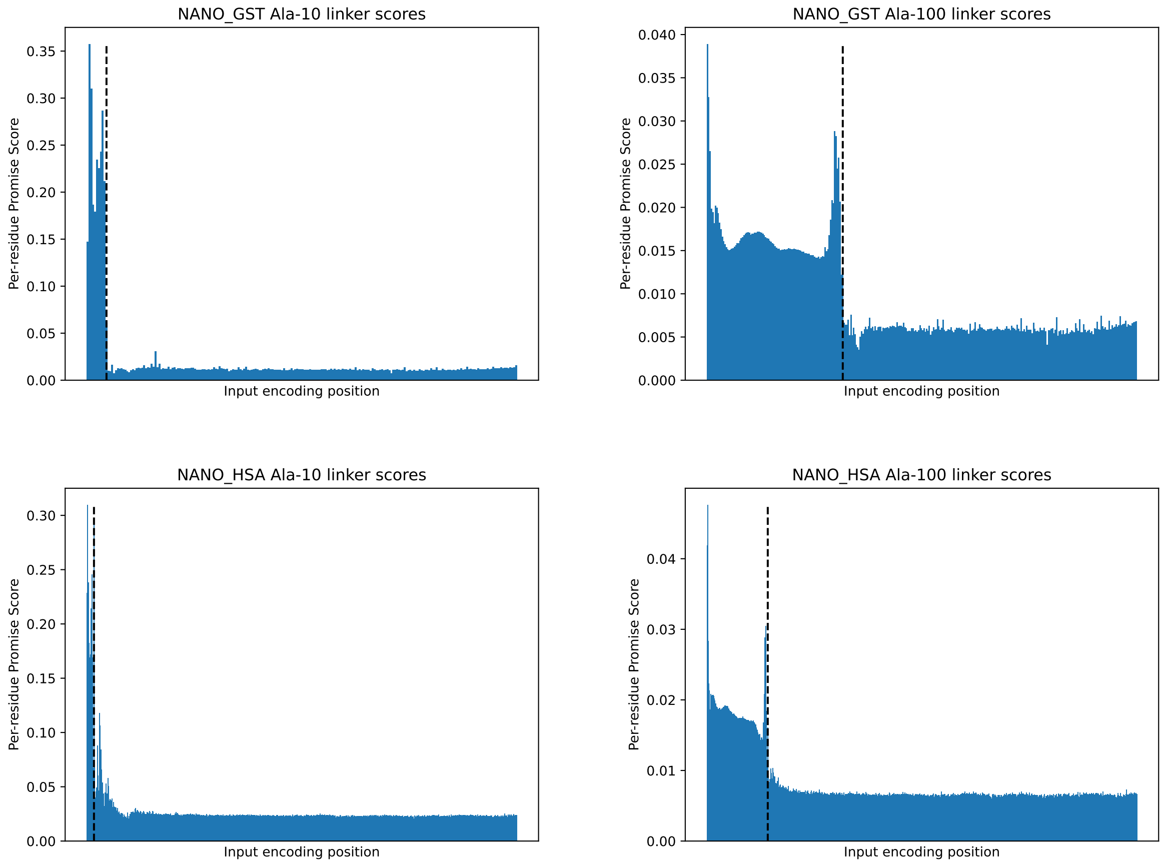


Figure B.11: The interactions between Alanine linkers and the s_{binder}-s_{linker}-s_{target} sequence encoding according to the attention map for two Nb repertoires. The per-residue Promise Scores left of the vertical dashed line shows predicted interactions of s_{linker} to itself. Promise Scores to the right of the dashed line shows predicted interactions between s_{linker} and s_{target}.

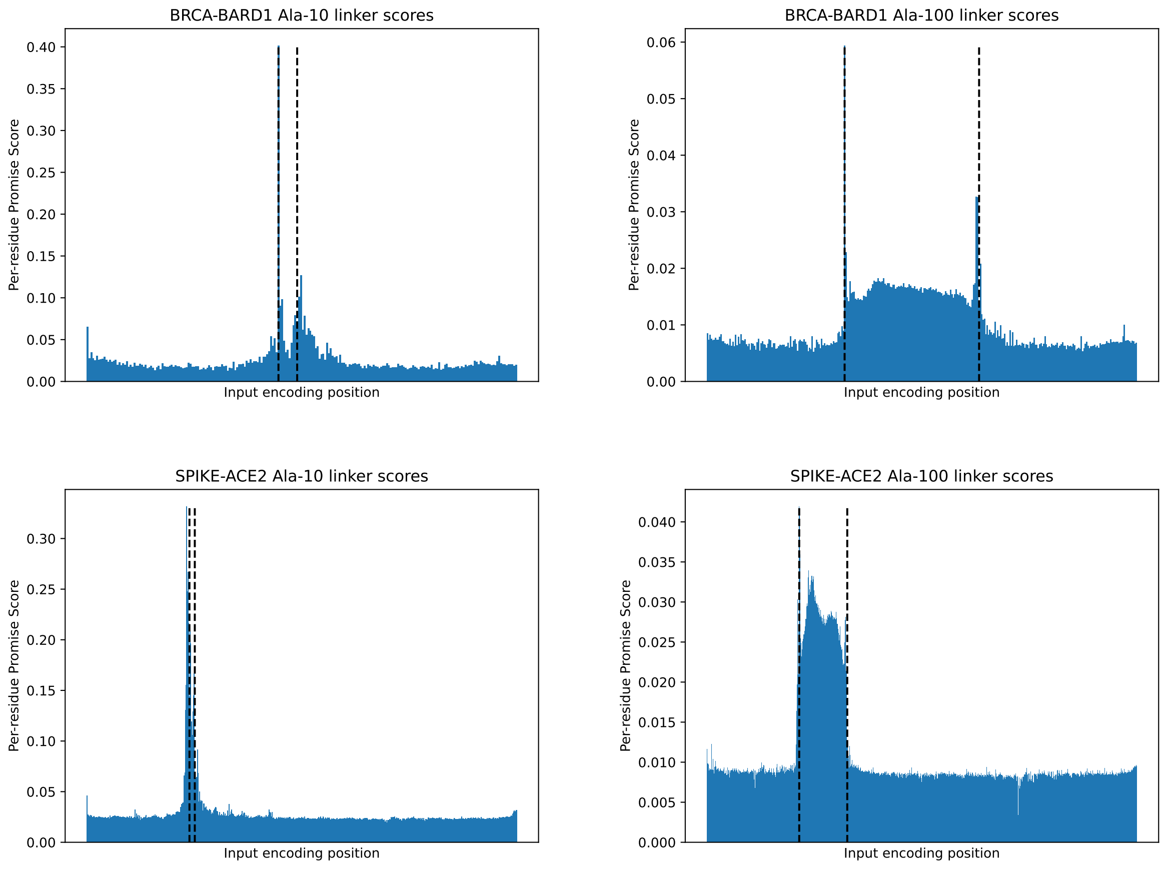


Figure B.12: The interactions between Alanine linkers and the s_{binder}-s_{linker}-s_{target} sequence encoding according to the attention map for the BRCA1-BARD1 and Spike-ACE2 protein complexes. The per-residue PROMISE SCORES left of the vertical dashed line shows predicted interactions between s_{linker} and s_{binder}. PROMISE SCORES in between the vertical dashed lines shows predicted interactions of s_{linker} with itself. PROMISE SCORES to the right of the dashed line shows predicted interactions between s_{linker} and s_{target}.

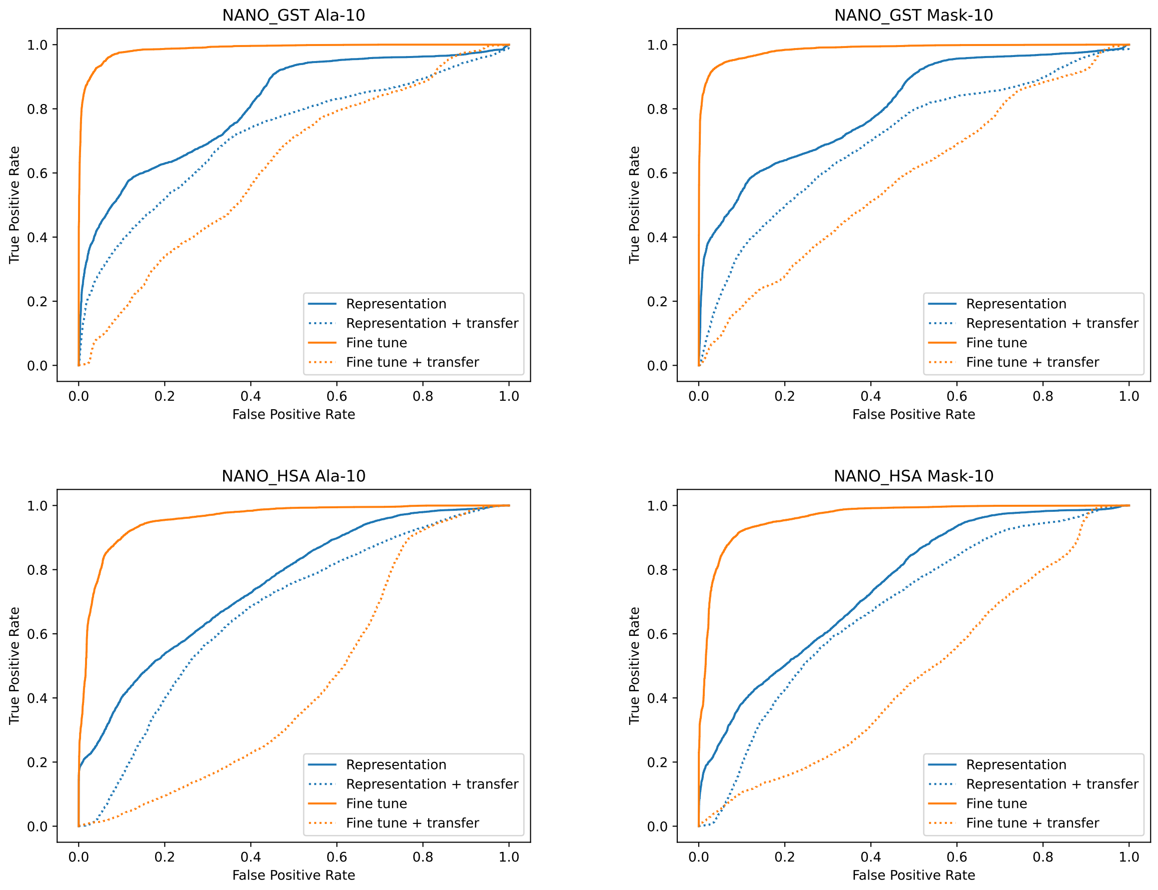


Figure B.13: ROC curves for models trained to classify strong and weak sequences with Nb discovery. The top row shows Nano-GST, and the bottom Nano-HSA. The left column shows models trained with a length 10 alanine linker, and the right columns shows models trained with a length ten mask linker. Orange curves correspond to models that used fine-tuning, and blue those that did not. Solid curves correspond to models applied in a traditional machine learning paradigm, whereas dashed lines used transfer learning. Fine-tuned models used in the traditional setting were very strong, regardless of linker type or length.

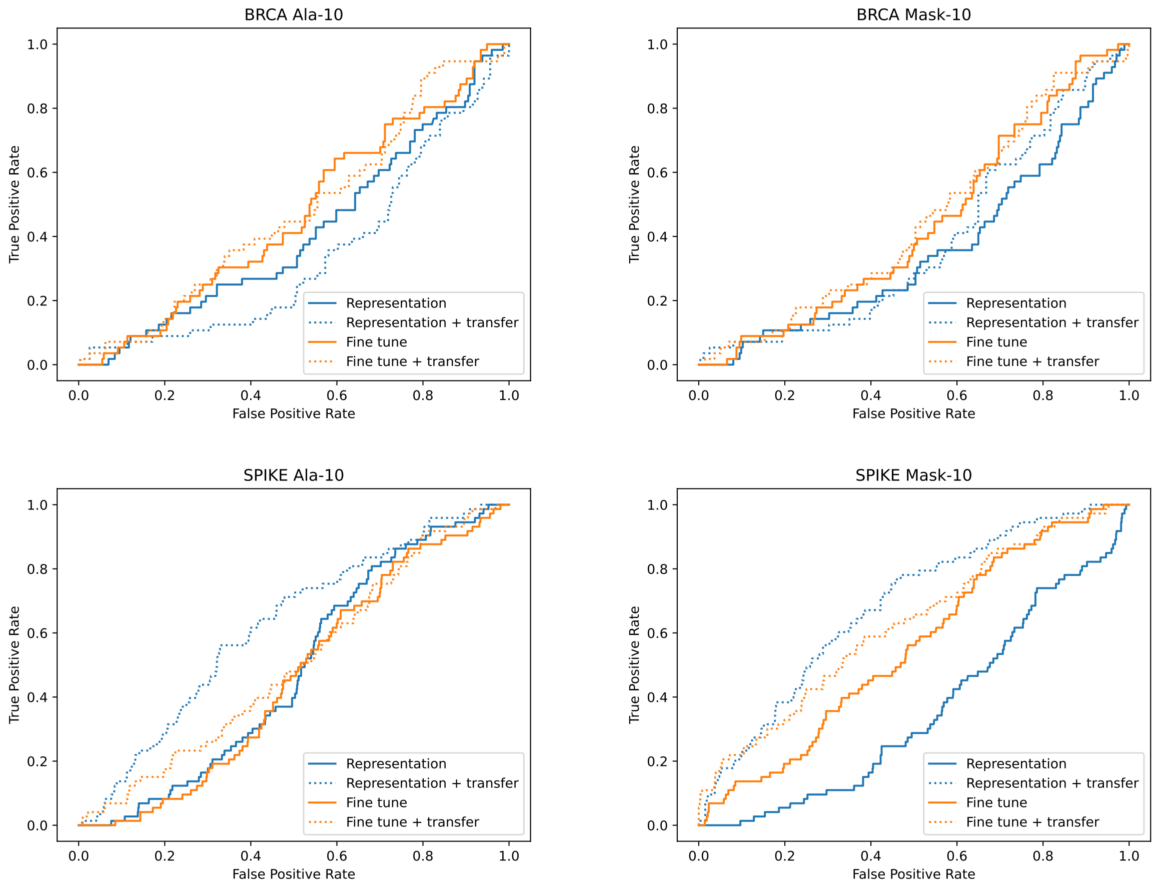


Figure B.14: ROC curves for models trained to classify strong and weak sequences in protein optimization. The top row shows BRCA1, and the bottom Spike. The left column shows models trained with a length 10 alanine linker, and the right columns shows models trained with a length ten mask linker. Orange curves correspond to models that used fine-tuning, and blue those that did not. Solid curves correspond to models applied in a traditional machine learning paradigm, whereas dashed lines used transfer learning. While all models learned in protein optimization were of lower quality, we saw evidence that transfer learner led to model improvement.

Bibliography

[1] Lukas C. Gerber et al. "Liquid-handling Lego robots and experiments for STEM education and research". In: *PLOS Biology* 15.3 (Mar. 2017), pp. 1–9.

[2] Miroslav Pohanka et al. "Automated assay of the potency of natural antioxidants using pipetting robot and spectrophotometry". In: *Journal of Applied Biomedicine* 10.3 (2012), pp. 155–167.

[3] Tom Alisch et al. "MAPLE (modular automated platform for large-scale experiments), a robot for integrated organism-handling and phenotyping". In: *eLife* 7 (Aug. 2018), e37166.

[4] Mathew M Jessop-Fabre and Nikolaus Sonnenschein. "Improving Reproducibility in Synthetic Biology". In: *Frontiers in Bioengineering and Biotechnology* 7 (2019), p. 18.

[5] David Waltz and Bruce G. Buchanan. "Automating Science". In: *Science* 324.5923 (2009), pp. 43–44.

[6] Ross D. King et al. "Functional genomic hypothesis generation and experimentation by a robot scientist". In: *Nature* 427.6971 (2004), pp. 247–252.

[7] Ying Liu. "Active Learning with Support Vector Machine Applied to Gene Expression Data for Cancer Classification". In: *Journal of Chemical Information and Computer Sciences* 44.6 (Nov. 2004), pp. 1936–1941.

[8] Samuel A. Danziger et al. "Predicting positive p53 cancer rescue regions using Most Informative Positive (MIP) active learning". In: *PLoS computational biology* 5.9 (Sept. 2009), e1000498–e1000498.

[9] Anindya Halder and Ansuman Kumar. "Active learning using rough fuzzy classifier for cancer prediction from microarray gene expression data". In: *Journal of Biomedical Informatics* 92 (2019), p. 103136.

[10] Iiris Sundin et al. "Improving genomics-based predictions for precision medicine through active elicitation of expert knowledge". In: *Bioinformatics* 34.13 (June 2018), pp. i395–i403.

[11] Manfred K. Warmuth et al. "Active Learning with Support Vector Machines in the Drug Discovery Process". In: *Journal of Chemical Information and Computer Sciences* 43.2 (Mar. 2003), pp. 667–673.

[12] Daniel Reker and Gisbert Schneider. "Active-learning strategies in computer-assisted drug discovery". In: *Drug Discovery Today* 20.4 (2015), pp. 458–465.

[13] Jocelyn Duffy and Toby Blackburn. *Carnegie Mellon University and Emerald Cloud Lab to Build World's First University Cloud Lab*. URL: https://www.cmu.edu/news/stories/archives/2021/august/first-academic-cloud-lab.html.

[14] Chase Armer, Florent Letronne, and Erika DeBenedictis. "Support academic access to automated cloud labs to improve reproducibility". In: *PLOS Biology* 21.1 (Jan. 2023), pp. 1–4.

[15] Michael Segal. "An operating system for the biology lab". In: *Nature* 573 (2019), S112–S113.

[16] Ben Miles and Peter L. Lee. "Achieving Reproducibility and Closed-Loop Automation in Biological Experimentation with an IoT-Enabled Lab of the Future". In: *SLAS TECHNOLOGY: Translating Life Sciences Innovation* 23.5 (2018), pp. 432–439.

[17] Ian Holland and Jamie A. Davies. "Automation in the Life Science Research Laboratory". In: *Frontiers in Bioengineering and Biotechnology* 8 (2020).

[18] Muhammad Bilal et al. "State-of-the-art protein engineering approaches using biological macromolecules: A review from immobilization to implementation view point". In: *International Journal of Biological Macromolecules* 108 (2018), pp. 893–901.

[19] David Brookes, Hahnbeom Park, and Jennifer Listgarten. "Conditioning by adaptive sampling for robust design". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 773–782.

[20] Anvita Gupta and James Zou. "Feedback GAN for DNA optimizes protein functions". In: *Nature Machine Intelligence* 1.2 (Feb. 2019), pp. 105–111.

[21] Trevor S. Frisby, Zhiyun Gong, and Christopher James Langmead. "Asynchronous parallel Bayesian optimization for AI-driven cloud laboratories". In: *Bioinformatics* 37.Suppl_1 (July 2021), pp. i451–i459.

[22] Trevor S. Frisby and Christopher James Langmead. "Bayesian optimization with evolutionary and structure-based regularization for directed protein evolution". In: *Algorithms for Molecular Biology* 16.1 (July 2021), p. 13.

[23] Trevor S. Frisby and Christopher James Langmead. "Fold Family-Regularized Bayesian Optimization for Directed Protein Evolution". In: *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Vol. 172. Leibniz International Proceedings in Informatics (LIPIcs). 2020, 18:1–18:17.

[24] Trevor S. Frisby and Christopher James Langmead. "Identifying promising sequences for protein engineering using a deep Transformer Protein Language Model". In: *bioRxiv* (2023).

[25] Trevor S. Frisby et al. "Harvestman: a framework for hierarchical feature learning and selection from whole genome sequencing data". In: *BMC Bioinformatics* 22.1 (Apr. 2021).

[26] M. K. K. Leung et al. "Machine Learning in Genomic Medicine: A Review of Computational Problems and Data Sets". In: *Proceedings of the IEEE* 104.1 (Jan. 2016), pp. 176–197.

[27] Rahul C. Deo. "Machine Learning in Medicine". In: *Circulation* 132.20 (2015), pp. 1920–1930.

[28] Burr Settles. *Active learning literature survey*. Tech. rep. 2010.

[29] Fridolin Gross. "Heuristic Strategies in Systems Biology". In: *HUMANA.MENTE Journal of Philosophical Studies* 9.30 (June 2016), pp. 1–18.

[30] James Wilson, Frank Hutter, and Marc Deisenroth. "Maximizing acquisition functions for Bayesian optimization". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., Dec. 2018, pp. 9884–9895.

[31] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization*. Ed. by Carlos A. Coello Coello. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[32] James Bergstra et al. "Algorithms for Hyper-Parameter Optimization". In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. NIPS'11. Granada, Spain: Curran Associates Inc., 2011, pp. 2546–2554.

[33] Maarten Rubens et al. "Precise polymer synthesis by autonomous self-optimizing flow reactors". In: *Angewandte Chemie - International Edition* 58.10 (Mar. 2019), pp. 3183–3187.

[34] Jonas Mockus. *Bayesian Approach to Global Optimization: Theory and Applications*. en. Vol. 37. Mathematics and Its Applications. Dordrecht: Springer Netherlands, 1989.

[35] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012.

[36] Daniel J. Lizotte et al. "Automatic Gait Optimization with Gaussian Process Regression." In: *IJCAI*. 2007, pp. 944–949.

[37] P. Ilten, M. Williams, and Y. Yang. "Event generator tuning using Bayesian optimization". In: *Journal of Instrumentation* 12.04 (Apr. 2017), P04028–P04028.

[38] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005.

[39] Dipti Jasrasaria and Edward O. Pyzer-Knapp. *Dynamic Control of Explore/Exploit Trade-Off In Bayesian Optimization*. 2018.

[40] Andrew Ng and Michael Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes". In: *Advances in neural information processing systems* 14 (2001).

[41] P Baldi et al. "Hidden Markov models of biological primary sequence information." In: *Proceedings of the National Academy of Sciences* 91.3 (1994), pp. 1059–1063.

[42] S. Balakrishnan et al. "Learning Generative Models for Protein Fold Families". In: *Proteins: Structure, Function, and Bioinformatics* 79.6 (2011), pp. 1061–1078.

[43] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013.

[44] Ian Goodfellow et al. "Generative Adversarial Networks". In: *Commun. ACM* 63.11 (Oct. 2020), pp. 139–144.

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

[46] Emerald Cloud Lab Inc. *Emerald Cloud Lab*. URL: https://www.emeraldcloudlab.com/.

[47] Strateos Inc. *Strateos*. URL: https://strateos.com/.

[48] Kenji Kawaguchi, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Bayesian Optimization with Exponential Convergence". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015, pp. 2809–2817.

[49] Rémi Munos. "Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness". In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., 2011, pp. 783–791.

[50] Ziyu Wang et al. "Bayesian Multi-Scale Optimistic Optimization". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, Apr. 2014, pp. 1005–1014.

[51] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. "Lipschitzian optimization without the Lipschitz constant". In: *Journal of Optimization Theory and Applications* 79.1 (Oct. 1993), pp. 157–181.

[52] Sébastien Bubeck and Nicolò Cesa-Bianchi. "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems". In: *Foundations and Trends® in Machine Learning* 5.1 (2012), pp. 1–122.

[53] Jeffrey M. Spraggins et al. "Next-generation technologies for spatial proteomics: Integrating ultra-high speed MALDI-TOF and high mass resolution MALDI FTICR imaging mass spectrometry for protein analysis". In: *PROTEOMICS* 16.11-12 (2016), pp. 1678–1689.

[54] Kyoung-Soon Jang and Young Hwan Kim. "Rapid and robust MALDI-TOF MS techniques for microbial identification: a brief overview of their diverse applications". In: *Journal of Microbiology* 56.4 (Apr. 2018), pp. 209–216.

[55] Chad S. Cummings et al. "Design of Stomach Acid-Stable and Mucin-Binding Enzyme Polymer Conjugates". In: *Biomacromolecules* 18.2 (2017), pp. 576–586.

[56] Bibifatima Kaupbayeva and Alan J. Russell. "Polymer-enhanced biomacromolecules". In: *Progress in Polymer Science* 101 (2020), p. 101194.

[57] Kirthevasan Kandasamy et al. *Asynchronous Parallel Bayesian Optimisation via Thompson Sampling*. 2017.

[58] Mukthi Thammana. *A Review on High Performance Liquid Chromatography (HPLC)*. 2016.

[59] Winston Chang et al. *shiny: Web Application Framework for R*. R package version 1.5.0. 2020.

[60] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. "Batched Gaussian Process Bandit Optimization via Determinantal Point Processes". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4213–4221.

[61] Stefan Lutz and Uwe Theo Bornscheuer. *Protein Engineering Handbook*. English. Weinheim: Wiley-VCH, 2012.

[62] Janes S. Richardson and David C. Richardson. "The de novo design of protein structures". In: *Trends in Biochemical Sciences* 14.7 (July 1989), pp. 304–309.

[63] Frances H. Arnold. "Directed Evolution: Bringing New Chemistry to Life". In: *Angewandte Chemie International Edition* 57.16 (2018), pp. 4143–4148.

[64] Zachary Wu et al. "Machine learning-assisted directed protein evolution with combinatorial libraries". In: *Proceedings of the National Academy of Sciences* 116.18 (2019), pp. 8852–8858.

[65] Tyler N. Starr and Joseph W. Thornton. "Epistasis in protein evolution". In: *Protein science : a publication of the Protein Society* 25.7 (July 2016), pp. 1204–1218.

[66] Alexander Rives et al. “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”. In: *Proceedings of the National Academy of Sciences* 118.15 (2021), e2016239118.

[67] Anders Krogh et al. “Hidden Markov Models in Computational Biology”. In: *Journal of Molecular Biology* 235.5 (Feb. 1994), pp. 1501–1531.

[68] Joost Schymkowitz et al. “The FoldX web server: an online force field”. In: *Nucleic Acids Research* 33.suppl_2 (July 2005), W382–W388.

[69] Pietro Gatti-Lafranconi et al. “Evolution of Stability in a Cold-Active Enzyme Elicits Specificity Relaxation and Highlights Substrate-Related Effects on Temperature Adaptation”. In: *Journal of Molecular Biology* 395.1 (Jan. 2010), pp. 155–166.

[70] Robert E. Hawkins, Stephen J. Russell, and Greg Winter. “Selection of phage antibodies by binding affinity”. In: *Journal of Molecular Biology* 226.3 (Aug. 1992), pp. 889–896.

[71] Lars Giger et al. “Evolution of a designed retro-aldolase leads to complete active site remodeling”. In: *Nature Chemical Biology* 9.8 (Aug. 2013), pp. 494–498.

[72] Fathima Aidha Shaikh and Stephen G. Withers. “Teaching old enzymes new tricks: engineering and evolution of glycosidases and glycosyl transferases for improved glycoside synthesis”. In: *Biochemistry and Cell Biology* 86.2 (Apr. 2008), pp. 169–177.

[73] Philip A. Romero and Frances H. Arnold. “Exploring protein fitness landscapes by directed evolution”. In: *Nature Reviews Molecular Cell Biology* 10.12 (Dec. 2009), pp. 866–876.

[74] Kevin K. Yang, Zachary Wu, and Frances H. Arnold. “Machine-learning-guided directed evolution for protein engineering”. In: *Nature Methods* 16.8 (Aug. 2019), pp. 687–694.

[75] Adi Goldenzweig and Sarel J. Fleishman. "Principles of Protein Stability and Their Application in Computational Design". In: *Annual Review of Biochemistry* 87.1 (2018), pp. 105–129.

[76] B. Kuhlman and D. Baker. "Native protein sequences are close to optimal for their structures". In: *Proceedings of the National Academy of Sciences of the United States of America* 97.19 (Sept. 2000), pp. 10383–10388.

[77] Marziyeh Movahedi, Fatemeh Zare-Mirakabad, and Seyed Shahriar Arab. "Evaluating the accuracy of protein design using native secondary sub-structures". In: *BMC Bioinformatics* 17.1 (Sept. 2016), p. 353.

[78] Nicholas C Wu et al. "Adaptation in protein fitness landscapes is facilitated by indirect paths". In: *eLife* 5 (July 2016), e16965.

[79] C. Anders Olson, Nicholas C. Wu, and Ren Sun. "A Comprehensive Biophysical Description of Pairwise Epistasis throughout an Entire Protein Domain". In: *Current Biology* 24.22 (Nov. 2014), pp. 2643–2651.

[80] Richard W. Roberts and Jack W. Szostak. "RNA-peptide fusions for the in vitro selection of peptides and proteins". In: *Proceedings of the National Academy of Sciences* 94.23 (1997), pp. 12297–12302.

[81] Serena Clark et al. "Structure-Function of the Tumor Suppressor BRCA1". In: *Computational and structural biotechnology journal* 1 (Apr. 2012).

[82] L. M. Starita et al. "Massively Parallel Functional Analysis of BRCA1 RING Domain Variants". In: *Genetics* 200.2 (June 2015), pp. 413–422.

[83] L. M. Starita et al. "Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis". In: *Proceedings of the National Academy of Sciences* 110.14 (Mar. 2013), E1263–E1272.

[84] Tyler 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.

[85] Jesse D. Bloom. "An Experimentally Determined Evolutionary Model Dramatically Improves Phylogenetic Fit". In: *Molecular Biology and Evolution* 31.8 (May 2014), pp. 1956–1978.

[86] The UniProt Consortium. "UniProt: the universal protein knowledgebase in 2021". In: *Nucleic Acids Research* 49.D1 (Nov. 2020), pp. D480–D489.

[87] Sara El-Gebali et al. "The Pfam protein families database in 2019". In: *Nucleic Acids Research* 47.D1 (Oct. 2018), pp. D427–D432.

[88] Helen M. Berman et al. "The Protein Data Bank". In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 235–242.

[89] Ali Madani et al. "ProGen: Language Modeling for Protein Generation". In: *bioRxiv* (2020).

[90] Paul J Carter. "Introduction to current and future protein therapeutics: a protein engineering perspective". In: *Experimental cell research* 317.9 (2011), pp. 1261–1269.

[91] Mark L Chiu et al. "Antibody structure and function: the basis for engineering therapeutics". In: *Antibodies* 8.4 (2019), p. 55.

[92] Ruei-Min Lu et al. "Development of therapeutic antibodies for the treatment of diseases". In: *Journal of biomedical science* 27.1 (2020), pp. 1–30.

[93] Brian Kelley. "Developing therapeutic monoclonal antibodies at pandemic pace". In: *Nature biotechnology* 38.5 (2020), pp. 540–545.

[94] Shivcharan Prasad and Ipsita Roy. "Converting enzymes into tools of industrial importance". In: *Recent Pat. Biotechnol.* 12.1 (2018), pp. 33–56.

[95] Jing Mu et al. "Engineering of nanoscale coordination polymers with biomolecules for advanced applications". In: *Coord. Chem. Rev.* 399.213039 (Nov. 2019), p. 213039.

[96] Grazia M Borrelli and Daniela Trono. “Recombinant lipases and phospholipases and their use as biocatalysts for industrial applications”. In: *Int. J. Mol. Sci.* 16.9 (Sept. 2015), pp. 20774–20840.

[97] Baotong Zhu, Dong Wang, and Na Wei. “Enzyme discovery and engineering for sustainable plastic recycling”. In: *Trends in biotechnology* 40.1 (2022), pp. 22–37.

[98] Hyeoncheol Francis Son et al. “Rational protein engineering of thermo-stable PETase from Ideonella sakaiensis for highly efficient PET degradation”. In: *ACS Catalysis* 9.4 (2019), pp. 3519–3526.

[99] George J Weiner. “Building better monoclonal antibody-based therapeutics”. In: *Nature Reviews Cancer* 15.6 (2015), pp. 361–370.

[100] Louis M Weiner, Joseph C Murray, and Casey W Shuptrine. “Antibody-based immunotherapy of cancer”. In: *Cell* 148.6 (2012), pp. 1081–1084.

[101] Dennis R Goulet and William M Atkins. “Considerations for the design of antibody-based therapeutics”. In: *Journal of pharmaceutical sciences* 109.1 (2020), pp. 74–103.

[102] Rahmad Akbar et al. “A compact vocabulary of paratope-epitope interactions enables predictability of antibody-antigen binding”. In: *Cell Rep.* 34.11 (Mar. 2021), p. 108856.

[103] Inbal Sela-Culang, Vered Kunik, and Yanay Ofran. “The Structural Basis of Antibody-Antigen Recognition”. In: *Frontiers in Immunology* 4 (2013).

[104] Emily Y. Yang and Khalid Shah. “Nanobodies: Next Generation of Cancer Diagnostics and Therapeutics”. In: *Frontiers in Oncology* 10 (2020).

[105] Ivana Jovčevska and Serge Muyldermans. “The therapeutic potential of nanobodies”. In: *BioDrugs* 34.1 (Feb. 2020), pp. 11–26.

[106] Ivan V Korendovych and William F DeGrado. “De novo protein design, a retrospective”. In: *Q. Rev. Biophys.* 53.e3 (Feb. 2020), e3.

[107] Xingjie Pan and Tanja Kortemme. “Recent advances in de novo protein design: Principles, methods, and applications”. In: *Journal of Biological Chemistry* 296 (2021), p. 100558.

[108] Rodrigo MP Siloto and Randall J Weselake. “Site saturation mutagenesis: Methods and applications in protein engineering”. In: *Biocatalysis and Agricultural Biotechnology* 1.3 (2012), pp. 181–189.

[109] M M Ling and B H Robinson. “Approaches to DNA mutagenesis: an overview”. In: *Anal. Biochem.* 254.2 (Dec. 1997), pp. 157–178.

[110] J Braman, C Papworth, and A Greener. “Site-directed mutagenesis using double-stranded plasmid DNA templates”. In: *Methods Mol. Biol.* 57 (1996), pp. 31–44.

[111] W Wang and B A Malcolm. “Two-stage PCR protocol allowing introduction of multiple mutations, deletions and insertions using QuikChange Site-Directed Mutagenesis”. In: *Biotechniques* 26.4 (Apr. 1999), pp. 680–682.

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

[113] Leyuan Ma et al. “CRISPR-Cas9–mediated saturated mutagenesis screen predicts clinical drug resistance with improved accuracy”. In: *Proceedings of the National Academy of Sciences* 114.44 (2017), pp. 11751–11756.

[114] Lucas F. Ribeiro et al. “Protein Engineering Strategies to Expand CRISPR-Cas9 Applications”. In: *International Journal of Genomics* 2018 (Aug. 2018), p. 1652567.

[115] Miles Congreve, Christopher W. Murray, and Tom L. Blundell. “Keynote review: Structural biology and drug discovery”. In: *Drug Discovery Today* 10.13 (2005), pp. 895–907.

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

[117] John Jumper et al. "Applying and improving AlphaFold at CASP14". In: *Proteins* 89.12 (Dec. 2021), pp. 1711–1721.

[118] Charles Yanofsky, Virginia Horn, and Deanna Thorpe. "Protein Structure Relationships Revealed by Mutational Analysis". In: *Science* 146.3651 (1964), pp. 1593–1594.

[119] D. Altschuh et al. "Coordinated amino acid changes in homologous protein families*". In: *Protein Engineering, Design and Selection* 2.3 (Sept. 1988), pp. 193–199.

[120] Jesse Vig et al. *BERTology Meets Biology: Interpreting Attention in Protein Language Models*. 2020.

[121] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[122] Yen-Chun Chen et al. "Distilling the Knowledge of BERT for Text Generation". In: *CoRR* abs/1911.03829 (2019).

[123] Ting Chen et al. "Big Self-Supervised Models Are Strong Semi-Supervised Learners". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020.

[124] Roshan Rao et al. "Evaluating protein transfer learning with TAPE". In: *Adv. Neural Inf. Process. Syst.* 32 (Dec. 2019), pp. 9689–9701.

[125] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[126] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2017.

[127] Yufei Xiang et al. "Integrative proteomics identifies thousands of distinct, multi-epitope, and high-affinity nanobodies". In: *Cell Systems* 12.3 (2021), 220–234.e9.

[128] Bengt Bjellqvist et al. “The focusing positions of polypeptides in immobilized pH gradients can be predicted from their amino acid sequences”. In: *ELECTROPHORESIS* 14.1 (1993), pp. 1023–1031.

[129] Bengt Bjellqvist et al. “Reference points for comparisons of two-dimensional maps of proteins from different human cell types defined in a pH scale where isoelectric points correlate with polypeptide compositions”. In: *ELECTROPHORESIS* 15.1 (1994), pp. 529–539.

[130] Zhenwei Zhong et al. “Positive charge in the complementarity-determining regions of synthetic nanobody prevents aggregation”. In: *Biochemical and Biophysical Research Communications* 572 (2021), pp. 1–6.

[131] Mario S Valdés-Tresanco et al. “Structural insights into the design of synthetic nanobody libraries”. In: *Molecules* 27.7 (Mar. 2022), p. 2198.

[132] Mauno Vihinen, Esa Torkkila, and Pentti Riikonen. “Accuracy of protein flexibility predictions”. In: *Proteins: Structure, Function, and Bioinformatics* 19.2 (1994), pp. 141–149.

[133] Peter S. Brzovic et al. “Structure of a BRCA1–BARD1 heterodimeric RING–RING complex”. In: *Nature Structural Biology* 8.10 (Oct. 2001), pp. 833–837.

[134] Jun Lan et al. “Structure of the SARS-CoV-2 spike receptor-binding domain bound to the ACE2 receptor”. In: *Nature* 581.7807 (May 2020), pp. 215–220.

[135] Trong Nguyen-Duc et al. “Nanobody(R)-based chromatin immunoprecipitation/microarray analysis for genome-wide identification of transcription factor DNA binding sites”. In: *Nucleic Acids Res.* 41.5 (Mar. 2013), e59.

[136] Tomer Cohen, Matan Halfon, and Dina Schneidman-Duhovny. “NanoNet: Rapid and accurate end-to-end nanobody modeling by deep learning”. In: *Frontiers in Immunology* 13 (2022).

[137] Alex Klarenbeek et al. “Camelid Ig V genes reveal significant human homology not seen in therapeutic target genes, providing for a powerful therapeutic antibody platform”. In: *MAbs* 7.4 (2015), pp. 693–706.

[138] Joshua Meier et al. “Language models enable zero-shot prediction of the effects of mutations on protein function”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 29287–29303.

[139] Zeming Lin et al. “Evolutionary-scale prediction of atomic level protein structure with a language model”. In: *bioRxiv* (2022).

[140] Chloe Hsu et al. “Learning inverse folding from millions of predicted structures”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 8946–8970.

[141] Zhiqiang Gong, Ping Zhong, and Weidong Hu. “Diversity in Machine Learning”. In: *IEEE Access* 7 (2019), pp. 64323–64350.

[142] Jon Louis Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: 18.9 (Sept. 1975), pp. 509–517.

[143] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022.

[144] Pedro Reviriego and Elena Merino-Gómez. *Text to Image Generation: Leaving no Language Behind*. 2022.

[145] Abel Chandra et al. “Transformer-based deep learning for predicting protein properties in the life sciences”. In: *eLife* 12 (Jan. 2023), e82819.

[146] Ahmed Elnaggar et al. “ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (July 2021).

[147] Nadav Brandes et al. “ProteinBERT: a universal deep-learning model of protein sequence and function”. In: *Bioinformatics* 38.8 (Apr. 2022), pp. 2102–2110.

[148] Armin Behjati et al. "Protein sequence profile prediction using ProtAlbert transformer". In: *Computational Biology and Chemistry* 99 (2022), p. 107717.

[149] Hideki Yamaguchi and Yutaka Saito. "Evotuning protocols for Transformer-based variant effect prediction on multi-domain proteins". In: *Briefings in Bioinformatics* 22.6 (June 2021).

[150] Theodore Jiang, Li Fang, and Kai Wang. *Deciphering the Language of Nature: A transformer-based language model for deleterious mutations in proteins*. 2021.

[151] Ananthan Nambiar et al. "Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks". In: *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. BCB '20. New York, NY, USA: Association for Computing Machinery, 2020.

[152] Jack Lanchantin et al. "Transfer Learning for Predicting Virus-Host Protein Interactions for Novel Virus Sequences". In: BCB '21. New York, NY, USA: Association for Computing Machinery, 2021.

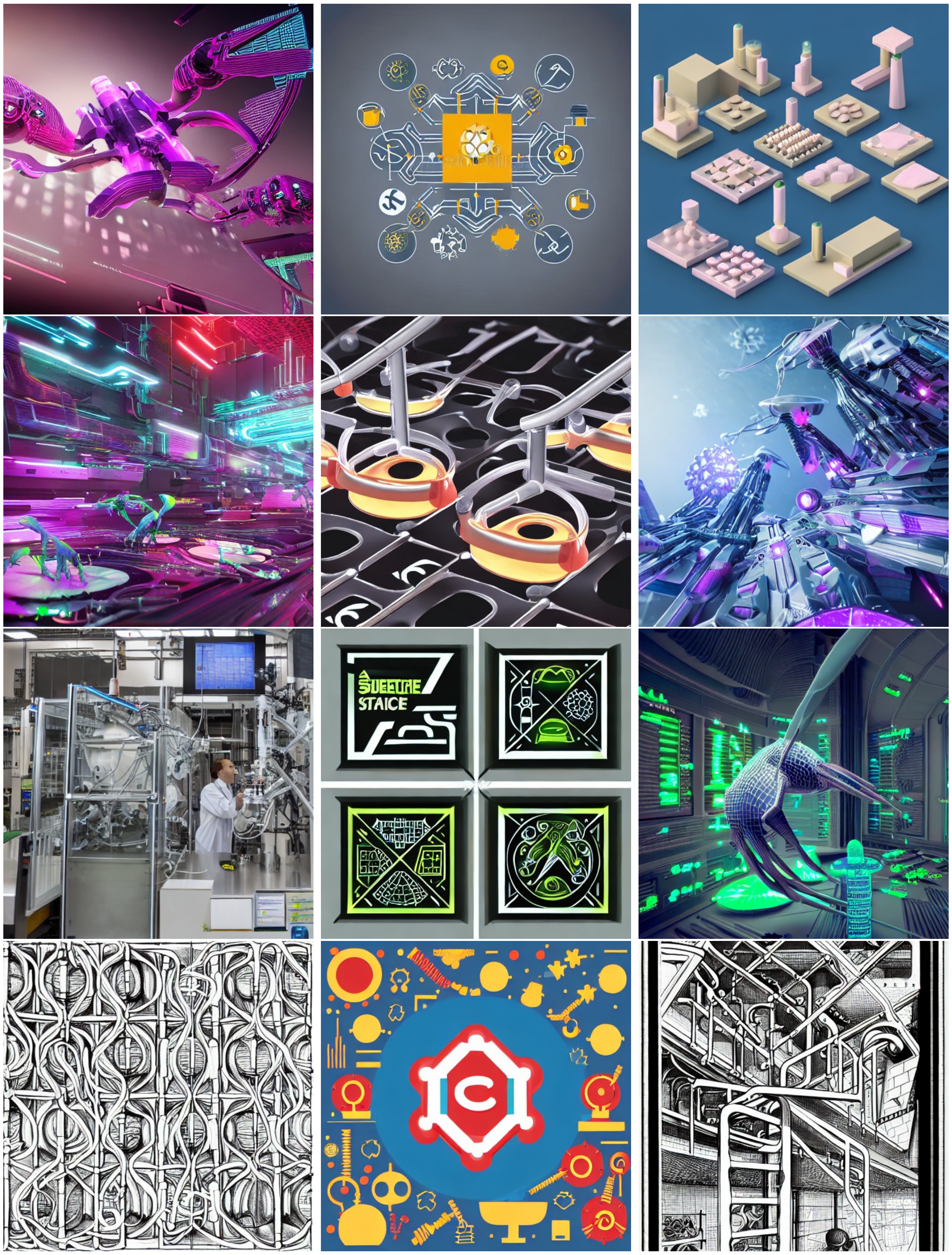
[153] Tian Cai et al. "MSA-Regularized Protein Sequence Transformer toward Predicting Genome-Wide Chemical-Protein Interactions: Application to GPCRome Deorphanization". In: *Journal of Chemical Information and Modeling* 61.4 (Apr. 2021), pp. 1570–1582.

[154] Junjie Wang et al. "ELECTRA-DTA: a new compound-protein binding affinity prediction model based on the contextualized sequence encoding". In: *Journal of Cheminformatics* 14.1 (Mar. 2022).

[155] Egbert Castro et al. "Transformer-based protein generation with regularized latent space optimization". In: *Nature Machine Intelligence* 4.10 (Oct. 2022), pp. 840–851.

[156] Christof Angermueller et al. "Model-based reinforcement learning for biological sequence design". In: *International Conference on Learning Representations*. 2020.

My Dissertation According to a Deep AI Image Generator



Examples generated from DeepAI's AI image generator [144] when given the prompt "Sequential Strategies for Automated Science and Protein Engineering."