Deep Generative Models for Biology: Represent, Predict, Design

Pascal Notin

Department of Computer Science

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

—

St Cross College Trinity 2023

Acknowledgements

I would like to first express my deepest gratitude to my supervisor, Yarin Gal, who has been an exceptional mentor and thought partner throughout my DPhil. His humility and brightness have been a constant source of inspiration, and I am forever grateful for his trust when he accepted me into the Oxford Applied and Theoretical Machine Learning group. I look forward to many more years of collaborating together.

I am incredibly fortunate to have been part of OATML, and I extend my heartfelt thanks to my wonderful colleagues there, especially those I have had the opportunity to collaborate with: Andrew Jesson, Clare Lyle, Ruben Weitzman, Lood van Niekerk, Shabbir Khan, Lisa Schut, Neil Band, and Jannik Kossen. Special thanks to Aidan Gomez for a fantastic research collaboration on many projects and from whom I have learned a great deal. I wish him all the best as he sets out to revolutionize the way we use and analyze language at Cohere. Cheers to Joost van Amersfoort, Angelos Filos and Panos Tigas - for the laughs, camaraderie, and inspiration - to Seb Farquhar, for showing me that a path in scientific research is possible after years in management consulting, and to Tim Rudner, for his great taste in literature. Thank you to Milad Alizadeh, Joost van Amersfoort, Lewis Smith, Andreas Kirsch, Lorenz Kuhn, and Muhammed Razzak for setting up and maintaining the OAT cloud, which has been an invaluable resource for several projects in this thesis. I would like to also acknowledge the teams behind the ARC and JADE compute clusters at Oxford for their support, which has made much of my research possible.

I am indebted to my collaborators at the Marks Lab for teaching me almost everything I know about Computational Biology. Special appreciation goes to Jonny and Mafalda for patiently going through the basics with me when we started working on EVE – I am thankful for our numerous collaborations and wish them the best as they establish their own lab in Barcelona. Additionally, I am grateful to Nikki Thadani, Sarah Gurev, Noor Youssef for teaching me most things I know about viruses; Sam Berry for everything I know about GPCRs; Aaron Kollasch and Dan Ritter for a wonderful collaboration on indels; and Nathan Rollins and Chris Sander for their help in navigating the complex, inspiring and fast-moving field of protein design. Last but not least, I want to express my deepest thanks to Debora Marks, without whom my DPhil journey would have been markedly different. Her

mentorship and guidance throughout our many joint projects have been invaluable, and I eagerly anticipate our continued collaboration in the years ahead.

I have had the immense pleasure to collaborate with Miguel Lobato, who taught me the basics of molecular optimization and provided invaluable guidance in the early stages of my DPhil. I would like to acknowledge Mark Woolrich and Chetan Gohil for our collaboration on fascinating neuroscience projects.

I am deeply grateful to GSK for their support throughout my DPhil, and I extend my appreciation to Lindsay Edwards for settings things up and Kim Branson for his ongoing support thereafter. I am particularly thankful to Patrick Schwab, who has provided guidance and thought leadership on many collaborations – from conducting cutting-edge research projects at the intersection of Active Learning and Biology, to bootstrapping the Machine Learning for Drug Discovery workshop at ICLR, to running the GeneDisco and CausalBench challenges. I would also like to thank Stefan Bauer, Arash Mehrjou, Ashkan Soleymani and Mathieu Chevalley for their partnership during these various initiatives, and would like to acknowledge my numerous collaborators at the Machine Learning for Drug Discovery workshop at ICLR and the workshop on Computational Biology at ICML.

Lastly, I want to thank my family and friends for their encouragements throughout, and in particular my wife, Sara, for her unwavering support since the beginning of this adventure, her patience with me around the much dreaded 'conference deadlines', and for her strength and love as we have just welcomed our first child, Luca. This thesis is dedicated to them.

Abstract

Deep generative models have revolutionized the field of artificial intelligence, fundamentally changing how we generate novel objects that imitate or extrapolate from training data, and transforming how we access and consume various types of information such as texts [1, 2], images [3, 4], speech [5, 6], and computer programs [7, 8]. They have the potential to radically transform other scientific disciplines, ranging from mathematical problem solving [9, 10], to supporting fast and accurate simulations in high-energy physics [11, 12] or enabling rapid weather forecasting [13, 14]. In computational biology, generative models hold immense promise for improving our understanding of complex biological processes [15, 16], designing new drugs and therapies [17–19], and forecasting viral evolution during pandemics [20, 21], among many other applications. Biological objects pose however unique challenges due to their inherent complexity, encompassing massive spaces, multiple complementary data modalities, and a unique interplay between highly structured and relatively unstructured components.

In this thesis, we develop several deep generative modeling frameworks that are motivated by key questions in computational biology. Given the interdisciplinary nature of this endeavor, we first provide a comprehensive background in generative modeling, uncertainty quantification, sequential decision making, as well as important concepts in biology and chemistry to facilitate a thorough understanding of our work. We then deep dive into the core of our contributions, which are structured around three chapters. The first chapter introduces methods for learning representations of biological sequences, laying the foundation for subsequent analyses. The second chapter illustrates how these representations can be leveraged to predict complex properties of biomolecules, focusing on three specific applications: protein fitness prediction, the effects of genetic variations on human disease risk and viral immune escape. Finally, the third chapter is dedicated to methods for designing novel biomolecules, including drug target identification, de novo molecular optimization, and protein engineering.

This thesis also makes several methodological contributions to broader machine learning challenges, such as uncertainty quantification in high-dimensional spaces or efficient transformer architectures, which hold potential value in other application domains. We conclude by summarizing our key findings, highlighting shortcomings of current approaches, proposing potential avenues for future research, and discussing emerging trends within the field.

Contents

List of Figures

1 Introduction

Contents

1.1 The deep generative modeling revolution

Deep generative models have revolutionized the field of artificial intelligence by offering new ways to model and understand data [22]. The primary objective of generative models is to learn the distribution of the data they are trained on, subsequently allowing the generation of new data samples that statistically resemble the observed data [23]. Unlike discriminative models, which learn a mapping between input features and targets, generative models focus on understanding the process that produces the data. Reaching this understanding encourages the corresponding networks to learn rich yet compact representations of the underlying data, which are often highly valuable in practical applications. Moreover, as labels are not strictly required during the training process, these models can learn complex representations from large, unlabelled datasets.

Different classes of deep generative models have been developed over time, each with distinct strengths and suitable for different tasks. Variational Autoencoders [24] can be effective when a structured latent space is desirable, as they provide a clear mechanism linking latent variables to observed data. Generative Adversarial Networks [25] excel in producing high-quality data samples with a generator-discriminator setup in which the two networks compete against each other to generate new, synthetic instances of data that can pass as authentic. Autoregressive models such as Recurrent Neural Networks [26, 27] or transformer-based architectures [2, 28], have shown impressive results in modeling sequential data.

While deep generative models offer significant potential, they are not without limitations (§ 2.2). These include the computational cost of model training, especially in scenarios involving large-scale data or high-dimensional settings, issues of mode collapse where the model overspecializes on certain data characteristics, limited control and interpretability of generated data, and adherence to real-world constraints, which is crucial in fields like chemistry or biology.

Despite these hurdles, the potential of deep generative models continues to expand, as demonstrated by their impressive successes across a broad spectrum of application domains, from natural language processing [1, 2] to computer vision [3, 4] and speech recognition [5, 6]. With ongoing advancements, they promise to usher in new era for creativity and scientific research.

1.2 Deep generative models for Biology

The past two decades have witnessed tremendous advancements in sequencing technologies, facilitating the accumulation of vast amounts of unlabeled biological datasets across multiple modalities [29, 30]. The analysis and interpretation of these datasets, given their scale and the complexity of the biological entities involved, necessitate the use of computational methods. In this context, deep generative models emerge as powerful tools to effectively address these challenges.

One of the key areas where generative models can make a substantial impact is in modeling complex biochemical objects such as proteins. Proteins carry out a wide

range of functions in nature, facilitating chemical reactions, transporting molecules, signaling between cells, and providing structural support to cells and organisms [31]. This astonishing functional diversity is uniquely encoded in their amino acid sequence, which spans a massive space: the number of possible arrangements for protein sequences as short as 64 amino acids is greater than the number of atoms in the universe. If inductive biases are properly modeled, deep generative models could accelerate the process by predicting the properties of novel proteins, or even designing novel proteins that hold the key to address our most pressing problems in healthcare, sustainability and agriculture [32, 33]. However, the vastness of protein space, coupled with the intricate relationship between the structure of a protein and its properties, make this a particularly challenging task.

In the context of genetic variations and their link to human disease, generative models could provide valuable insights. Variations in human genomes, such as single-nucleotide polymorphism or larger structural variations, play a substantial role in disease susceptibility [34]. Understanding the implications of these genetic variations can potentially unravel the etiology of various diseases and support earlier treatment for patients with genetic predispositions. Here, generative models could be leveraged to simulate different genetic variations and predict their impact on disease risk.

Furthermore, the emergence of new viral strains also present a crucial area where generative models could contribute. By predicting which mutation are most likely to evade the immune response, they could play a significant role in pandemic surveillance efforts and assist in the development of effective vaccines [21, 35].

1.3 Outline and contributions

This thesis aims to explore several questions at the intersection of deep generative modeling and computational biology, harnessing the potential of generative models to understand, predict, and design biological objects.

Given the interdisciplinary nature of our research, the next chapter provides the necessary background to facilitate the understanding of domain-specific concepts used throughout the study (§ 2). It starts with a broad introduction to generative

models, with a focus on the main model classes used in the rest of the work, namely Variational Autoencoders [24, 36] and Transformers [1, 28, 37]. Given their relevance to several applications tackled in this manuscript, we also delve into uncertainty quantification and sequential decision-making, in particular Bayesian optimization and active learning. Lastly, this chapter provides a brief overview of key concepts in computational biology and chemistry, starting with the various biological objects considered in our work. These objects include proteins, which play a central role in this thesis and are thoroughly discussed across all chapters, DNA (e.g., missense mutation effects in § 4.3, gene target identification § 5.2) and small molecules (e.g., de novo molecular optimization in § 5.3). This background chapter concludes with a description of several bioinformatics (e.g., multiple sequence alignment) and experimental methods (e.g., deep mutational scanning, CRISPR) which are frequently referred to throughout the text.

The core of this thesis is organized into three chapters, each targeting a key aspect of applying generative models to biology.

Represent (§ 3). The first chapter delves into the task of learning representations of biological sequences, and introduces several model architecture capable of learning compact, meaningful representations of these sequences. For consistency, we focus on proteins throughout, but we note that similar model architectures can be employed for learning representations of other biological sequences (e.g., DNA, RNA). Historically, models for learning protein representations have been trained on sets of homologous sequences recovered with a multiple sequence alignment [38] (§ 3.2). However, doing so presents several limitations as certain protein families are difficult to align (e.g., disordered proteins) or have few known homologs. This has recently motivated the development of generative models trained across protein families [39, 40](§ 3.3). Despite their ability to derive valuable representations across different families, these early family-agnostic methods are unable to match the performance of their family-specific counterparts, without explicitly leveraging homology [41]. This chapter seeks to address the following scientific questions:

- **Q1.** How can we best learn general-purpose representations of protein sequences across families?
- **Q2.** Do the corresponding model architectures perform on par with family-specific methods?
- **Q3.** Do family-specific and family-agnostic models capture redundant information, or could they be used synergistically?

Predict (§ 4). Building on the foundations laid in the first chapter, the second chapter focuses on leveraging the learned representations to predict complex properties of the underlying biomolecules. Three applications are covered. First, we focus on the task of predicting the fitness of mutated proteins, both in the zero-shot and the supervised settings. A robust evaluation framework is needed to do that well given the high model performance variability across DMS assays, and data leakage risks when dealing with biological sequences § 4.2.5. Second, we focus on the task of predicting the effects of genetic variations on human disease risk. This is challenging in practice as labels are scarcely available and subject to various biases (§ 4.3.1), and we therefore sought to address this task in a fully unsupervised fashion. Lastly, we investigate the task of predicting which viral mutations are likely to lead to immune escape. In this setting, a fully unsupervised approach is highly desirable as well, as it would guide pandemic surveillance efforts in the early stages of a pandemic, when limited experimental data or antibody information are available. This chapter thus aims to answer the following questions:

- **Q4.** How can we robustly assess the capability of various models to predict protein fitness, in the zero-shot and supervised settings?
- **Q5.** What is the performance lift provided by supervised methods over their zero-shot counterparts when predicting protein fitness?
- **Q6.** Can we predict the effects of missense mutations on human disease risk in a fully unsupervised manner?
- **Q7.** Can we predict the propensity of viral mutations to induce immune escape, early in a pandemic?

Design (§ 5). The third chapter explores strategies to support the design of novel biomolecules, primarily focusing on drug discovery. As the pharmaceutical industry grapples with high failure rates and escalating costs, integrating machine learning into the drug discovery process promises efficiency, cost reduction, and accelerated development timelines. This chapter examines three critical steps of drug development. Initially, we turn our attention to gene target identification for the creation of new therapeutics. This endeavor poses several challenges, such as the large design space necessitating data-efficient methods, and the need to diversify candidate selection to mitigate failure risk further down the drug development pipeline § 5.2. Subsequently, our focus shifts to de novo molecular optimization of small molecules, another crucial step in the drug discovery process. Recent works [17, 42] have proposed to transform the original discrete optimization task over molecular representations into a continuous one via the use of variational autoencoders. However, this approach has its own limitations, especially when the optimization leads to regions of the latent space far from the training data, causing difficulties in decoding sensible molecules § 5.3. Finally, we concentrate on a protein engineering task where the goal is to iteratively modify a sequence of interest to improve certain of its properties § 5.4. Throughout the chapter, we set out to answer the following main questions:

- **Q8.** In the process of identifying gene targets for new therapeutics, can we guide the experiments towards selecting interventions that are both diverse and optimize the phenotype of interest?

- **Q9.** During the de novo optimization of small molecules with variational autoencoders, how can we ensure the optimization in latent space eventually yields high-quality decodings?

- **Q10.** When optimizing protein properties iteratively, how can we best leverage the scarce labels we progressively collect to enhance the overall design performance?

The material discussed across these three chapters is based on several papers referenced in Table 1.1. Regarding the interdependencies between chapters, the various applications discussed in § 4 directly leverage models previously introduced in § 3. Additionally, the iterative design approach discussed in § 5.4 leverages protein language models discussed § 3, the DMS assays from ProteinGym introduced in § 4.2.2 and the ProteinNPT framework discussed in § 4.2.4.

While our research is motivated by biological and chemical applications, the thesis also contributes to broader machine learning challenges. These methodological contributions include approaches for uncertainty quantification in high-dimensional spaces, novel algorithms for sequential decision making and the development of more efficient transformer architectures.

The thesis concludes by summarizing our answers to the aforementioned scientific questions, highlighting current limitations of the discussed approaches, proposing potential avenues for future research, and discussing emerging trends within the field.

Declaration of Authorship. I hereby declare that this thesis has been composed by myself, and is based on my own work unless stated otherwise. No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification.

Table 1.1: List of papers covered in this thesis

Title	Authors	Thesis section	Journal / Conference
Disease variant prediction with deep generative models of evolutionary data	J. Frazer*, P. Notin*, M. Dias*, A. Gomez, J. Min, K. Brock, Y. Gal, D. Marks	§ 3.2, § 4.2.3, § 4.3	Nature (2021)
Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval	P. Notin, M. Dias, J. Frazer, J. Marchena-Hurtado, A. Gomez, D. Marks, Y. Gal	§ 3.3, § 3.4.2, § 4.2.3	ICML (2022)
RITA: A study on scaling up generative protein sequence models	D. Hesslow, N. Zanichelli, P. Notin, I. Poli, D. Marks	§ 4.2.3, § 6	ICML, WCB (2022)
TranceptEVE: Combining family-specific and family-agnostic models of protein sequences for improved fitness prediction	P. Notin, L. Van Niekerk, A. Kollasch, D. Ritter, Y. Gal, D. Marks	§ 3.4.2, § 4.2.3	NeurIPS, LMRL workshop (2022)
ProteinNPT: Improving protein property prediction and design with non-parametric transformers	P. Notin*, R. Weitzman*, D. Marks, Y. Gal	§ 4.2.5, § 5.4	NeurIPS (2023)
Learning from pre-pandemic data to forecast viral antibody escape	N. Thadani*, S. Gurev*, P. Notin*, N. Youssef, N. Rollins, C. Sander, Y. Gal, D. Marks	§ 4.4	Nature (2023)
GeneDisco: A benchmark for experimental design in drug discovery	A. Mehrjou*, A. Soleymani*, A. Jesson, P. Notin, Y. Gal, S. Bauer, P. Schwab	§ 5.2	ICLR (2022)
DiscoBAX: Discovery of optimal intervention sets in genomic experiment design	C. Lyle*, A. Mehrjou*, P. Notin*, A. Jesson, S. Bauer, Y. Gal, P. Schwab	§ 5.2	ICML (2023)
Principled uncertainty estimation for high dimensional data	P. Notin, J.M. Hernández-Lobato, Y. Gal	§ 2.3, § 5.3	ICML, UDL Workshop (2020)
Improving black-box optimization in VAE latent space using decoder uncertainty	P. Notin, J.M. Hernández-Lobato, Y. Gal	§ 5.3	NeurIPS (2021)
Machine Learning for functional protein design	P.Notin*, N. Rollins*, Y. Gal, C. Sander, D. Marks	§ 5.4	Nature Biotechnology (2023)

Legend: * Equal contributions; WCB is the Workshop on Computational Biology at ICML; UDL is the Uncertainty and Robustness in Deep Learningw Workshop at ICML; LMRL is the Learning Meaningful Representations of Life workshop at NeurIPS.

2 Background

Contents

2.1 Introduction

Given the interdisciplinary nature of the work undertaken in this thesis, we first provide the necessary background information to facilitate the understanding of the various contributions of this thesis. The first subject we delve into is generative modeling (§ 2.2), emphasizing on model classes such as VAEs and transformers,

which are extensively used in this thesis. We then define fundamental concepts related to uncertainty quantification (§ 2.3), followed by an introduction to Bayesian optimization and active learning (§ 2.4). This chapter concludes with a description of crucial concepts in computational biology and chemistry (§ 2.5) which will be recurrently referred to throughout the text (e.g., Multiple Sequence Alignment, Deep Mutational Scanning assay).

2.2 Generative modeling

2.2.1 Overview of generative modeling

Generative models are a class of machine learning models that aim to learn the underlying probability distribution of a given dataset and generate new data samples from that distribution. If we denote the true underlying distribution of the data as $p(\boldsymbol{x})$ and the distribution estimated by our model as $p(\boldsymbol{x};\boldsymbol{\theta})$, the primary goal of these models is to learn the parameters $\boldsymbol{\theta}$ such that the model distribution closely approximates the true data distribution. This is typically achieved by minimizing a divergence measure between the two distributions. A commonly used measure is the Kullback-Leibler (KL) divergence, which leads to an optimization problem: $\boldsymbol{\theta}^* = \arg\min_\theta KL[p(\boldsymbol{x})||p(\boldsymbol{x};\boldsymbol{\theta})]$. However, other measures such as Jensen-Shannon divergence or Wasserstein distance can also be employed, depending on the specifics of the problem and the model. Here, $\boldsymbol{x}$ represents a data point, $\boldsymbol{\theta}$ represents the model's parameters, and $KL[p||q]$ stands for the KL divergence, a measure of the dissimilarity between the two probability distributions p and q.

Learning the correct parameters $\boldsymbol{\theta}$ presents several challenges in practice. The main difficulty lies in dealing with high-dimensional data, a problem often referred to as the *curse of dimensionality*. As the dimensionality of data increases, the volume of the space grows exponentially, which means that the available data becomes sparse. This sparsity is problematic for any method that requires statistical significance, and increases the risk of overfitting. To achieve reliable statistical results, the amount of data needed often grows exponentially with the dimensionality of the data. Consequently, organizing and retrieving data efficiently in high-dimensional

spaces can become extremely difficult. Another common issue is the problem of *mode collapse*. This issue arises when the model identifies certain "modes" (peaks or high-density regions) in the target data distribution that it can replicate well, and focuses on those to the exclusion of others. The term "collapse" refers to the model's diversity of output collapsing down to these few modes, resulting in a failure to accurately capture the full complexity and variation of the true data distribution. Lastly, evaluating the performance of generative models is also a complex task because, besides domain-specific metrics (e.g., Fréchet Inception Distance, Inception score), there are no general-purpose metrics to evaluate the quality and diversity of the generated sample, and one often has to rely on a combination of quantitative metrics and qualitative (human-based) evaluation.

Generative models vary significantly in how they tackle these challenges, particularly in how they explicitly represent and learn the data distribution. They primarily differ along three dimensions:

1. **How they explicitly represent and learn the data distribution**: for example, Variational Autoencoders (VAEs) [24] seek to model the data distribution explicitly by assuming a specific form for it while Generative Adversarial Networks (GANs) [25], do not explicitly model the distribution but learn it implicitly during training;

2. **The approximations made when estimating the likelihood**: some models provide an exact likelihood estimate while others only offer a bound on the likelihood;

3. **The computational efficiency of sampling from the model**: some models allow efficient, direct sampling while others require iterative methods.

The choice of a specific generative model architecture often depends on the balance between these factors and the requirements of the task at hand. We now review the key architectures that are leveraged in subsequent chapters of this thesis.

2.2.2 Variational Autoencoders

Standard Variational Autoencoders

Latent variables and decoder network. Variational Autoencoders (VAEs) [24, 36] are a prominent class of generative models that make use of latent variables to model the data distribution explicitly. A VAE is based on a directed graphical model where the observed data $\boldsymbol{x}$ is generated by latent variables $\boldsymbol{z}$. The joint distribution over the observed and latent variables is expressed as $p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta})$, which can be factored into $p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})p(\boldsymbol{z})$ with the chain rule of probability. In the VAE framework, the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})$, known as the decoder, is modeled with a neural network generating data given the latent variables, while $p(\boldsymbol{z})$ is a prior distribution placed on the latent variables, typically a standard normal distribution.

Variational inference and encoder network. Performing inference over the latent variables, specifically computing the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$, can be challenging due to the high-dimensional integration involved in the computation of the marginal likelihood $p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$. To address this, VAEs leverage Variational Inference (VI) [43, 44], a method which approximates the true posterior by a simpler parametric distribution $q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})$, and learns its parameters $\boldsymbol{\phi}$ to minimize a divergence – typically the KL divergence – from the true posterior. Furthermore, VAEs use a specific type of VI known as amortized inference, where the parameters $\boldsymbol{\phi}$ of the approximate posterior are shared across different data points, effectively 'amortizing' the cost of inference over the entire dataset. In VAEs, the approximate posterior distribution $q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})$ is known as the encoder and is parameterized by a neural network as well.

Evidence Lower Bound. The aim of Variational Inference (VI) is to find the best approximating distribution $q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})$ that minimizes the Kullback-Leibler (KL) divergence to the true posterior $p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})$. The KL divergence is defined as:

$$KL[q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})||p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})] = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}[\log \frac{q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})}{p(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta})}] \tag{2.1}$$

However, as mentioned above, computing $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ is usually intractable in practice. Therefore, we instead maximize the Evidence Lower BOund (ELBO), which is derived as follows.

We start with the log-likelihood of the data:

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \log \int p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) d\boldsymbol{z} \tag{2.2}$$

We then introduce the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ and multiply and divide the integrand by it:

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \log \int q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})} d\boldsymbol{z} \tag{2.3}$$

Now, we take the expectation with respect to $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$, which gives:

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \log \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})} \left[\frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})} \right] \tag{2.4}$$

Applying Jensen's inequality ($\log \mathbb{E}[X] \geq \mathbb{E}[\log X]$), we then get the ELBO:

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}[\log p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})] - KL[q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})||p(\boldsymbol{z})] = \text{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}) \tag{2.5}$$

The ELBO consists of two terms: the expected log likelihood of the data under the model (first term), which encourages the decoder to reconstruct the data well, and the KL divergence between the approximate posterior and the prior (second term), which acts as a regularization term pushing the approximate posterior to be close to the prior. Maximizing the ELBO is equivalent to minimizing the KL divergence between the approximate posterior and the true posterior.

Reparametrization trick. VAEs utilize a technique known as the reparametrization trick to backpropagate gradients through the stochastic nodes in the computational graph. Normally, the derivative of a random variable with respect to its parameters is undefined, making it impossible to use gradient-based methods. The reparametrization trick overcomes this by transforming the random variable $\boldsymbol{z}$, drawn from $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$, into a deterministic variable $\boldsymbol{\epsilon}$, drawn from a simple

distribution like a standard normal, and a function of $\boldsymbol{\epsilon}$, $\boldsymbol{x}$, and $\boldsymbol{\phi}$. This allows gradients to be propagated through the deterministic nodes of the graph, enabling the use of gradient-based optimization methods like stochastic gradient descent. Additionally, VAEs often model the log variance instead of the variance itself to ensure that the variance stays positive during the optimization process.

Limitations. Despite their various advantages, VAEs have certain limitations. The most commonly observed drawback is that they often generate blurry samples compared to other generative models (e.g., GANs). This is usually attributed to the use of simple priors and approximating distributions, as well as the optimization of the ELBO, which may not align perfectly with perceptual quality. Another issue is the potential imbalance between the two terms in the ELBO. The reconstruction term can dominate the KL divergence term, leading to under-utilization of the latent space, where certain regions in the latent space do not correspond to any realistic samples. There is also a fundamental trade-off between the reconstruction and regularization terms in the ELBO, making it challenging to generate samples that are both diverse and high-quality. Additionally, VAEs often assume that the latent variables are independent, which might be too strong an assumption for many real-world datasets.

Bayesian Variational Autoencoders

Definition. In Bayesian VAEs [24], priors are placed not only on the latent variables but also on the parameters of the decoder, introducing an additional layer of Bayesian inference. The decoder parameters $\boldsymbol{\theta}$ are now drawn from a distribution $p(\boldsymbol{\theta})$ and the objective becomes maximizing the ELBO over both $\boldsymbol{\theta}$ and $\boldsymbol{z}$:

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}[\log p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})] - KL[q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})||p(\boldsymbol{\theta}, \boldsymbol{z})] \tag{2.6}$$

Often, fully-factorized Gaussian distributions are used to model the prior and approximate posterior distributions of the decoder parameters:

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \boldsymbol{I}) \tag{2.7}$$

$$q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) = q(\boldsymbol{\theta}|\boldsymbol{x}; \boldsymbol{\phi_\theta})q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}_z) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu_\theta}, \boldsymbol{\sigma_\theta^2 I})\mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2 \boldsymbol{I}) \tag{2.8}$$

The ELBO for a Bayesian VAE then becomes:

$$\begin{aligned} \text{ELBO}(\boldsymbol{\phi}; \boldsymbol{x}) &= \mathbb{E}_{q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})}[\log p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})] - KL[q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})||p(\boldsymbol{\theta}, \boldsymbol{z})] \\ &= \mathbb{E}_{q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi})}[\log p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})] - KL[q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}_z)||p(\boldsymbol{z})] - KL[q(\boldsymbol{\theta}|\boldsymbol{x}; \boldsymbol{\phi_\theta})||p(\boldsymbol{\theta})] \end{aligned} \tag{2.9}$$

The ELBO is composed of three terms: the expected log-likelihood of the data under the model, which encourages the model to reconstruct the data well; the KL divergence between the approximate posterior and the prior over the latent variables, pushing the latent space to be regular and well-structured; and the KL divergence between the approximate posterior and the prior over the decoder parameters, which acts as a regularization term ensuring the learned parameters do not deviate too much from the prior.

Advantages. Bayesian VAEs present several advantages. First, by treating the model parameters as random variables, Bayesian VAEs can capture the uncertainty associated with these parameters. This could be beneficial in many applications where knowing the uncertainty of the model can lead to better decision making or improved robustness. Second, the introduction of priors over the model parameters can act as a form of regularization, helping to prevent overfitting and leading to improved generalization on unseen data. The regularization effect of the Bayesian approach can be especially beneficial when dealing with small datasets.

Limitations. Bayesian VAEs require inference over both the latent variables and the model parameters, leading to increased computational complexity compared to standard VAEs. Furthermore, they often assume a simple form for the posterior over the decoder parameters (e.g., Gaussian), which might not be flexible enough to accurately capture the true posterior. Lastly, as with any Bayesian model, the choice of priors can significantly influence the results. Selecting appropriate priors is often non-trivial and requires careful consideration.

2.2.3 Transformers

Attention mechanism

Transformers [1] are a class of models initially designed for processing sequential data. At the heart of the Transformer architecture is the self-attention mechanism, which allows each symbol in a sequence to take into account all other symbols within the same sequence during its processing. Given a sequence of inputs $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, each symbol in the sequence is associated with a query Q, a key K, and a value V. These are vectors obtained by applying learned linear transformations to the input embeddings. The self-attention mechanism calculates the similarity of each query with all keys, applies a softmax to obtain attention scores, and uses these to take a weighted sum of the values. This can be formalized with the following formula mapping queries Q, keys K and values V ("scaled dot-product attention"):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.10}$$

where d_k is the dimension of the keys. This operation essentially computes a weighted sum of values, with weights determined by the query-key similarity $\frac{QK^T}{\sqrt{d_k}}$. The scaling by $\sqrt{d_k}$ is a heuristic to prevent the dot products from growing large in magnitude. Otherwise, the softmax function used to calculate the attention weights may become very 'sharp', effectively behaving like an argmax function and potentially leading to vanishing gradients.

To capture different types of relationships from the same input, transformers often use *multi-head* self-attention. In this setup, the self-attention mechanism is applied multiple times in parallel, each with different learned linear transformations of the input. Each of these parallel applications is called a 'head'. The outputs of each head are then concatenated and linearly transformed to produce the final output. Mathematically, for each head i, the self-attention operation is:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.11}$$

where W_i^Q, W_i^K, and W_i^V are the learned linear transformations for the i-th head. The final output of the MHSA mechanism is then:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O \tag{2.12}$$

where W_O is another learned linear transformation, and Concat denotes the concatenation operation.

There are various self-supervised learning strategies for training Transformer models on unlabeled datasets, such as autoregressive language modeling [2, 28], masked language modeling [37], permutation language modeling [45], fill-in-the-middle modeling [46], sentence order prediction [47] or replaced token detection [48]. These strategies define a certain "pretext task" that provides a learning signal from the unlabeled dataset and, in turn, entice the corresponding networks to learn a rich representation of the underlying data. The particular choice of self-supervised learning strategy is often guided by the requirements of the eventual task, as different strategies can lead to distinct types of representations in the Transformer model. Given their importance in modeling biological sequences and their use throughout this thesis, we provide a detailed description of autoregressive language modeling and masked language modeling in the next two subsections.

Autoregressive language modeling

In autoregressive language modeling, also known as Causal Language Modeling (CLM), the goal is to model the joint probability of a sequence of tokens $(x_1, x_2, ..., x_T)$ by factorizing it as a product of conditional probabilities:

$$p(x_1, x_2, ..., x_T) = \prod_{t=1}^{T} p(x_t|x_{<t}) \tag{2.13}$$

where $x_{<t}$ represents all the preceding tokens in the sequence. Each token is predicted one at a time, with each prediction only depending on the preceding ones in the sequence.

Models such as GPT (Generative Pretrained Transformer) [2, 28, 49, 50] leverage this objective. GPT consists of a stack of so-called transformer *decoder* layers (the

terminology originates from the seminal Transformer paper [1]) that use *masked* self-attention, where the attention mask ensures that the prediction for position i can only depend on the known outputs at positions less than i. This is done by setting the upper triangle of the attention matrix to $-\infty$ before the softmax step in the attention calculation. Each transformer layer in GPT consists of two main sub-layers: a masked self-attention mechanism and a position-wise feed-forward network. The same feed-forward network is applied independently to each position. It consists of two linear transformations with a ReLU activation in between:

$$\mathrm{FFN}(x) = W_2 \cdot \mathrm{ReLU}(W_1 \cdot x + b_1) + b_2 \tag{2.14}$$

where W_1, W_2 are weight matrices, b_1, b_2 are bias vectors, and x is the input to the feed-forward layer.

To stabilize the learning process and improve the representation power of the model, normalization and residual connections are incorporated throughout the network. Specifically, after each sub-layer (self-attention or feed-forward operation), the input to that sub-layer is added to its output via a residual connection, and then layer normalization is applied. This is referred to as post-layer normalization. Alternative pre-layer normalization architectures, which apply layer normalization before the self-attention and feed-forward sub-layers, have also been proposed and demonstrated to improve training stability [51].

Autoregressive transformers have demonstrated strong performance on a variety of language tasks due to their ability to capture long-range dependencies in text. However, these models come with certain limitations. They inherently operate in a sequential manner during inference, generating tokens one at a time, which can be time-consuming for longer sequences. Furthermore, given their left-to-right generative pattern, they can only leverage preceding context, neglecting any future context. This model design can lead to occasional inconsistencies in the generated sequence and prevents the embedding for a token at a given position to incorporate information about subsequent tokens in the sequence.

Masked language modeling

Masked Language Modeling (MLM) models, such as BERT [37], mask out a random subset of the tokens in the input sequence and try to predict these masked tokens from the unmasked ones. The masking procedure usually involves replacing a certain percentage of the tokens (typically 15%) in the input with a special [MASK] token, while the rest are left unaltered.

The MLM objective is formalized as follows. Given an input sequence $\boldsymbol{x} = (x_1, x_2, ..., x_N)$, a subset of the input tokens $\boldsymbol{m} = \{m_1, m_2, ..., m_M\}$ are masked out and the model is tasked to maximize the conditional probability of the masked tokens given the observed ones:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\boldsymbol{m}} \left[\sum_{i\in\boldsymbol{m}} \log p(x_i|\boldsymbol{x}_{\backslash\boldsymbol{m}}; \boldsymbol{\theta}) \right] \tag{2.15}$$

where $\boldsymbol{x}_{\backslash\boldsymbol{m}}$ denotes the sequence $\boldsymbol{x}$ with the positions in $\boldsymbol{m}$ masked out, and $\boldsymbol{\theta}$ represents the model parameters.

MLM models typically use transformer *encoders* (using the terminology from the original Transformer paper [1]), where each layer is composed of self-attention and position-wise feed-forward networks. Unlike autoregressive models, the self-attention mechanism in MLM models is not masked, allowing each token to attend to all tokens in the sequence. As a result, the embedding for a token at a given position incorporate information both from the left and right contexts in the sequence. MLM models thus learn rich representations that can be leveraged in downstream tasks. However, as they do not explicitly model the joint distribution of the full input, they are typically less suitable for full sequence generation tasks.

Position encoding

Unlike Recurrent Neural Networks, Transformers are designed to handle inputs in parallel for efficiency, and thus lack the inherent capability to consider the order or position of tokens in a sequence. Hence, to allow these models to consider the sequential nature of the data, an additional positional encoding is added to the input embeddings. These encodings are vectors that represent the position of the

word in the sequence and are created in such a way that the model can learn to use these to understand the relative positions of the words in the sequence.

One common approach is to use sinusoidal position encodings [1], which are constant and non-learned vectors that are added to token embeddings on input to the first layer of the transformer. For each position, the model generates a d-dimensional positional encoding vector, where each dimension corresponds to a sinusoid with a different frequency:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{2.16}$$

The use of both sine and cosine functions allows each position to have a unique encoding and helps the model to learn to attend to relative positions. The $10000^{2i/d}$ term represents a "wavelength" that grows exponentially with the dimension. This results in a mix of fast and slow oscillations in the positional encodings. These varying "wavelengths" allow the model to capture patterns at different temporal resolutions.

Another prevalent technique involves learned position embeddings [28, 37]. In this approach, a position embedding matrix is learned during training, where each row corresponds to a unique position in the sequence, and the matrix is of size $(\text{max sequence length}, d)$.

Rather than adding sinusoidal embeddings at the base of the transformer, Rotary Position Embeddings (RoPE) [52] multiply the keys and queries of each attention layer by sinusoidal embeddings. This procedure deviates from the traditional sinusoidal or learned positional embedding approaches by injecting positional information at every layer of the model, as opposed to only the initial one. Moreover, the Rotary method deliberately avoids inserting positional information into the values of the self-attention sub-layer. This is significant because the output of a self-attention sub-layer is essentially a weighted sum of the input value vectors, linearly transformed. Hence, by eschewing the insertion of positional information into the values, the transformer-layer outputs do not contain any explicit positional

information. It is conjectured that this segregation of positional information may be beneficial for extrapolation tasks.

Finally, instead of adding position embeddings to the input representation, ALiBi [53] introduces positional biases to the attention scores computed before the softmax operation. These biases depend on the relative positions of the key and query. When a key and query are close to one another, the bias is small, and when they are far apart, the bias is large. Each attention head has a different bias, determined by a head-specific slope parameter. This can be formalized as follows. Given a sequence of tokens, the attention score a_i for the i-th token is computed as:

$$a_i = \text{softmax}(q_i K^T + m \cdot [-(i-1), \ldots, 1, 0]) \tag{2.17}$$

where $q_i \in \mathbb{R}^d$ is the query of the i-th token, $K \in \mathbb{R}^{i \times d}$ are the keys up to position i, d is the dimension of the attention head, and m is a head-specific slope parameter that is fixed before training. ALiBi has been shown to extrapolate better at inference time to sequences longer than the maximum context length.

Axial Transformer

The Axial Transformer architecture [54] presents an innovative approach to handling high-dimensional objects. Traditional Transformers handle input data as flat sequences, which can be suboptimal for data with inherent dimensional structure, such as images or volumetric data. Axial Transformers, on the other hand, maintain the multi-dimensional structure of the data and apply self-attention mechanisms along each axis separately. For example, for a 2D data like an image, it separately applies self-attention to the rows and columns. This reduces the complexity of the attention computation and makes it feasible to apply Transformer models to higher-dimensional data.

2.3 Uncertainty quantification

Adopting a Bayesian viewpoint, the overall uncertainty of a model in a given region of the input space can be broken down into two types of uncertainty [55]:

- **Epistemic uncertainty:** Uncertainty due to lack of knowledge about that particular region of the input space — the posterior predictive distribution is broad in that region due to lack of information that can be reduced by collecting more data;
- **Aleatoric uncertainty:** Uncertainty due to inherent stochasticity/noise in the observations in that region — collecting additional data would not further reduce that uncertainty.

We denote input points as x, outputs as y and the training data as $\mathcal{D}$. The total uncertainty $\mathcal{U}$ of a model at an input point x is typically measured by the predictive entropy, ie. the entropy of the predictive posterior distribution $P(y|x, \mathcal{D})$:

$$\mathcal{U}(x) = \mathcal{H}(P(y|x, \mathcal{D})) = \sum_y -P(y|x, \mathcal{D}) \log P(y|x, \mathcal{D}) dy. \tag{2.18}$$

Denoting $P(\theta|\mathcal{D})$ the posterior distribution over model parameters θ, we can further decompose the predictive entropy $\mathcal{U}$ as the sum of two terms:

$$\mathcal{U}(x) = \underbrace{(\mathcal{H}(P(y|x, \mathcal{D})) - \mathbb{E}_{P(\theta|\mathcal{D})}(\mathcal{H}(P(y|x, \theta)))}_{\text{Mutual Information } \mathcal{M}} + \underbrace{\mathbb{E}_{P(\theta|\mathcal{D})}(\mathcal{H}(P(y|x, \theta)))}_{\text{Expected entropy } \mathcal{E}} \tag{2.19}$$

The first term — the Mutual Information $\mathcal{M}$ between model parameters θ and the prediction y — is a measure of epistemic uncertainty, as it quantifies the magnitude of the change in model parameters that would result from observing y. If the model is uncertain about its prediction for y, the change in model coefficients from observing y should be high. Conversely, if the model is confident about its prediction for y, model parameters will not vary from observing y:

$$\mathcal{M}(x) = \mathcal{H}(P(y|x, \mathcal{D})) - \mathbb{E}_{P(\theta|\mathcal{D})}(\mathcal{H}(P(y|x, \theta))). \tag{2.20}$$

The second term — the Expected Entropy $\mathcal{E}$ — is a measure of the residual uncertainty, ie. the aleatoric uncertainty:

$$\mathcal{E}(x) = \mathbb{E}_{P(\theta|\mathcal{D})}(\mathcal{H}(P(y|x, \theta))). \tag{2.21}$$

In high dimensional settings, an exact estimation of these different quantities is not tractable, therefore, several approximations and heuristics have been introduced.

The softmax variance, i.e. the variance of predictions across model parameters, has been shown to approximate epistemic uncertainty well in certain settings [56–58].

In the context of sequential data, the inherent structure in the data generating process often introduces strong dependencies between the output dimensions, e.g., the tokens in a generated sentence. In cases where there exist weak correlations between tokens, quantifying the different types of uncertainties above can be made tractable by ignoring these dependencies [59], in which case the predictive entropy for a sequence $y = (y_1, y_2, ..., y_L)$ may be approximated as the sum of token-level predictive entropies over the L tokens:

$$\mathcal{U}(x) = \sum_{l=1}^{L} \mathbb{E}_{P(y|x,\mathcal{D})}[\log P(y_l|x, y_{k<l}, \mathcal{D})] \approx \sum_{l=1}^{L} \mathbb{E}_{P(y_l|x,y_{k<l},\mathcal{D})}[\log P(y_l|x, y_{k<l}, \mathcal{D})]. \tag{2.22}$$

Unlike the standard expectation definition which integrates over all y, here only y_l is integrated over, and we condition on $y_{k<l}$, which are obtained from a sample from $P(y|x, \mathcal{D})$. The process is repeated for several of these samples and an average is finally computed to reduce variance.

While the above has been shown to work well in certain experiments [59], valuable information is being discarded when we ignore dependencies across tokens. These dependencies are likely to be informative in applications such as the ones considered in §5.3.

2.4 Sequential decision making

2.4.1 Bayesian Optimization

Bayesian Optimization (BO) is a sequential strategy for the global optimization of black-box functions [60–62]. The term "black-box" reflects the fact that we can only observe the outputs of the function for any given input, without any knowledge of its analytical expression. In many real-world optimization problems, we often have access to such black-box functions. They could represent expensive experiments, simulations, or processes that generate an outcome based on certain

control parameters, and we aim to find the inputs that optimize that outcome. BO is particularly relevant when the evaluations of the function are costly, the function is not convex, and derivatives for the function are not available.

BO builds a probabilistic surrogate model of the objective function. Typically, models such as Gaussian Processes (GPs) [63] or Bayesian Neural Networks [64–66] are used due to their capacity to estimate model uncertainty. In the case of GPs, the surrogate model is specified as $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'}))$, where $m(\boldsymbol{x})$ is the mean function, $k(\boldsymbol{x}, \boldsymbol{x'})$ is the covariance function, and $\boldsymbol{x} \in \mathcal{X}$ is the input vector.

BO iteratively updates this model using data collected from previous evaluations and selects the next query point $\boldsymbol{x}_{\text{new}}$ based on an acquisition function $\alpha(\boldsymbol{x}; \mathcal{D}_n, \boldsymbol{\theta})$, where $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$ is the observed data, and $\boldsymbol{\theta}$ are the GP hyperparameters:

$$\boldsymbol{x}_{\text{new}} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}_n, \boldsymbol{\theta}) \tag{2.23}$$

Several choices of acquisition functions exist, including Expected Improvement (EI) [67], Upper Confidence Bound (UCB) [68], and Probability of Improvement (PI) [69]. These acquisition functions are designed to balance exploitation (searching in areas of high predicted performance) and exploration (searching in areas of high uncertainty).

An illustrative example of a Bayesian Optimization process using a Gaussian Process is summarized in Algorithm 1, recapitulating the different steps discussed above.

Algorithm 1 Bayesian Optimization

1: **Input:** Initial training data $\mathcal{D}_0$, Gaussian process prior with kernel k, acquisition function α
2: **for** $t \in 1, 2, \ldots, N$ **do**
3: Fit Gaussian process on $\mathcal{D}_{t-1}$
4: Find $\boldsymbol{x}_{\text{new}} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}_{t-1}, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are the GP hyperparameters
5: Evaluate objective function to get $y_{\text{new}} = f(\boldsymbol{x}_{\text{new}})$
6: Augment the data $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_{\text{new}}, y_{\text{new}})\}$
7: **end for**
8: **Output:** The input-output pair $(\boldsymbol{x}^*, f(\boldsymbol{x}^*))$, where $\boldsymbol{x}^* = \arg\max_{x \in \mathcal{D}_N} f(x)$.

Despite its effectiveness in managing costly black-box optimization, BO can be computationally intensive, especially in high-dimensional spaces. This is due

to the cost of retraining or updating the surrogate after each acquisition. Various improvements and approximations have been proposed to scale BO to higher dimensions [62, 70].

2.4.2 Active Learning

Active Learning (AL) is a learning paradigm where the algorithm strategically selects the data it wants to learn from, typically in the form of querying labels from an oracle (e.g., human annotation, experiment) [71]. This is particularly useful when unlabeled data is abundant but labels are scarce or expensive to obtain.

In the simplest pool-based scenario (see algorithm 1), the AL procedure starts with a small labeled dataset $\mathcal{D}_L$ and a large pool of unlabeled data $\mathcal{D}_U$. The learning model, trained on $\mathcal{D}_L$, selects samples from $\mathcal{D}_U$ to be labeled based on an acquisition function $\alpha(\boldsymbol{x}; \mathcal{D}_L, \boldsymbol{\theta})$, where $\boldsymbol{x}$ is the input vector and $\boldsymbol{\theta}$ are the model parameters. This instance is then queried to the oracle for its label. The new labeled instance is added to the labeled dataset, and removed from the unlabeled pool. This process continues for a predetermined number of iterations or until a stopping condition is met. The final output of the algorithm is the learned model.

Algorithm 2 Active Learning

1: **Input:** Initial labeled data $\mathcal{D}_L$, Unlabeled data $\mathcal{D}_U$, Acquisition function α
2: **for** $t = 1, 2, \ldots, N$ **do**
3: Train model on $\mathcal{D}_L$
4: Select $\boldsymbol{x}_{\text{new}} = \arg\max_{\boldsymbol{x} \in \mathcal{D}_U} \alpha(\boldsymbol{x}; \mathcal{D}_L, \boldsymbol{\theta})$
5: Obtain label y_{new} for $\boldsymbol{x}_{\text{new}}$
6: Update $\mathcal{D}_L \leftarrow \mathcal{D}_L \cup \{(\boldsymbol{x}_{\text{new}}, y_{\text{new}})\}$, $\mathcal{D}_U \leftarrow \mathcal{D}_U \backslash \{\boldsymbol{x}_{\text{new}}\}$
7: **end for**
8: **Output:** Trained model

Several strategies exist to define the acquisition function, including uncertainty sampling [72], query-by-committee [73], or information-theoretic approaches [74–76]. While traditional AL strategies select one instance at a time, it is often more practical to label a batch of instances at once. Several batch active learning methods have been proposed to address this issue [77, 78].

2.5 Computational biology and chemistry

2.5.1 From small molecules to macromolecules

Biochemical processes involve a wide spectrum of molecular entities. On one end of the spectrum, small molecules (e.g., water, glucose) are molecules with a relatively low molecular weight and size, typically less than 900 Daltons. They play various roles including serving as substrates, metabolites, or signaling messengers in metabolic pathways [79]. They also have a significant role in pharmaceutical science, as they can be synthesized or modified to form drugs for therapeutic purposes [80]. Small molecules can be represented as linear sequences, for example via the Simplified Molecular Input Line Entry System (SMILES) [81], enabling efficient storage and analysis in computational systems. They can also be represented as 2D or 3D graphs, providing a useful structural view for predicting properties and understanding molecular interactions [82, 83].

On the other end of the spectrum are macromolecules, such as DNA, RNA and proteins, with substantially larger size. For instance, the molecular weight of proteins ranges from a few thousand Daltons to several hundred thousand Daltons for large complexes. The central dogma of molecular biology [84] outlines the fundamental process of information flow in biological systems: from DNA to RNA to proteins. DNA, a double-stranded macromolecule, stores the genetic information of an organism. Genes, sections of the DNA, are transcribed into single-stranded RNA molecules, of which the coding RNAs are subsequently translated into proteins, the workhorses of the cell. However, not all RNAs encode proteins. There is a vast universe of non-coding RNAs (ncRNAs) that perform a variety of functions beyond protein coding, such as regulating gene expression, maintaining the genome's structural integrity, and guiding protein synthesis among others [85, 86].

2.5.2 Protein structure

Proteins are complex macromolecules that are critical for the structure and function of living organisms, performing a vast array of roles, including catalyzing metabolic

reactions, DNA replication, responding to stimuli, and transporting other molecules. They are composed of one or more polypeptide chains, which are sequences of amino acids linked by peptide bonds. The protein backbone, made up of repeating sequences of nitrogen, alpha carbon, carbonyl carbon, and oxygen atoms, forms the core structure of the protein, while the side chains, or R groups, unique to each type of amino acid, extend outward from this backbone and confer specific properties and functionalities to the protein. The way these chains fold and interact gives rise to four levels of protein structure [31]. The primary structure refers to the linear sequence of amino acids. The secondary structure refers to the local sub-structures (e.g., alpha-helices, beta-sheets), primarily characterized by patterns of hydrogen bonding between the peptide backbone atoms. The tertiary structure represents the three-dimensional shape of the protein, determined by various interactions including hydrophobic interactions, disulfide bonds, and ionic bonding. Lastly, the quaternary structure describes the assembly of multiple protein subunits into a functional complex. The advancement of sequencing technologies has lead to the creation of vast repertoires of protein sequences, such as UniProt [87], a comprehensive resource for protein sequence and annotation data, or the Protein Data Bank (PDB) [88], which additionally provides 3D structure data. These databases provide invaluable resources for understanding protein function and studying evolutionary relationships.

2.5.3 Multiple Sequence Alignments

Multiple Sequence Alignments (MSAs) are a crucial tool in bioinformatics, allowing for the comparison and analysis of multiple related sequences simultaneously, typically proteins or nucleic acids [89]. In an MSA, sequences are arranged in rows with the goal of aligning identical or similar characters (amino acids or nucleotides) in the same column. The quality of the alignment can be evaluated using a scoring system, which assigns values based on matches, mismatches, and gaps (i.e., empty spaces introduced to maximize alignment).

There are various approaches to generate MSAs. Progressive alignment methods, such as ClustalW [89], start with pairwise alignments of the most similar sequences

and then add less similar sequences or groups of sequences iteratively. Iterative methods, like MUSCLE [90] or PSI-BLAST [91], improve the initial alignment through repeated realignment and adjustment. Other strategies involve the use of hidden Markov models [92] or genetic algorithms [93].

However, MSAs are not without limitations. The accuracy of the alignment depends heavily on the choice of scoring system and gap penalties, which can be subjective and difficult to optimize. Moreover, the problem is computationally complex, especially with long sequences or large datasets, often necessitating the use of heuristic approaches that might not find the optimal alignment [94]. Lastly, certain protein families are challenging to align, for example disordered proteins or proteins with few known homologs.

2.5.4 Deep Mutational Scanning assays

Deep Mutational Scanning (DMS) is a powerful, high-throughput technique used to measure the effects of mutations on protein function. It provides insights into protein structure-function relationships, fitness landscapes, and the molecular mechanisms underlying diseases [95, 96]. The procedure involves two main steps. First, a library of mutants is created for the protein of interest, typically covering all possible single amino acid substitutions. This is often accomplished using error-prone PCR or site-directed mutagenesis [97]. The mutated genes are then transformed into a suitable host organism, such as yeast or bacteria, and the mutants are subjected to a selection or screening assay to measure their relative fitness [95]. High-throughput sequencing is used to measure the frequency of each mutant before and after selection, allowing the fitness effects of all mutations to be quantified simultaneously. For example, this might involve examining growth rates, protein expression levels, protein stability, or other phenotype relevant to the function of the protein.

While extremely useful in practice, DMS assays present several challenges. For instance, DMS studies are often conducted in a model organism (like yeast or bacteria) or in vitro. These systems might not perfectly reflect the native cellular environment of the protein in question, and the observed effects might not fully

translate to the organism of interest. Furthermore, the phenotype used for selection must be relevant to the function of the protein, but it might not capture all aspects of that function, or the phenotype might not be amenable to high-throughput selection. Lastly, given the massive size of the sequence space, experiments are often limited to exhaustive enumeration of shallow mutants (i.e., singles or doubles) or going deeper for a limited subset of sequence positions.

2.5.5 CRISPR experiments

The Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR) system, coupled with the CRISPR-associated (Cas) proteins, most notably Cas9, has revolutionized the field of genetic engineering due to its ability to precisely edit genomic sequences. Originating from a naturally occurring bacterial immune system against viruses, CRISPR-Cas9 allows for targeted DNA breaks at specific sites in the genome, dictated by a guide RNA molecule that matches the target DNA sequence [98]. The cellular machinery repairs these breaks, often incorporating or deleting genetic material in the process, resulting in gene editing [99]. CRISPR-Cas9 has broad applications across biology, from creating genetically modified organisms to studying gene function, developing disease models, and even potential applications in gene therapy. However, off-target effects, where cuts are made at unintended sites, can occur, though continuous advancements are being made to improve the specificity of the system. Moreover, delivery of the CRISPR-Cas9 system into cells and organisms, especially for therapeutic purposes, remains a challenge [100].

3
Represent

Contents

3.1 Introduction

Biochemical processes involve a wide range of molecular entities with singular characteristics – from small molecules with a relatively low molecular weight and simple structure (e.g., water, glucose, caffeine) to more complex macromolecules (e.g., DNA, RNA, proteins). The first challenge one is concerned with when developing machine learning models for biochemical applications is to learn a

compact representation that best captures the characteristics of the relevant entities depending on various factors such as their structure, dimensionality, symmetries and invariants.

In this chapter, we will use *proteins* as our main running example, as several of the models we developed were specifically designed with proteins in mind. However, similar approaches could be leveraged to model other biological entities (eg., DNA/RNA, small molecules) given their shared attributes (e.g., they all possess a 3D structure, yet can all be represented as linear sequences as well). For instance, some of the applications discussed in § 5.3 are based on generative models trained on SMILES representations (§ 2.5.1) of small molecules.

Proteins are the molecular machines of life and play critical roles in the vast majority of biological processes. As discussed in § 2.5.2, they exhibit a hierarchical structure with four levels of organization. The primary structure refers to the linear sequence of amino acids in a protein, which determines its unique identity and function. The secondary structure arises from local interactions between nearby amino acids, resulting in common motifs such as alpha helices and beta sheets. The tertiary structure results from the three-dimensional folding of the entire protein, driven by interactions between amino acid side chains and the surrounding environment. Finally, the quaternary structure emerges when multiple protein subunits come together to form a functional protein complex.

While protein structure can be explicitly determined through experimental techniques, such as X-ray crystallography and cryo-electron microscopy, obtaining protein structure data is challenging and time-consuming. As a result, the availability of experimentally-validated protein structure data is limited and many proteins still lack experimentally determined structures: there are orders of magnitude more protein sequences available than there are structures. For example, at the time of this writing, there are a bit less than 180k proteins with experimentally-derived structure in the Protein Data Bank (PDB) [101], while there are over 2 billion non-redundant protein sequences in each of the MGnify protein database [102] and the Big Fantastic Database (BFD) [16]. Given this massive discrepancy,

models that are devised to learn from sequential information only have much larger scope, and may be able to learn more general and robust features from a larger and more diverse pool of proteins [1].

For this reason, we focused on developing protein representations that are based on *sequence* only. We broadly categorize existing protein sequence models in three groups: family-specific models (§ 3.2), family-agnostic models (§ 3.3), and hybrid models (§ 3.4). At a high level, family-specific models are trained on evolutionarily-related sequences from a given protein family, while family-agnostic are typically trained across unaligned protein sequences across families. Hybrid models integrate both family-specific and family-agnostic characteristics to leverage the strengths of both approaches. This chapter is itself split in three subsequent sections, each one focusing on one of the three aforementioned groups. In each section, we first review existing methods and their objectives, then introduce the novel methods that we developed. The methods discussed in this chapter are key building blocks for the approaches discussed in the rest of the text.

3.2 Learning family-specific representations

3.2.1 Background and objectives

The protein sequences that we observe today across living organisms are the result of billions of evolutionary experiments that contributed to select out unfit variants and promote the ones with biomolecular properties that benefit the corresponding organisms. By training a generative model on a set of natural protein sequences, we learn a model of what constitutes valid protein sequences and thus implicitly learn the rules that characterize the corresponding biochemical constraints. Given the vastness and diversity of sequences across protein families, sequence-based representations were initially derived for each protein family separately, where

[1] It may be argued that Alphafold2 [16] *predicted* structures offer a path to addressing that gap – at least partially if one were to consider the subset of higher confidence predictions. This is an active area of research, with early works that either illustrate the benefits from simultaneously training on predicted and experimentally-derived structures [103] or combining representations from structure and sequence models together [104].

the sequences from the same family are typically identified and collected via a Multiple Sequence Alignment § 2.5.3). While initial family-specific models focused on extracting position-specific information from alignments [105], subsequent work sought to capture more complex patterns. [106] proposed to model interactions between pairs of distinct positions with energy based models. [107] later expanded on the concept with DeepSequence, in which Variational Autoencoders trained on protein-specific MSAs to learn a distribution of amino acid sequence which capture higher-order interactions. The method we introduce in the next subsection builds on DeepSequence, with the aim to learn superior protein sequence representations and increase its performance for mutation effects prediction (§ 4.3).

3.2.2 EVE – A family-specific Bayesian VAE

Learning distributions over protein spaces

As discussed in § 2.2.2, Variational Autoencoders (VAEs) [24, 36] have been shown to be effective at learning complex high-dimensional distributions across a wide range of tasks – from computer vision, to natural language processing [108], to molecules modeling [17], and many others. In this work, we train VAEs to infer a distribution over amino-acid sequences for each protein. More formally, for a given protein family p, we learn a distribution $p(\mathbf{s}|\theta_\mathbf{p})$ where $\mathbf{s}$ is a fixed-length amino-acid sequence and $\theta_\mathbf{p}$ are the model parameters for that protein family. Variational Autoencoders make the assumption that the data $\mathbf{s}$ are generated from a latent variable $\mathbf{z}$, and model the conditional distribution $p(\mathbf{s}|\mathbf{z}, \theta_\mathbf{p})$ with a neural network architecture (the "decoder", with parameters $\theta_\mathbf{p}$). We use amortized inference and model the approximate posterior distribution $q(\mathbf{z}|\mathbf{s}, \phi_\mathbf{p})$ with a neural network as well (the "encoder", with parameters $\phi_\mathbf{p}$). Similarly to what Riesselman et al. observed in DeepSequence [107], we obtain stronger downstream task performance [2] with a Bayesian VAE in which we learn a fully-factorized Gaussian distribution over the decoder weights $\theta_\mathbf{p}$ (§ 2.2.2). We interpret this observation by the fact that, in a VAE architecture, the encoder will extrapolate arbitrarily when dealing with

[2] We specifically mean 'fitness prediction' here. We discuss that task in detail in § 4.2

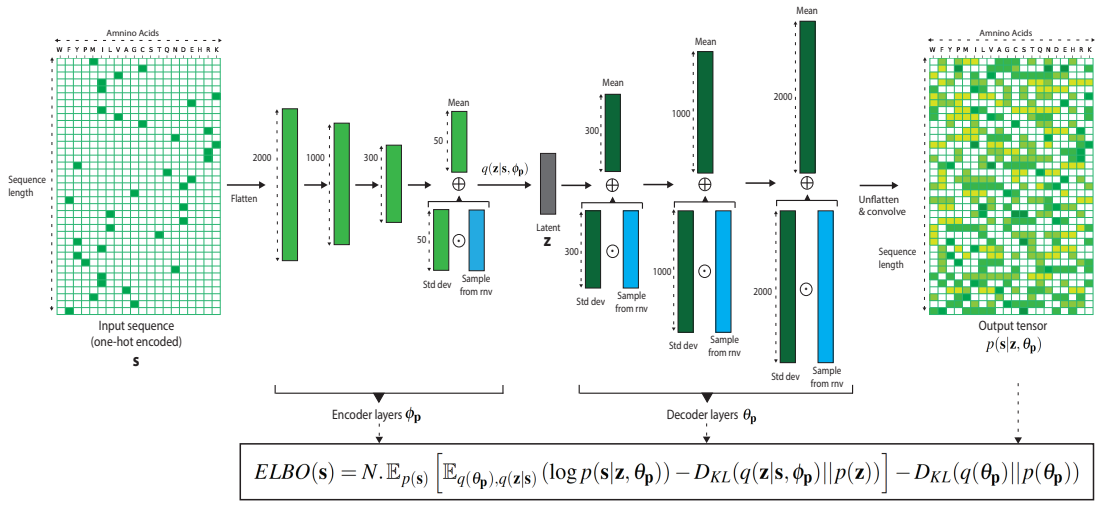

Figure 3.1: Bayesian VAE architecture details. The Bayesian VAE architecture in EVE is comprised of a symmetric 3-layer encoder & decoder architecture (with 2,000-1,000-300 and 300-1,000-2,000 units respectively) and a latent space of dimension 50. After performing a one-hot encoding of the input sequence across amino acids (zeros in white, ones in green), we flatten the input before performing the forward pass through the network. We use a single set of parameters for the encoder (ϕ_p) and learn a fully-factorized gaussian distribution over the weights of the decoder (θ_p): weight samples for the decoder are obtained by sampling a random normal variable (rnv), multiplying that sample by the standard deviation parameters, and subsequently adding the mean parameters. A one-dimensional convolution is applied on the un-flattened output of the decoder to capture potential correlations between amino-acid usage. Finally, a softmax activation turns the final output into probabilities over amino acids at each position of the sequence (low values in white, high values in dark green). The overall network is trained by maximizing the Evidence Lower Bound (ELBO), which forms a tractable lower bound to the log-marginal likelihood

out-of-distribution points. Thus, for mutants 'far' from the training data, latent positions obtained from the encoder will be less reliable. While a standard VAE will then decode that mutant to an arbitrary position, the output from a Bayesian VAE will average over decodings (the decoder being a Bayesian Neural Network), which will dampen the corresponding probability estimates over amino acid positions [109]. We operate several structural changes compared to the model architecture and training procedure of DeepSequence, which we summarize in Appendix A.1. Our final model architecture is represented in Fig. 3.1.

Training VAEs over Multiple Sequence Alignments

The distribution of protein sequences available in genomic databases is biased by human sampling (e.g., certain species of interest are more sequenced than others) and evolutionary sampling (e.g., phylogeny). Not correcting for these biases in the Multiple Sequence Alignments (MSAs) that we extract from genomic databases to train our models will lead to learning an improper probability distribution. Following the approach described in Ekeberg et al. [110], we correct for these two biases by re-weighting each protein sequence $\mathbf{s}_i$ from a given MSA according to the reciprocal of the number of sequences in the corresponding MSA within a given Hamming distance cutoff T.

$$\pi_{s_i} = \left(\sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbb{1}[Dist(\mathbf{s}_i, \mathbf{s}_j) < T] \right)^{-1} \tag{3.1}$$

where N is the number of sequences in the MSA, i indexes over proteins, and $\mathbf{s}_j$ are other protein sequences in the MSA. Similarly to Hopf et al. [38], we set $T = 0.2$ for all human proteins. During model training, we sample each mini-batch element by sampling sequences according to their weight π_s.

We train the VAE models by maximizing the Evidence Lower Bound (ELBO) which forms a tractable lower bound to the log-marginal likelihood (§ 2.2.2). Following the 'Full Variational Bayesian' approach described in [24], the ELBO is expressed as follows:

$$\begin{aligned} ELBO(\mathbf{s}) = N.\, \mathbb{E}_{p(\mathbf{s})} \Big[\mathbb{E}_{q(\theta_\mathbf{p}), q(\mathbf{z}|\mathbf{s})} \left(\log p(\mathbf{s}|\mathbf{z}, \theta_\mathbf{p}) \right) - D_{KL}(q(\mathbf{z}|\mathbf{s}, \phi_\mathbf{p}) || p(\mathbf{z})) \Big] \\ - D_{KL}(q(\theta_\mathbf{p}) || p(\theta_\mathbf{p})) \end{aligned} \tag{3.2}$$

where N is the size of the training data and $p(z)$ is the prior distribution over latent variable z (standard Gaussian), and D_{KL} is the Kullback–Leibler divergence (a measure of dissimilarity between two probability distributions). In our Bayesian VAE formulation, we learn a fully-factorized Gaussian distribution $q(\theta_\mathbf{p})$ over the decoder parameters with standard Gaussian prior $p(\theta_\mathbf{p})$. Similar to Riesselman

et al. [107] and in agreement with our sequence re-weighting scheme, we set $N = N_{eff} = \sum \pi_{s_i}$, the effective number of sequences in the MSA defined as the sum of the different sequence weights.

3.3 Learning family-agnostic representations

3.3.1 Background and objectives

While training models on protein-specific alignments has proved to be effective to address several downstream tasks – from mutation effects predictions [106, 107] to contacts prediction [111] and protein design [112], it nevertheless brings severe limitations. For instance, such models can not make predictions for sequences which are incompatible with the coordinate system of the MSA used in training (eg., insertions and deletions), thereby limiting scope. Additionally, a large fraction of the proteome corresponds to regions that can not be aligned such as so-called disordered regions – around half of all human proteins contain regions of at least 40 amino acids classified as disordered [113, 114]. Even when alignments are accessible, the protein function might be taxa specific, and the MSA algorithm may not retrieve a large enough set of homologous sequences for model training. Alignment-based models may be relatively sensitive to the characteristics of the MSAs they are trained on – including the choice of hyperparameters used to retrieve these MSAs. Lastly, there is a lack of information sharing across models that are independently trained on different data subsets. This has led to a stream of research investigating alternative modeling approaches that do not rely on aligned sequences. Shin et al. [112] and Weinstein et al. [115] developed models that could be trained on non-aligned sequences, although they still relied on MSA routines to recover protein-specific sets of homologous sequences to serve as training data. Alley et al. [39] and Heinzinger et al. [116] were the first to introduce models trained across protein families, relying on LSTM architectures [117]. Building on advances in the Natural Language Processing literature to train larger-scale language models, several research groups [40, 41, 118–120] leveraged transformer architectures to model protein sequences. However, models trained on non-aligned sequences still failed to

match the performance of alignment-based methods without further fine-tuning on evolutionarily-related sequences obtained with a MSA [41]. Furthermore, approaches based on masked language modeling objectives (e.g., ESM-1b [40], ESM-1v [41]) are trained to predict the probability of masked tokens conditioned on unmasked tokens (§ 2.2.3). As such, they do not allow to estimate the log-likelihood of the full protein sequences, leading to heuristics when predicting mutation effects. These heuristics ignore for instance the dependencies between mutated positions – which may be particularly suboptimal when predicting the effects of multiple mutants with strong epistatic effects. Several of these heuristics, including the masked-marginal introduced in Meier et al. [41], are also unable to score insertions and deletions (indels). We discuss how we address these limitations in the next subsection.

3.3.2 Tranception – A family-agnostic autoregressive transformer

Learning probabilities over full protein sequences with autoregressive transformers

To address the limitations discussed above, we introduce Tranception, an autoregressive transformer architecture which is pretrained on large quantities of non-aligned sequences. We specifically chose an autoregressive architecture as it learns a probability over full sequences (§ 2.2.3) which, we hypothesized, should yield higher performance when scoring multiple mutants. Since it is trained on non-aligned sequences, it should also perform better than family-specific methods on proteins with shallow alignments. Lastly, it also provides a natural way to score indels and to generate novel protein sequences – which makes it a very compelling approach for protein design applications. From a model architecture standpoint, Tranception combines ideas from the standard GPT architecture [28] and the Inception architecture [121] – which inspired its name. It was designed with two core principles in mind: 1) promoting specialization across attention heads 2) explicitly extracting patterns from contiguous subsequences.

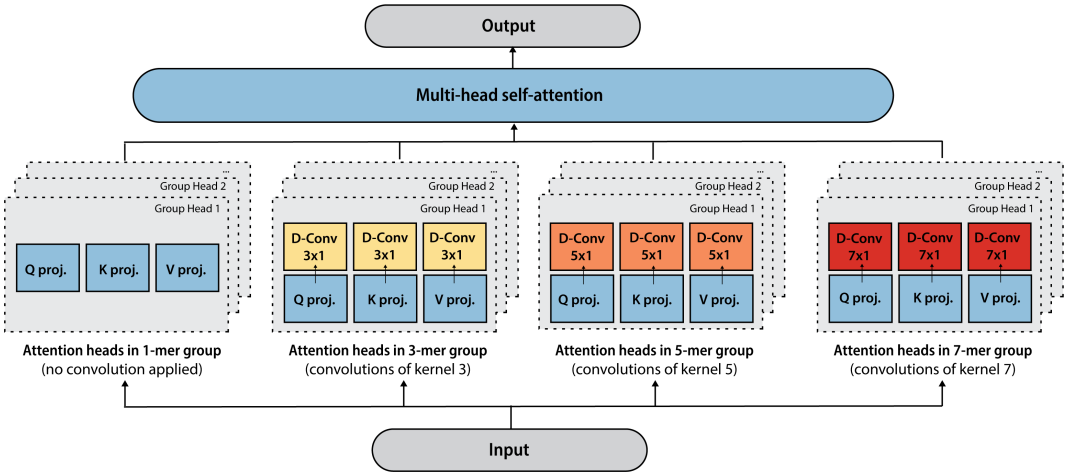

Figure 3.2: Tranception attention mechanism. Attention heads at each layer are split into 4 distinct groups. Except for the first group that does not mix information across tokens (i.e., '1-mer' group), separate spatial depthwise convolutions are applied after the Query, Key and Value projections of the other 3 groups with kernel size of 3, 5 and 7 respectively. This incentivizes attention heads within each group to mix information at various ranges across the sequence length. We preserve autoregressiveness of the architecture by applying the right amount of left padding for each convolution.

Tranception attention

The concept of 'k-mers' is well-established in biological sequence analysis: k-mers are contiguous subsequences of k elements (typically nucleotides or amino acids) which have proved to be critically useful abstractions in several applications such as de novo assembly, read correction, repeat detection, comparison of genomes, metagenomes [122]. The majority of protein language models to date (§ 3.3.1) have focused on extracting patterns (via sequence tokenization or attention mechanisms) at the amino acid level only. With Tranception, we investigate the benefits from explicitly attending over contiguous subsequences of amino acid tokens via a novel attention mechanism – Tranception attention (Fig. 3.2). As in Primer [123], we leverage squared ReLU activations and depthwise convolutions after the different multi-head attention projections. Yet, instead of using similar-sized kernels for each depthwise convolution, we split the attention heads at each layer in 4 groups and apply convolutions with different kernel sizes on each group, thereby combining information at different resolutions as in Inception [121]. This incentivizes each attention head to specialize to pattern extractions at different k-mer sizes and leads

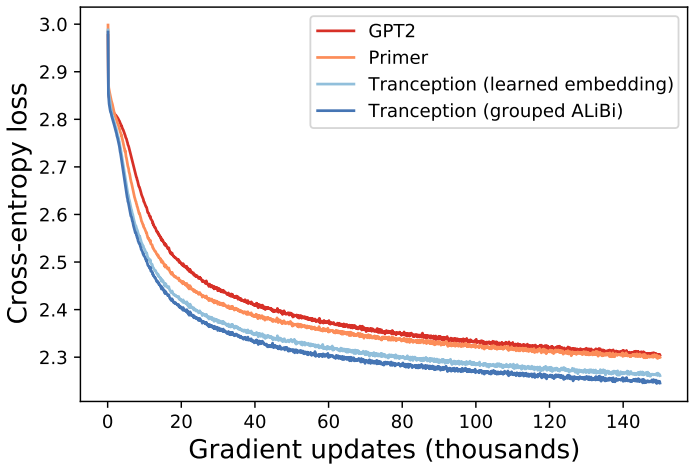

Figure 3.3: Training loss comparison across transformer architectures for protein modeling. We plot the training loss as a function of the number of gradient steps for GPT2 [28], Primer [123], Tranception with learned position embeddings and Tranception with grouped AliBi. All models have similar number of parameters and only differ by they attention mechanism, non-linear activations and position encodings. Tranception converges faster and to a lower loss compared with other architectures. This translates into higher downstream task performance (Appendix A.2.1).

to both more efficient training and downstream task performance compared with Primer and GPT2 [28] (Fig. 3.3 and Appendix A.2.1).

Grouped ALiBi position encoding

In order to further promote specialization across attention heads and enhance predictions for protein sequences that are longer than the context length, we replace the learned or sinusoidal position encodings typically used in autoregressive transformer architectures [1, 49], with a variant of ALiBi [53] called 'Grouped ALiBi'. Similar to ALiBi, we remove the position encodings added to input token embeddings, and bias the query-key attention scores with a term proportional to their distance (see § 2.2.3). We however apply the mechanism on each group of attention heads independently, in adequacy with the Tranception attention scheme, to enable each group to learn attention patterns at different distances. Compared to using learned position encodings, this helps reduce the number of parameters, converge faster during training and leads to better downstream task performance (Fig. 3.3 and Appendix A.2.1).

Data processing and augmentations

Our models are trained on UniRef [124], a large scale protein sequence database. We perform thorough ablations when developing Tranception (Appendix A.2.1). Similar to Meier et al. [41], we investigate the impact of training on protein sequences clustered at different levels of similarity. Unlike what was observed for masked-language model architectures, we find that keeping as much of the granularity available in the dataset is beneficial to downstream task performance. We therefore train our final model (700M parameters) on UniRef100 which, after preprocessing (Appendix A.2.2), leads to a training dataset of ∼250 million protein sequences. Our vocabulary is comprised of the standard 20 amino acids [125]. We find that averaging the predictions obtained by scoring each sequence and its reverse at inference time leads to higher downstream performance (§ 4.2.3 and Appendix A.2.4), and therefore apply sequence mirroring at random during training to teach our model to score sequences from both directions. Our model has a maximum context size of 1024 tokens, which is wider than the length of 98% of protein sequences in UniRef100 (Table A.3). At train time, if a protein is longer than the maximum context size of the model (after accounting for the special start and end of sequence tokens), we extract a randomly-selected contiguous slice of width equal to that maximum context size. Indeterminate amino acids are imputed at random during training and inference.

Protein sequence likelihood

The likelihood of a protein sequence under the distribution learned by the generative model being considered plays a central role in the majority of the applications we will be discussing in § 4 and § 5: it is a measure of how plausible a given sequence is under the learned model, implicitly recapitulating the constraints between residues that have been observed on the training data.

We represent each protein x as a sequence of amino acids $(x_1, x_2, ..., x_l)$. Tranception is trained in a self-supervised fashion to predict the next token x_i in the sequence based on the context of the prior $i-1$ tokens, such that the probability

of the full sequence factorizes as:

$$P(x) = \prod_{i=1}^{l} P(x_i|x_1, ..., x_{i-1}) = \prod_{i=1}^{l} P(x_i|x_{<i}) \tag{3.3}$$

In practice, we often work with log-likelihoods for numerical stability. At inference time and building on our data augmentations, we take the arithmetic average of the log-likelihood ratios obtained by scoring each sequence $x_{\rightarrow}$ and its mirror image $x_{\leftarrow}$:

$$\log P(x) = \frac{1}{2}(\log P(x_{\rightarrow}) + \log P(x_{\leftarrow})) \tag{3.4}$$

Additional details are provided in Appendix A.2.4.

As we will discuss in § 4.2, the Tranception model based on the above achieves fitness prediction performance higher than all other protein language models we compared against: Unirep [39], ESM1v [41], Progen2 [126], RITA [127] and ProtGPT2 [128]. Since it learns a generative models of full protein sequences, it performs particularly well on multiple mutants, with the gap widening with mutation depth. This has important ramifications in several applications, in particular in machine-learning guided protein design (see § 5.4) for which accurate extrapolation of sequence likelihood far away in sequence space is critical to uncover novel desirable candidate proteins. However, the base Tranception autoregressive model does not fully close the gap in terms of overall performance with state-of-the-art family-specific models like EVE (§ 3.2), which lead us to introduce the methods discussed in the next section.

3.4 Learning hybrid representations

3.4.1 Background and objectives

A standard approach to combine the strengths of family-agnostic and family-specific models is to leverage model *fine-tuning*. Alley et al. [39] demonstrated that fine-tuning Unirep, a family-agnostic model, on evolutionarily-related proteins obtained through a Multiple Sequence Alignment (MSA), a process they called 'evo-tuning', resulted in improved downstream task performance, such as protein

property prediction. Meier et al. [41] subsequently showed the benefits of fine-tuning large protein language models on homologous sequences. In the zero-shot setting, ESM-1v falls short of simple family-specific models – as illustrated in Table 4.2, a single ESM-1v model and an ensemble of 5 independently trained ESM-1v models respectively underperform a site-independent model and a Potts model (e.g., EVcouplings [38]). However, fine-tuning an ensemble of 5 ESM-1v models on evolutionarily-related sequences helped surpass the performance of more complex family-specific models such as DeepSequence [107] (see Table 1 in [41]).

As noted in Meier et al. [41], naively fine-tuning large family-agnostic models is non-trivial and may lead to overfitting: to achieve the performance lift discussed above, the authors introduced 'spiked fine-tuning' – a process in which the large protein language model is fine-tuned on a mix of evolutionarily-related and non-related protein sequences. As an alternative, we demonstrated in RITA [127] the benefits of *prompt tuning* [129] in which we fine tune a family-specific 'soft prompt' on evolutionarily-related sequences (keeping all other parameters of the language model frozen), and subsequently use that trained prompt to condition sequence generation for downstream tasks with that particular family.

Fine-tuning large protein language models is however time and compute-intensive for downstream tasks involving a large number of protein families. An elegant solution to that problem is offered by the MSA Transformer [130], a transformer architecture derived from the axial transformer (§ 2.2.3) used to learn a model of MSAs across thousands of protein families. It achieves very strong performance across a range of tasks (e.g., contacts prediction, fitness prediction) without relaying on fine-tuning. As noted in [41] and as we will also see in § 3.4.3, its performance is nonetheless sensitive to the characteristics of the retrieved MSAs – in particular their depths. As discussed in § 3.3.1, there are many proteins that are difficult to align (e.g., disordered proteins) or for which the obtained MSA is relatively shallow. Thus, models that are too sensitive to the depth of the MSA for good performance are limited in scope. The methods we introduce in the rest of the chapter are aimed at addressing this issue.

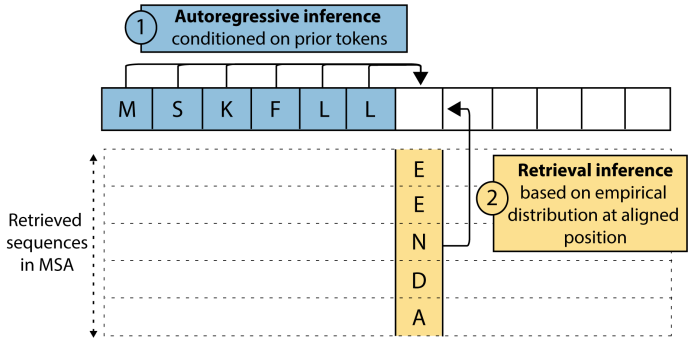

Figure 3.4: Combining autoregressive inference and retrieval inference. Predictions in Tranception are based on two complementary modes of inference: autoregressive predictions based on the context of previously generated tokens and predictions based on the empirical distribution of amino acid at each position in the retrieved set of homologous sequences.

3.4.2 Introducing retrieval at inference

We propose to augment the autoregressive inference mode from Tranception with a second inference mode – retrieval inference (Fig. 3.4) – that is based on the empirical distribution of amino acids observed across sequences in the MSA. To that end, the first step consists of retrieving a MSA at inference time for a wild-type sequence from the protein family of interest. When focusing on amino acid substitutions, the retrieved set of homologous sequences is common to both the wild-type and the mutated sequences: we do a retrieval step once per family, and amortize the cost over all mutated sequences to be scored. When dealing with insertions and deletions, we tailor the retrieved MSA to each mutated sequence by deleting columns in the MSA corresponding to deleted positions and adding zero-filled columns in the MSA at inserted positions in the mutated protein. At inference time, the inserted columns or fully non-covered positions are ignored and the model solely relies on its autoregressive mode to make predictions at these positions. The second step is to compute the empirical distribution of amino acids for each aligned position based on pseudocounts (ignoring gaps) and Laplace smoothing [131] (Appendix A.2.5). Since the distribution of sequences found in protein databases is biased by human sampling (certain organisms are more studied than others) and evolutionary sampling (groups of species that arose from large radiations will have over-represented proteins), we re-weigh sequences in the MSA using the scheme described in Hopf et al. [106]. Finally,

we estimate the log-likelihood $\log P(x)$ for a protein sequence x by a weighted arithmetic average of the log-likelihood $\log P_A(x)$ from the autoregressive inference mode and the log-likelihood $\log P_R(x)$ obtained from the retrieval inference mode. This can be equivalently viewed as a weighted geometric average in probability space, and form a proper probability distribution up to a normalization constant:

$$\log P(x) = \frac{1}{C}[(1-\alpha)\log P_A(x) + \alpha \log P_R(x)] \tag{3.5}$$

In the majority of downstream applications (e.g., § 4.2, § 4.3, § 5.4), we are only interested in the log-likelihood of a protein sequence relative to that a reference sequence (e.g., the wild-type sequence). In practice, we are mainly interested in relative log-likelihoods between sequences (as discussed in § 4.2.3) and we thus do not need to estimate C. Using equation 3.3, we further obtain:

$$\log P(x) \propto \sum_{i=1}^{l} [(1-\alpha)\log P_A(x_i|x_{<i}) + \alpha \log P_R(x_i)] \tag{3.6}$$

The above scoring decomposition as the sum of position level scores is advantageous in practice as it allows us to ignore one of the two inference modes as needed (eg., ignoring retrieval inference at positions with insertions when scoring indels). Following the sequence likelihood estimation procedure described in § 3.3.2, we first traverse the sequence in the canonical order from left to right (ie., from the N-terminus to C-terminus as discussed in § 2.5.2) and compute the sequence log probability as per equation 3.6 (in practice, all computations are performed in parallel across positions using masks in the self-attention layers to preserve autoregressiveness). We then perform the symmetric operation traversing the protein sequence from right to left, and finally average the two log-likelihood ratios.

As Tranception is trained on a diverse set of non-aligned sequences it is not subject to the biases that may stem from solely training on proteins that can be aligned, and thus exhibits higher performance on difficult to align sequences such as disordered proteins.

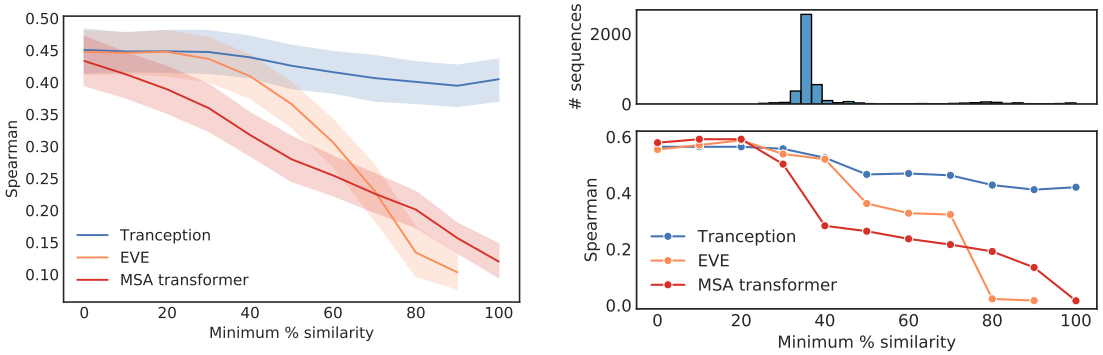

Figure 3.5: Robustness to alignment depth. We measure the average performance (Spearman's correlation with DMS measurements) of Tranception, EVE and MSA Transformer as we filter out an increasing proportion of MSA sequences based on their similarity to the seed sequence. The left figure aggregates results across all 87 substitution assays in ProteinGym (§ 4.2.2), the right figure focuses on the tumor protein P53 (Kotler et al. [132] assay; other examples are provided in Appendix A.2.6). The performance of Tranception is robust to MSA depth, while that of EVE and MSA transformer drops significantly as diversity in the MSA is reduced. Rightmost points for Tranception and MSA transformer correspond to performance with no retrieval and with a single-sequence input MSA (the seed) respectively. On the right figure, the histogram reports the number of sequences in the MSA per similarity to the seed sequence groupings.

3.4.3 Advantages of retrieval at inference time

Gain of scope for proteins with shallow alignments. Retrieving the MSA and computing the corresponding pseudocounts at inference is relatively cheap computationally. Since we only rely on aggregate statistics per position, our proposed approach is not very sensitive to the characteristics of the MSA for the protein family of interest, nor to the hyperparameters chosen to retrieve that MSA. For instance, when progressively removing sequences in the MSA based on their minimum similarity to the seed sequence (Fig. 3.5), we observe that Tranception has consistently high performance, unlike EVE or MSA Transformer. Given that one can obtain shallow MSAs for far more proteins than deep MSAs, our lightweight inference-time retrieval can be effectively leveraged for the vast majority of proteins. Disordered proteins are examples of proteins that are notoriously difficult to align, and Tranception markedly outperforms all baselines on these proteins (see § B.1.3, A4 HUMAN and GCN4 YEAST assays). Finally, our approach benefits from the additional flexibility to fully ignore the retrieval-based mode of inference when

Domain	Tranception (w/o retrieval)	Tranception (retrieval full MSA)	Tranception (retrieval domain MSA)	EVE (full MSA)	EVE (domain MSA)
RING	0.567	0.588	**0.607**	0.320	0.573
BRCT	0.354	0.490	0.504	N/A	**0.593**

Table 3.1: BRCA1 model performance summary by domain, as measured by Spearman's ρ between model scores and experimental measurements. Leveraging domain-specific MSAs helps improve performance. Tranception benefits from using domain-specific MSAs, while keeping the ability to model the large difficult-to-align region in-between the 2 domains.

scoring positions that are not present in the MSAs (e.g., indels) and proteins that are not alignable or for which the corresponding set of homologous sequences is extremely shallow (e.g., a few sequences).

Modularity and gain of coverage. A major benefit from Tranception is gain of coverage. For example, BRCA1, the gene encoding Breast cancer type 1 susceptibility protein, has 1863 amino acids making it challenging to model with alignment based methods. Trying to obtain an alignment for the full protein results in poor diversity. Consequently, not only do alignment-based models (e.g., EVE) obtain relatively weak performance when trained on such alignments (Table 3.1), but they may not be able to provide scores at all (e.g., the default minimum column coverage used in EVE is not satisfied on the BRCT domain when using a full sequence-wide alignment). It is however possible to obtain higher quality alignments and model performance when dealing with the two RING and BRCT domains of BRCA1 separately. but this comes at the cost of not being able to make predictions for the majority of the protein (outside of these two domains), and ignore potential dependencies across domains. Tranception is not subject to these coverage limitations and is able to model all potential mutations of the protein. It obtains relatively high performance without retrieval, yet benefits from leveraging the retrieval inference mode based on full-protein alignment or domain-specific alignments, with slightly higher performance for the latter.

3.4.4 TranceptEVE – A hybrid architecture with state-of-the-art fitness prediction performance

One additional advantage of retrieval at inference is that we have the freedom to leverage a more complex model of evolutionarily-related sequences at inference when it is advantageous to do so – for example to learn more complex dependencies across residues when the retrieved MSA is sufficiently deep. This observation lead us to the development of TranceptEVE, a hybrid protein language model which combines the respective strengths of the models discussed earlier in this Chapter as shown in Fig. 3.6.

In TranceptEVE, the log-likelihood at each position $\log P(x_i|x_{<i})$ is obtained as a weighted arithmetic average between the log-likelihoods from two distinct models: 1) $\log P_T(x_i|x_{<i})$ from the family-agnostic Tranception (§ 3.3.2) and 2) $\log P_E(x_i|x_{<i})$ from the family-specific EVE (§ 3.2.2). More precisely, after training the the EVE model for a family of interest, we obtain $\log P_E(x_i|x_{<i})$ by inputting into the VAE the wild-type sequence from which the MSA was acquired and then averaging the resulting log-softmax outputs from the decoder network across a large number of samples from the approximate posterior $z \sim q(z|\mathbf{x}^{wt})$ and from the distribution over decoder parameters. As such, the decoder output from the EVE model acts as a *constant* family-specific prior distribution over amino acids at each sequence position that is *independent* from the particular protein sequence we wish to model: $\log P_E(x_i|x_{<i}) = \log P_E(x_i)$. Besides preserving autoregressiveness – which is desirable for novel proteins generation, this approach presents several practical advantages over naive ensembling (Appendix A.3.6) and, critically, it allows to handle the scoring of insertions and deletions (Appendix A.3.3).

We thus estimate each residue probability in Eq. 3.3 via the following expression:

$$\log P(x_i|x_{<i}) = (1-\alpha_P)[(1-\beta_P)\log P_T(x_i|x_{<i}) + \beta_P \log P_M(x_i)] + \alpha_P \log P_E(x_i) \tag{3.7}$$

where $\log P_M$ is the empirical distribution over amino acids at each position of the retrieved MSA (i.e., the inner bracket corresponds to Tranception with

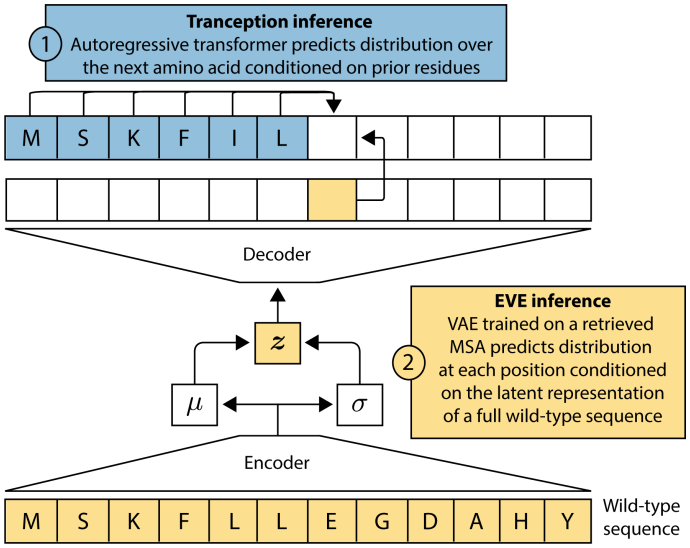

Figure 3.6: TranceptEVE combines a family-agnostic autoregressive transformer and a family-specific EVE model at inference. TranceptEVE predictions are based on two complementary modes of inference: 1) an autoregressive model – Tranception – makes predictions at a given residue based on the context of previous amino acids in the sequence; 2) a protein-specific variational autoencoder – EVE – learns a distribution over amino acids at each position based on dependencies across the full sequence observed in a retrieved Multiple Sequence Alignment.

MSA retrieval); α_P and β_P are constants that solely depends on the depth of the corresponding alignment: when the alignment is shallow we rely fully on the autoregressive predictions $\log P_T$, and when the MSA is deeper we lean more heavily on the MSA and EVE log priors ($\log P_M$ and $\log P_E$ resp., § A.3.2). Additional details about sequence likelihood inference are provided in Appendix A.3.1). Since the aggregation coefficients between the autoregressive transformer and the EVE model are dependent on the depth of the retrieved MSA, our approach gracefully adapts to all types of proteins: if the protein has few homologs, we mainly rely on the autoregressive transformer; if the retrieved MSA is deep, we lean more heavily on the family-specific EVE model.

4
Predict

Contents

4.1 Introduction

In this chapter, we demonstrate how the models introduced in § 3 can be utilized to predict useful properties of the corresponding biological objects. Specifically, we explore three main applications of these models. First, in § 4.2, we demonstrate how the models enable protein fitness prediction, both in zero-shot (§ 4.2.3) and

supervised settings (§ 4.3). Second, in § 4.4, we show how these models can be used to aid the clinical annotation of genetic mutations in humans. Finally, in § 4.4, we discuss the potential use of these models to predict viral escape, which is a critical task for pandemic preparedness efforts and vaccine design.

4.2 Predicting protein fitness

4.2.1 Background and objectives

The fitness of a protein is a measure of how well a protein can perform its function within an organism. Factors that influence protein fitness are diverse and include for example its stability, folding efficiency, catalytic activity (for enzymes), binding specificity and affinity. Protein fitness is closely related to the concept of evolutionary fitness, as proteins with increased fitness are more likely to be conserved and passed on to future generations, while proteins with lower fitness may be subject to negative selection pressure and eventually be eliminated from the gene pool. Predicting the fitness of proteins is therefore a key task to understand the effects of mutations in organisms and towards more effective protein engineering. In this section we are concerned with learning a mapping between the sequential representation of a protein and its fitness, which is often referred to as learning the 'fitness landscape'. To that end we will leverage the family-specific, family-agnostic and hybrid models discussed in § 3 and who how these methods can be used to predict the fitness of proteins in the zero-shot and supervised settings.

4.2.2 ProteinGym benchmarks

ProteinGym is an extensive set of Deep Mutational Scanning (DMS) assays (§ 2.5.4) curated to enable thorough comparisons of various mutation effect predictors in different regimes. ProteinGym is comprised of two benchmarks: 1) a *substitution benchmark* which consists of the experimental characterisation of ∼1.5M missense variants across 87 DMS assays 2) an *indel benchmark* that includes ∼300k mutants across 7 DMS assays.

The relationship between protein fitness measured experimentally and as reflected by the distribution of sequences selected by evolution is complex. The fitness landscape of naturally occurring proteins is the result of an intricate set of overlapping constraints that these proteins are subjected to in an organism. Consequently, it is often challenging to identify a single molecular property that is both easy to measure experimentally and that reflects that complexity. To build this benchmark we therefore prioritized assays where both the experimentally-measured property for each mutated protein is expected to reflect the role of the protein in organism fitness as well as - where available - their quality measured via experimental replicates. The resulting set of DMS assays covers a wide range of functional properties (eg., thermostability, ligand binding, aggregation, viral replication, drug resistance), spans a diverse protein families (e.g., kinases, ion channel proteins, g-protein coupled receptors, polymerases, transcription factors, tumor suppressors) and different taxa (humans, other eukaryotes, prokaryotes, viruses).

ProteinGym is the largest and most diverse set of DMS experiments specifically targeted at the task of variant effect prediction. It contains more than twice the number of assays and variants present in the DeepSequence benchmark [107], which had been created for the same purpose (Table 4.1). While most of the curated DMS assays (in our benchmark or others) probe the effect of single amino-acid substitutions, our collection also includes several multiple amino-acid variants which are critical to assess the ability of models to extrapolate further away in sequence space from the naturally occurring proteins they are trained on. Lastly, as most mutation effect predictors are not able to quantify the effect of insertions and deletions, indels have been absent from the majority of prior benchmarks. We expand on the set of DMS available in Shin et al. [112] and Dallago et al. [133] to address this gap.

The relationship between protein function and organism fitness has been shown to often be non-linear [134]. As such, we use Spearman's rank correlation coefficient between model scores and the experimental measurements as the standard measure of model performance [41, 107]. In certain instances, the DMS measurements are

Measure	Category	DeepSequence	ProteinGym	Fold increase
Number of assays by taxon	Human	9	34	3.7
	Other eukaryotes	10	14	1.4
	Prokaryotes	13	24	1.8
	Virus	5	22	4.4
	All taxa	**37**	**94**	**2.5**
Number of variants by type	Single substitutions	0.12M	0.36M	2.9
	Multiple substitutions	0.55M	1.26M	2.3
	Indels	0	0.27M	-
	All variants	**0.67M**	**1.89M**	**2.8**

Table 4.1: Comparison of the ProteinGym and DeepSequence benchmarks. ProteinGym contains a substantially higher number of assays and variants compared to DeepSequence. It addresses notable gaps such as the limited number of viral DMS assays, limited multiple substitutions assays and absence of indels benchmark.

characterized by a bimodal profile for which rank correlations are not well suited. To that end, we provide additional measures of model performance: the area under the ROC curve (AUC) and Matthews correlation coefficient (MCC) between model scores and the experimental measurements (Appendix B.1.3).

4.2.3 Zero-shot fitness prediction

Method

We compare the ability of various models to predict the effects of mutations across the DMS assays in ProteinGym. We focus on the main approaches described in § 3, including a number of family-specific alignment-based methods – Site independent model and EVmutation [106], DeepSequence [107], EVE [135]; family-agnostic models such as Unirep [39], ESM-1b (latest model checkpoint from Rives et al. [40] with extensions from Brandes et al. [136]), ESM-1v [41], Progen2 [126], RITA [127] and Tranception (without retrieval) [137]; and hybrid models such as Unirep with evotuning [39], the MSA Transformer [130], Tranception (with retrieval) [137] and TranceptEVE [138]. Although technically trained on subsets of unaligned sequences, Wavenet [112] models are protein-specific and use MSAs to extract their training data. Consequently, we group them with other family-specific models.

We next describe the scoring method for the different models we introduced in § 3, and include all details for all baselines in Appendix B.1.3.

EVE We define the *evolutionary index* of a protein variant $\mathbf{s}$ as the relative fitness of $\mathbf{s}$ compared with that of a wild-type sequence $\mathbf{w}$. Building from the probabilistic viewpoint that gave rise to the Bayesian VAE described in § 3.2.2, the fitness of a sequence may be measured by the difference in log-likelihood of that sequence with the wild type. Since estimating the exact log-likelihood is intractable in practice, we use as an approximation the negative ELBO, a bound on the log marginal likelihood (§ 2.2.2). The evolutionary index E_s of protein $\mathbf{s}$ may therefore be expressed in terms of a tractable difference between two ELBOs:

$$E_s \sim \log \frac{p(\mathbf{s}|\theta_\mathbf{p})}{p(\mathbf{w}|\theta_\mathbf{p})} \sim ELBO(\mathbf{s}) - ELBO(\mathbf{w}) \tag{4.1}$$

For each variant $\mathbf{s}$ of interest, since the integral over $\mathbf{z}$ is intractable in practice, we estimate ELBO values via Monte Carlo sampling of the latent space, taking a large number of samples (20k samples) from the approximate posterior distribution $q(\mathbf{z}|\mathbf{s}, \phi_\mathbf{p})$.

Tranception and TranceptEVE Similarly for Tranception and TranceptEVE, we assess the relative fitness $F(\mathbf{s})$ of a mutated sequence $\mathbf{s}$ as the log-likelihood ratio between $\mathbf{s}$ and a naturally occurring reference sequence $\mathbf{w}$ for that protein family:

$$F(\mathbf{s}) = \log \frac{P(\mathbf{s})}{P(\mathbf{w})} \tag{4.2}$$

We then plug in the expressions obtained in Eq. 3.6 and Eq. 3.4.4 respectively.

Results

ProteinGym substitution benchmark We compute the Spearman's rank correlation coefficient ρ and AUC between model scores and experimental measurements for all DMS assays in the ProteinGym substitution benchmark, and report aggregated results by MSA depth grouping in Table 4.2 and by mutational depth in Table 4.3. Performance at the DMS level, by taxon and for other metrics (e.g., MCC) are provided in Appendix B.1.3. We find that TranceptEVE, the state-of-the-art hybrid model introduced in § 3.4.4, outperforms all other baselines on all benchmarks, with markedly higher performance in the regime of low-depth

Model type	Model name	Spearman by MSA depth ↑ Low	Medium	High	All	AUC ↑ All
Alignment-based models	Site independent	0.434	0.378	0.309	0.375	0.715
	Wavenet	0.312	0.396	0.457	0.391	0.724
	EVmutation	0.404	0.405	0.444	0.413	0.732
	DeepSequence (ensemble)	0.393	0.403	0.498	0.421	0.737
	EVE (ensemble)	0.423	0.441	0.498	0.449	0.754
Protein language models	Unirep	0.224	0.149	0.159	0.166	0.595
	ESM-1b	0.344	0.327	0.463	0.358	0.706
	RITA (ensemble)	0.332	0.409	0.385	0.388	0.722
	ESM-1v (ensemble)	0.372	0.373	**0.510**	0.401	0.731
	Progen2 (ensemble)	0.367	0.416	0.447	0.413	0.735
	Tranception (w/o retrieval)	0.379	0.399	0.429	0.401	0.727
Hybrid models	Unirep (evotuned)	0.314	0.351	0.373	0.348	0.700
	MSA Transformer (ensemble)	0.398	0.426	0.485	0.432	0.745
	Tranception	0.447	0.436	0.472	0.446	0.753
	TranceptEVE	**0.460**	**0.463**	**0.508**	**0.472**	**0.767**

Table 4.2: Average AUC and Spearman's rank correlation between model scores and experimental measurements by MSA depth on the ProteinGym substitution benchmark. Alignment depth is measured by the ratio of the effective number of sequences N_{eff} in the MSA, following [38], by the length covered L (Low: N_{eff}/L <1; Medium: 1< N_{eff}/L <100; High: N_{eff}/L >100). Statistical significance analysis reported in Table B.3 in Appendix B.1.3

MSAs and on multiple mutants, with the lift increasing with mutation depth. When analyzing performance at the taxon level (Table B.1), we observe consistently high performance from TranceptEVE across all categories, in particular on human proteins and viruses, which we will cover in greater depth in the rest of this Chapter. Tranception (with retrieval) performs on par with EVE, with the former performing better on proteins with shallow alignments, and the latter performing better on proteins with deeper alignments.

ProteinGym indel benchmark We also report the performance the ProteinGym indel benchmark (Table 4.4) and contrast the performance of Tranception and TranceptEVE against all baselines that are capable of scoring indels. As explained in § 3.3, family-specific methods trained on aligned data are typically unable to score indels due to the fixed coordinate system they are trained with. However, family-specific methods trained on unaligned data (e.g., Wavenet) offer a means to

Model type	Model name	Spearman by mutation depth ↑ 1	2	3	4	5+	All
Alignment-based models	Site independent	0.375	0.322	0.285	0.321	0.407	0.375
	Wavenet	0.389	0.344	0.324	0.282	0.35	0.391
	EVmutation	0.414	0.401	0.403	0.335	0.427	0.413
	DeepSequence (ensemble)	0.419	0.394	0.407	0.353	0.436	0.421
	EVE (ensemble)	0.45	0.409	0.405	0.351	0.429	0.449
Protein language models	Unirep	0.161	0.201	0.097	0.126	0.192	0.166
	ESM-1b	0.351	0.318	0.215	0.161	0.318	0.358
	RITA (ensemble)	0.380	0.236	0.135	0.181	0.246	0.388
	ESM-1v (ensemble)	0.400	0.309	0.203	0.165	0.253	0.401
	Progen2 (ensemble)	0.410	0.312	0.233	0.228	0.282	0.413
	Tranception (w/o retrieval)	0.393	0.398	0.418	0.334	0.422	0.401
Hybrid models	Unirep (evotuned)	0.339	0.284	0.279	0.246	0.323	0.348
	MSA Transformer (ensemble)	0.433	0.374	0.401	0.351	0.418	0.432
	Tranception	0.442	0.427	0.440	0.370	0.461	0.446
	TranceptEVE	**0.472**	**0.444**	**0.456**	**0.389**	**0.464**	**0.472**

Table 4.3: Average Spearman's rank correlation ρ between model scores and experimental measurements by mutation depth.

score indels. Moreover, family-agnostic methods trained using a masked-language model objective (e.g., ESM-1v, ESM-1b and MSA Transformer) and for which fitness is assessed through the masked-marginal scoring heuristic [41] cannot handle indels either, since the masked-marginals approach requires the position to exist in the wild-type sequence. We find that Tranception and TranceptEVE outperform other baselines on this benchmark as well. Not surprinsingly, there is a negligible difference between the performance of the two models on indels since the retrieved EVE models are unable to help with inserted or deleted residues for the reasons mentioned above. Thus predictions rely mainly on the underlying autoregressive transformer which is common to the two models.

4.2.4 ProteinNPT - A semi-supervised pseudo-generative model for protein fitness prediction and design

In cases where we have collected a sufficiently large number of labels for the property that we seek to predict, supervised methods may help reach even high performance over the fully unsupervised method discussed in § 4.2.3.

Model name	Spearman ↑	AUC ↑
Progen2 (ensemble)	0.407	0.748
Unirep (evotuned)	0.409	0.728
Wavenet	0.412	0.740
RITA (ensemble)	0.446	0.761
Tranception (w/o retrieval)	0.430	0.756
Tranception	**0.464**	**0.773**
TranceptEVE	**0.466**	**0.774**

Table 4.4: Average AUC and Spearman's rank correlation between model scores and experimental measurements on the ProteinGym indel benchmark.

The ProteinNPT (Fig. 4.1) is a semi-supervised pseudo-generative model that learns a joint representation of protein sequences and associated property labels. It leverages axial attention [54] (§ 2.2.3) in various ways throughout the architecture to achieve stat-of-the-art performance on supervised fitness prediction tasks (§ 4.2.5). Although we focus on protein fitness in this section, the labels can in theory represent any other characteristic of the corresponding proteins (e.g, thermostability, solubility). The ProteinNPT builds on several works from the self-attention literature, in particular the Axial Transformer [54], the MSA Transformer [130] and Non-Parametric Transformers [139]. We detail how they relate to and differ from one another in Appendix B.1.4.

Model architecture

The model takes as input *both* the primary structure representation of the proteins along with the corresponding labels for the property of interest. Depending on whether we are at training or inference time, we mask different parts of this combined input as per the procedure described later in this section. We separately embed protein sequences and labels, concatenate the resulting sequence and label embeddings into a single tensor, which we then feed into several ProteinNPT layers. A ProteinNPT layer (Fig. 4.1 (right)) learns joint representation of protein sequences and labels by applying successively self-attention between residues and labels for a given sequence (row-attention) and then self-attention across sequences in the input batch at a given position (column-attention). In the final stage, the

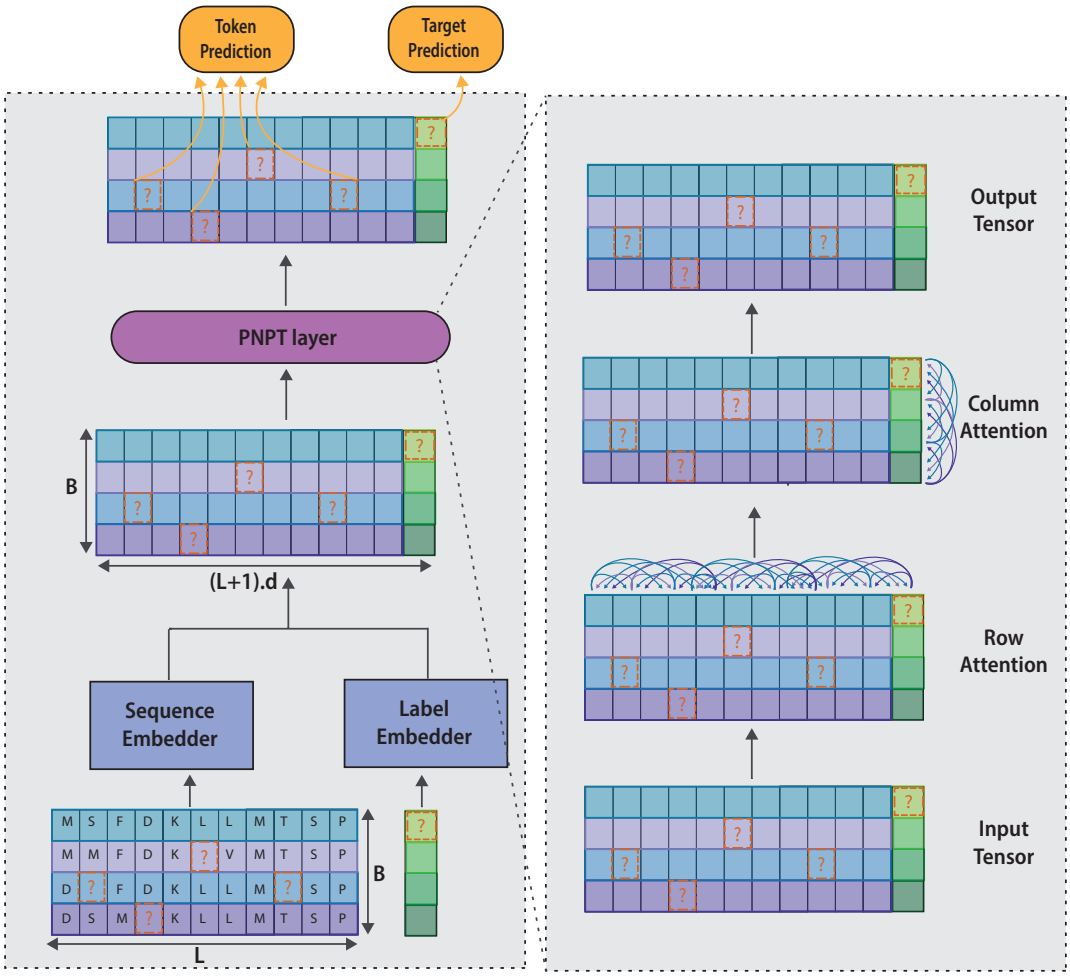

Figure 4.1: ProteinNPT architecture diagram (Right) The model takes as input the primary structure of a batch of proteins along with the corresponding labels, embed them separately, and concatenate them into a single tensor. Several ProteinNPT layers are then applied to learn a representation of the entire batch, which is ultimately used to predict both masked tokens and targets (depicted by question marks). (Left) A ProteinNPT layer alternates between row and column attention to learn rich embeddings of the labelled batch

learned embeddings from the last layer are used to predict both the masked tokens and targets: the embeddings of masked targets are input into a L2-penalized linear projection to predict masked target values, and the embeddings of masked tokens are linearly projected then input into a softmax activation to predict the corresponding original tokens.

To obtain the initial protein sequence embeddings, we experimented with several best-in-class protein language models and obtained better results with the MSA

Transformer (Appendix B.1.4). As a result, our overall architecture operates *tri-axial* self-attention to output predictions: across residues and labels, across homologous sequences from the retrieved MSA (in MSA Transformer layers only) and across labelled sequences (in ProteinNPT layers only).

There are several advantages conferred by this architecture. First, it seamlessly adapts to multiple target predictions, by concatenating as many label columns in its input as required. Second, it naturally captures correlations between these targets and automatically handles – at training and at inference – observations with partially available labels. Third, to avoid overfitting to sparse label sets, prior semi-supervised approaches typically apply a pooling operator (e.g., average pooling across the full sequence length) to reduce the dimensionality of input features. This pooling operation potentially destroys valuable information for the downstream task. In contrast, no such pooling operation is applied in the ProteinNPT architecture, and the network leverages self-attention to learn dependencies between labels and the embeddings of specific residues in the sequence.

Model training

Our overall architecture is trained with a two-stage semi-supervised procedure. First, due to the scarcity of labelled instances relative to the size of the underlying protein sequence space, we leverage embeddings from protein language models that have been pre-trained on large quantities of *unlabelled* natural sequences. Throughout training, the parameters of the model used to obtain these protein embeddings are frozen to prevent overfitting. This also helps alleviating several practical challenges pertaining to GPU memory bottlenecks at train time (Appendix B.1.4). Second, similarly to what was proposed in Non-Parametric transformers [139], we train our architecture on a composite task combining input sequence denoising and target prediction. In our case, the former is equivalent to a masked language modeling objective on amino acid tokens [37]. We randomly mask a fraction of amino acids and of input labels across all sequences in the mini batch, and seek to predict these masked items. Our training loss is thus comprised of two components: 1) a

reconstruction loss over masked amino acids 2) a target prediction loss over masked targets (e.g., mean squared error for continuous targets).

Inference

At inference, we form input batches by concatenating row-wise a fraction of test sequences (with unknown, masked labels) with training instances (with known, unmasked labels). Similar to the original NPT architecture, inference is therefore achieved in a semi-parametric fashion and predictions for a given sequence depends on the other sequences present in the mini batch. When the size of the training set is large enough, we cannot use the entire training set at inference due to memory bottlenecks, and randomly sample a fixed subset of labelled training sequences to form input batches.

Auxiliary labels

Critical to the performance attained by our architecture is the use of *auxiliary labels* at training and inference. We define auxiliary labels as additional inputs which are both 1) easy to obtain for all or for a subset of train and test instances 2) known to be correlated with the target of interest. They are concatenated column-wise to the input batch and are handled by ProteinNPT layers in the forward pass just like any other target. However, they are excluded from the loss computation and are only meant to increase performance on the target prediction task by instilling complementary information into the model. In the fitness prediction results discussed in § 4.2.5, we use as auxiliary labels the *unsupervised fitness predictions* obtained with the underlying protein language model used to extract the input sequence embeddings. As shown in ablations (Appendix B.1.4), and consistent with the findings from Hsu et al. [140], this helps significantly increase our prediction performance, in particular on the 'modulo' and 'contiguous' cross validation schemes.

4.2.5 Supervised fitness prediction

Method

Cross-validation schemes We leverage all assays from the ProteinGym substitution benchmark as discussed in § 4.2.2, to which we add another 13 new assays bringing our total to 100 DMS assays (Appendix B.1.5). This is referred to as the 'extended benchmark' in the rest of the text. We use three distinct cross-validation strategies to assess the capacity of each model to extrapolate to positions not encountered during training. In the 'Random scheme', a commonly used method in supervised fitness prediction benchmarks [141, 142], each mutation is randomly allocated to one of five distinct folds. In the 'Contiguous scheme', the sequence is split into five segments along its length, with mutations assigned to each segment based on the position they occur in the sequence. This scheme ensures that mutations in each fold come from contiguous sequence positions. The 'Modulo scheme' uses the modulo operation to assign mutated positions to each fold. For example, position 1 is assigned to fold 1, position 2 to fold 2, and so on, looping back to fold 1 at position 6. This pattern continues throughout the sequence.

Baselines We start from the top-performing model variant from [140], namely a ridge regression on one-hot encodings of protein sequences, augmented with unsupervised fitness predictions from DeepSequence [107]. Building on recent progress from the protein property prediction and engineering literature, we then improve this baseline in several ways. First, we show that augmenting the model from Hsu et al. [140] with the latest state-of-the-art unsupervised fitness predictors helps further boost the predictive performance, with the performance boost being consistent with the zero-shot performance of the corresponding unsupervised models (Appendix B.1.4). Then, considering the performance across the various cross validation schemes described in the prior paragraph, we show the limits of one-hot encoded features to generalize across positions and propose instead to use embeddings from large protein language models. There is a rich literature investigating this idea, and we test out a wide range of model architectures that have

been proposed to learn a mapping between embeddings and labels: L2-penalized regression, shallow Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), ConvBERT [143] and Light Attention [144]. Combining these different parts of literature together, we obtain the main baselines we compare against in subsequent sections: models that both leverage embeddings from pretrained language models, and are augmented with unsupervised fitness predictions. We carry out thorough ablations and hyperparameter search (Appendix B.1.4) to identify the best performing combinations.

Results

Single mutant property prediction We first focus on the task of predicting the fitness of single mutants, considering the three distinct cross validation schemes discussed above. For each assay and cross validation scheme, we perform a 5-fold cross validation, selecting the first 4 folds for training, and using the remaining one as test set. Intuitively, since the 'Random' cross validation scheme is such that mutations in the test set may occur at the same positions as certain mutations observed in the training data, model performance will typically be higher in that regime. Similarly, the 'Contiguous' scheme is intuitively the most difficult regime in which we assess the ability of the different models to extrapolate on positions relatively 'far' in the linear sequence from any mutation being part of the training data. We find that the ProteinNPT markedly outperforms all baselines, across all cross validation schemes, and both in terms of Spearman's rank correlation and Mean Squared Error (MSE) (Table 4.5). Our results also confirm that a model that would learn solely based on one-hot encoded inputs would be unable to make any valuable prediction at positions that were not observed at training time (Spearman's rank correlation near zero on the 'Contiguous' and 'Modulo' schemes). Augmenting these models with predictions from unsupervised predictors helps to generally address that issue, although the performance on the 'Contiguous' and 'Modulo' schemes is equal to the zero-shot performance of unsupervised model (the average Spearman's rank correlation for the MSA Transformer in the zero-shot setting is

0.41, when considering all 100 assays in the extended benchmark). The baseline combining a transform of embeddings from the MSA Transformer, augmented with unsupervised fitness predictions ('Embed. - Aug. (MSAT) in Table 4.5) performs better across the board than the one-hot-encoding equivalent, yet its performance is far from that of the ProteinNPT, in particular in the 'Random' cross validation scheme. We interpret the latter by the fact the column attention model in the ProteinNPT layer (Fig. 4.1) is more beneficial to performance when a mutation at that position has been observed at train time.

	Spearman (↑)				**MSE** (↓)			
Model name	**Contig.**	**Mod.**	**Rand.**	**Avg.**	**Contig.**	**Mod.**	**Rand.**	**Avg.**
OHE	0.08	0.02	0.54	0.21	1.17	1.11	0.92	1.06
OHE - Aug. (DS)	0.41	0.40	0.49	0.43	0.98	0.93	0.78	0.90
OHE - Aug. (MSAT)	0.41	0.40	0.50	0.44	0.97	0.92	0.77	0.89
Embed. - Aug. (MSAT)	0.47	0.49	0.57	0.51	**0.93**	0.85	0.67	0.82
ProteinNPT	**0.48**	**0.51**	**0.66**	**0.55**	**0.93**	**0.83**	**0.53**	**0.77**

Table 4.5: Fitness prediction performance - Extended ProteinGym benchmark. We compute the Spearman's rank correlation between model predictions and experimental measurements, averaged across the 100 DMS assays from the extended ProteinGym benchmark. DS and MSAT are shorthand for DeepSequence and MSA Transformer respectively. "Aug." indicates models augmented with unsupervised predictions. The Spearman for the MSA Transformer unsupervised is 0.41 (independent of CV split).

Multiple mutants property prediction

We next consider all assays in the extended ProteinGym benchmark that include multiple mutants. Given the challenges to assign multiple mutants to non-overlapping position sets, we only consider the 'Random' cross validation scheme in this analysis. We report the Spearman's rank correlation in Fig. 4.2 and MSE in Appendix B.1.5 (conclusions are consistent for the two metrics). The performance lift of ProteinNPT observed in the single mutants analyses translates to multiple as well: ProteinNPT has superior aggregate performance and outperforms all other baselines in 14 out of the 17 assays with multiple mutants (Fig. 4.2 left). The performance lift is also robust to the mutational depth considered (Fig. 4.2 right).

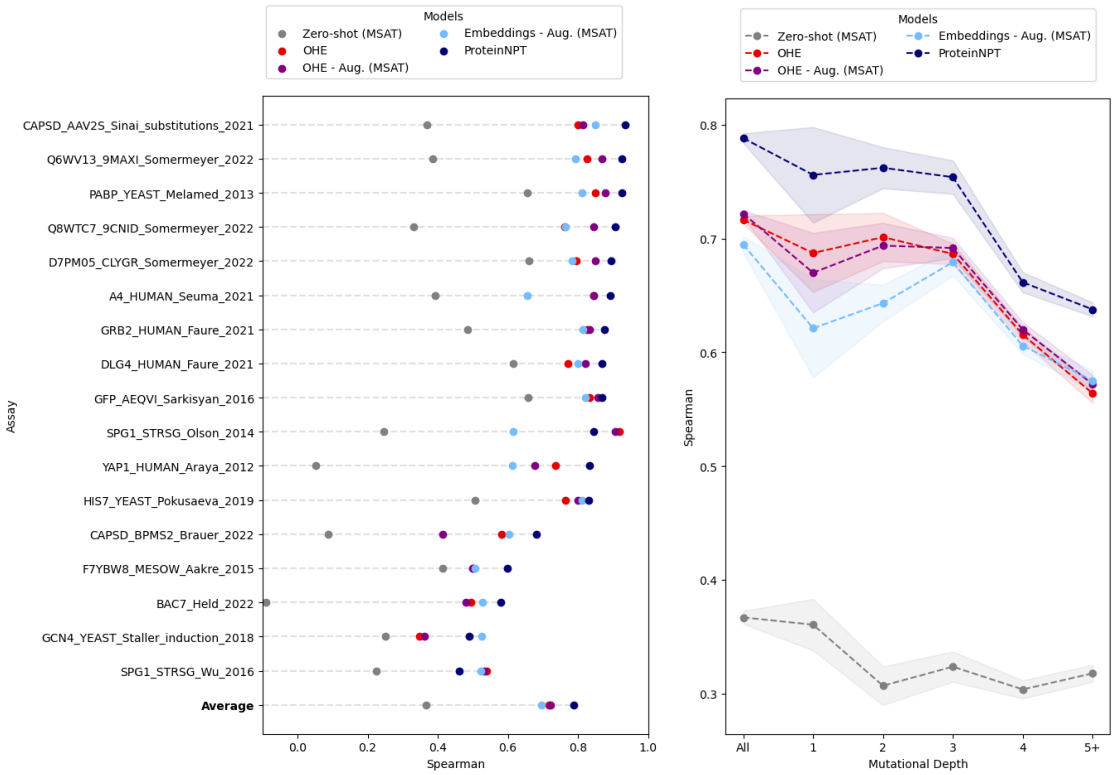

Figure 4.2: Multiples mutants prediction performance. (Left) Spearman's rank correlation between model predictions and experimental measurements, for each assay in ProteinGym with multiple mutants. (Right) Average Spearman's rank correlation, overall and by mutational depth.

Multiple property prediction

ProteinGym contains a subset of protein families for which several experimental assays have been carried out, offering the possibility to test out the ability of various models to predict multiple properties of interest simultaneously. We report the performance on the 'Random' cross validation scheme in Fig. 4.3, and other schemes are provided in Appendix B.1.5. The ProteinNPT outperforms all other baselines in that regime as well, consistently across all settings analyzed.

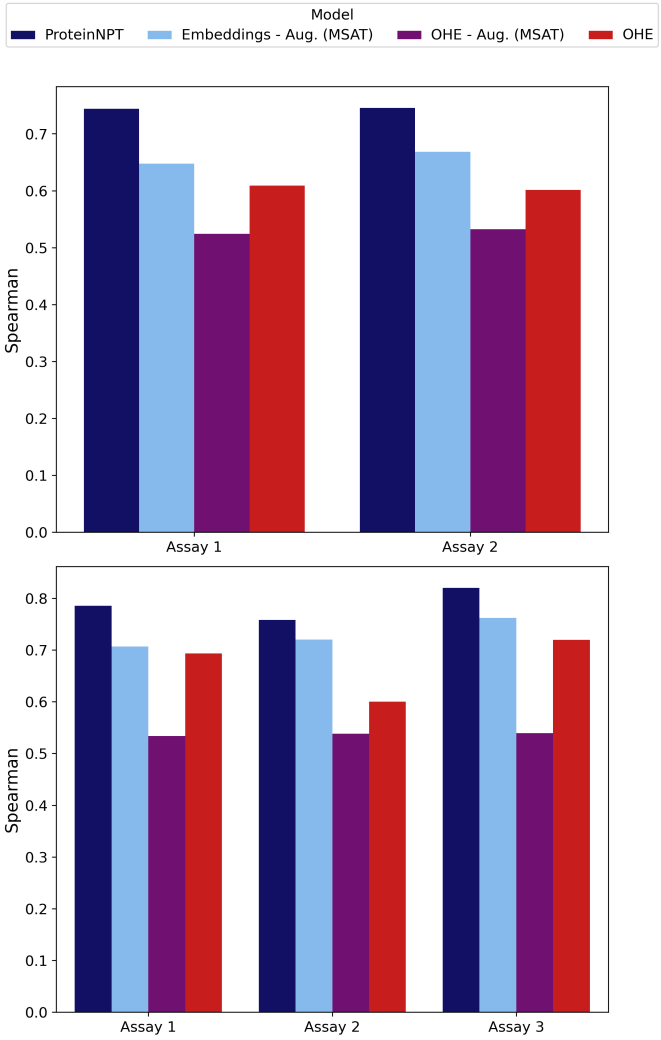

Figure 4.3: Multiple targets prediction performance. Avg. Spearman's rank correlation between model predictions and experimental measurements, for proteins with 2, 3 or 4 distinct experimental measurements predicted simultaneously (respectively top, middle and bottom plots)

4.3 Predicting the effects of human genetic mutations

4.3.1 Background and objectives

The rapid increase in human genome sequencing has highlighted the extensive genetic variation in the human population. Understanding the medical implications of these variations could revolutionize healthcare, motivating substantial investments in gathering human genomic information along with demographic and clinical data with efforts such as the UK BioBank [145], ChinaMAP [146], and deCODE

[147]. Associating specific genomic changes to disease phenotypes remains an open challenge given the vast number of variants in the human population, which drastically surpasses the variants we can feasibly investigate. Even just within protein-coding regions, significant variations exist between individuals; to date, an astounding 6.5 million missense variants have been recorded (gnomAD [148]), with a staggering 98% of these, even within disease-related genes, still undefined [34]. To address this challenge, new experimental technologies such as Deep Mutational Scans § 2.5.4 have emerged. These methods can assess the effects of thousands of mutations in parallel, and the corresponding results are then analyzed by expert panels such as ClinGen [149] for assigning clinical interpretation to human variants. However, these experimental methods can only explore a fraction of the theoretical genetic space, and depend on the availability of assays relevant to human disease phenotypes. Computational methods could potentially speed up clinical variant interpretation. Nonetheless, state-of-art computational methods [150–155] are supervised on clinical labels in a way that causes inflated accuracy in real-world prediction scenarios. This inflated performance results from labelling biases, label sparsity, label noise [156] and data leakage [157]. As a result, these methods are considered as "weak evidence" for variant classification by the guidelines from The American College of Medical Genetics and Association for Molecular Pathology (ACMG-AMP) [158]. By contrast, unsupervised probabilistic models of evolutionary sequences, such as the ones discussed in § 3.2.1, have been remarkably successful at predicting the effects of variants on protein function and stability [38, 159–161] and do not suffer from the aforementioned label issues. Yet, little progress has been made in using these models to determine disease relevance. In this section we introduce a new computational method for the classification of human genetic variants based on the Bayesian Variational Autoencoder introduced in § 3.2.2. We show that it outperforms current state-of-the-art computational methods at predicting variant pathogenicity (without the risk of overfitting clinical labels), and is surprisingly as accurate as predictions from high-throughput experiments. Lastly, we also show how TranceptEVE (§ 3.4.4) helps to further boost the prediction performance.

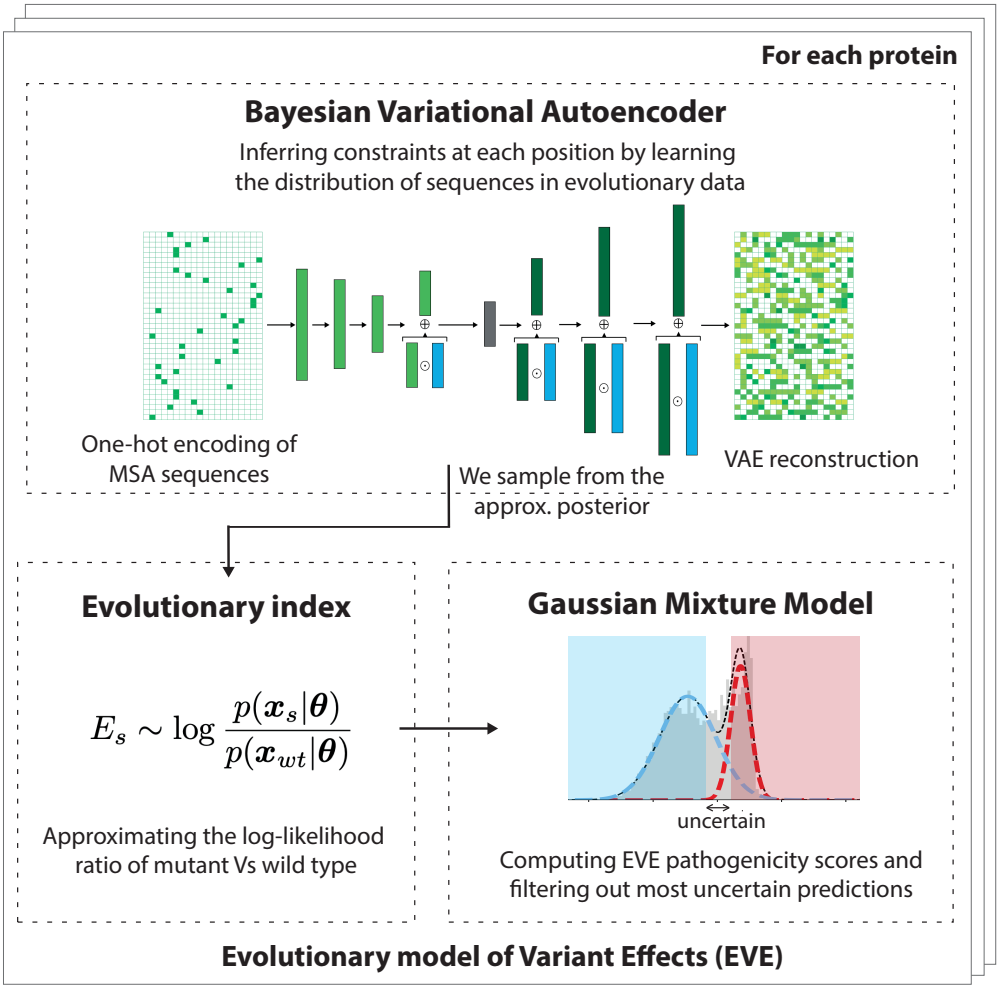

Figure 4.4: EVE - Overall modeling strategy. For each protein, a Bayesian VAE (top) learns a distribution over amino acid sequences in a multiple sequence alignment (MSA) of evolutionary data. This enables the computation of the evolutionary index (bottom left) for each single-variant sequence, which approximates the negative log-likelihood ratio of variant (X_v) versus wild-type (X_{WT}) sequences. A global-local mixture of a Gaussian mixture model (bottom right) separates variants into benign (blue dashed line) and pathogenic (red) clusters based on that index. The outcome of the model is both a continuous score that reflects pathogenicity propensity, and probabilistic assignment to benign and pathogenic classes (blue and red shaded areas, respectively) below a user-defined uncertainty threshold

4.3.2 Method

Overall process for predicting the effects of human mutations

Our overall approach learns the propensity of human missense variants to be pathogenic from the distribution of evolutionarily related sequences (Fig.4.4). In a first step, we capture constraints from natural sequences across evolution, including

complex dependencies between positions, by learning the distribution of amino acid sequences for each protein using the Bayesian Variational Autoencoder introduced in § 3.2.2 [1]. For each human protein of interest, a separate Bayesian VAE is trained on a multiple sequence alignment retrieved by searching 250 million protein sequences in UniRef [162] (Appendix A.1). After training on evolutionary sequences, we estimate the relative likelihood of each single amino acid variant with respect to the wild type – what we call the "evolutionary index" (see § 4.2.3) – by sampling from the approximate posterior distribution learned by the VAE. When comparing this evolutionary index against clinical labels, the value that separates pathogenic from benign labels is remarkably consistent across proteins (Fig. 4.5), suggesting we may use unsupervised methods to infer pathogenicity. Therefore, in a second step, rather than using (semi-)supervised learning to map scores to label categories, we fit a two-component global-local mixture of Gaussian Mixture Models (GMM) on the distributions of evolutionary indices for all single amino acid variants across proteins (§ 4.3.2). The output of this process is both the EVE score – a continuous pathogenicity score defined over the interval [0,1], with zero being most benign and one being most pathogenic – and class assignments. For these assignments, we use the predictive entropy of the GMM as a measure of classification uncertainty, and bin variants into one of three categories: Benign, Uncertain or Pathogenic (§ 4.3.2).

Separating pathogenic and benign variants with probabilistic clustering

Evolutionary indices show very strong correlations with existing clinical labels across proteins (Fig. 4.5a). We fit a Gaussian Mixture model with two components directly on the evolutionary index distribution, in order to automatically separate variants into Pathogenic and Benign clusters. Besides the advantage of being fully unsupervised, performing probabilistic clustering of variants allows us to quantify our uncertainty about the class assignment (see § 4.3.2).

[1] Strictly speaking, EVE refers to the entire modeling strategy illustrated in Fig. 4.4. However, for the sake of simplicity, we also refer to the Bayesian VAE as the 'EVE model' throughout this manuscript, as it represents the core component of the modeling process.

The Gaussian Mixture Model (GMM) [163] is a probabilistic model that assumes the data are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. We experiment with different architectures and training algorithms: single model Vs hierarchical, training with Variational Inference (VI) Vs Expectation-Maximization (EM) algorithm. We obtain the best performance on the downstream task with the following approach. We first train an overarching two-component GMM on the distribution of evolutionary indices for all single amino acid variants for all 3,219 proteins combined. We use the resulting parameters to initialize the Gaussians of the protein-specific GMMs, fit on all single amino acid variants for each protein separately. Intuitively, the cluster with higher mean will contain sequences with higher evolutionary indices, and therefore less likely sequences under the learnt sequence distribution. For each model after convergence, we thus define the component with the highest mean as the Pathogenic cluster, and the other one as the Benign cluster. We then form a global-local mixture of GMMs to combine the cluster predictions from the overarching GMM with the predictions from the protein-specific GMM for each protein. More formally, for a given protein $\mathbf{s}$ with evolutionary index $\mathbf{E_s}$, from a protein family p, we have:

$$p(\mathbf{X_s} = 1|\mathbf{E_s}) = \alpha * p(\mathbf{X_s} = 1|\mathbf{E_s}, \theta_p) + (1 - \alpha) * p(\mathbf{X_s} = 1|\mathbf{E_s}, \theta_o) \tag{4.3}$$

where X_s is a binary random variable equal to 1 if $\mathbf{s}$ is pathogenic (0 if benign), α represents the relative weight of protein-level GMM in the ensemble (set to 0.3 via grid search with respect to average accuracy and AUC), θ_o and θ_p are the parameters of the overarching GMM and protein-specific GMM respectively. $p(\mathbf{X_s} = 1|\mathbf{E_s})$ is what we refer to as the *EVE score* which quantifies the propensity of a given variant to be pathogenic (Fig. 4.5b).

Quantifying the uncertainty in the class assignment

A crucial benefit of a probabilistic clustering approach is the ability to identify the set of variants for which the classification is the most uncertain. We measure the

total uncertainty on the cluster assignment for a protein **s** via the corresponding predictive entropy PE:

$$PE = -\log p(\mathbf{X_s} = 1|\mathbf{E_s})p(\mathbf{X_s} = 1|\mathbf{E_s}) - \log p(\mathbf{X_s} = 0|\mathbf{E_s})p(\mathbf{X_s} = 0|\mathbf{E_s}) \quad (4.4)$$

As we increase the proportion of variants excluded from our classification based on their predictive entropy (excluding higher values first), the accuracy monotonically increases (Fig. 4.6). This confirms the ability of the predictive entropy metric to properly identify uncertain variants.

4.3.3 Results

Predicting pathogenicity from evolution

We apply EVE to a set of 3,219 human genes that have been associated with disease in ClinVar [34] (Appendix B.2.1). Our model is predictive of clinical significance for all labeled variants across all genes (average AUC 0.91, Fig. 4.6a) including 60 'clinically actionable' genes. Furthermore, the performance of EVE is robust to the number of labels per protein (Fig. 4.6b) suggesting generalizability to genes with less (or no) annotation, as we would expect from an unsupervised approach. EVE outperforms all pre-existing supervised and unsupervised methods at predicting known clinical labels (Fig. 4.7). This is despite a large fraction of these labels being used in training the top performing methods, as well as, in some cases, being used extensively in defining labels. As a second benchmark which avoids some of these circularities, we compare the model predictions against 40k experimentally measured variants across 10 proteins (Appendix B.2.2). Since these experiments are, in principle, independent of the ClinVar labeling process, we expect this benchmark to provide a less biased estimate of performance, albeit for a comparatively smaller number of proteins. On this benchmark EVE outperforms all methods including meta-predictors.

To put things in perspective with the other methods we introduced in § 3, we are also adding the family-agnostic ESM-1b and TranceptEVE. The former was shown to have good performance for human variants annotation in Brandes et al. [136] (similarly to § 4.2.3, we use the latest ESM-1b model from Rives et al. [40]

with extensions from Brandes et al. [136]), and the latter achieves state-of-the-art performance in terms of protein fitness prediction as per Table 4.2. TranceptEVE and EVE outperform all other baselines on the two benchmarks, with the former doing slightly better on each dimension.

As the consequence of variant classification significantly varies from one gene to another, an important feature of our method is the ability to assign a degree of uncertainty to the prediction, allowing a trade-off between predicted accuracy and coverage of variants. Setting aside an increasing number of variants as 'uncertain' enabled us to reach higher accuracy over the variants that we do classify as pathogenic or benign. For instance, excluding the 25% of most uncertain variants resulted in an accuracy of approximately 90% for pathogenic and benign classifications (Fig. 4.6b). In practice, we envision researchers deciding on specific trade-offs on a gene-by-gene and use case basis.

EVE is as accurate as experimental prediction

We evaluated whether our computational predictions are as accurate as experimental predictions. We found that, for five genes with a significant number of high-quality labels in ClinVar (BRCA1, TP53, PTEN, MSH2, and SCN5A), the performance of EVE in predicting clinical significance equals or surpasses that of deep mutational scan experiments designed to predict pathogenicity [156, 164–167] (Fig. 4.8 and Appendix B.2.2). In the case of TP53, EVE provided near-perfect separation of benign and pathogenic variants across the entire protein, outperforming experimental predictions [165] that are weaker in the tetramer domain (from position 300 to the end). In the case of the SCN5A gene, associated with Brugada syndrome [168] and long QT syndrome [169], EVE predicted the variant R814Q as pathogenic. Interestingly, this prediction contradicts the experimental findings from reference [164], which identified R814Q as a gain rather than a loss of function. This suggests that evolutionary data, like that used by EVE, contain information about gain of function and support the existing genetic knowledge [169]. EVE also has marginally better performance than experiments on a larger set of genes that have fewer

high-quality labels (Appendix B.2.2). As EVE and MAVEs are independent sources of evidence, comparison of their results may help to evaluate the clinical labels themselves. Across MSH2, PTEN and TP53, 23 of the 27 variants (85%) where the EVE score disagrees with ClinVar, MAVE experimental data support the EVE classification. Overall, our analysis demonstrates that the performance of EVE predictions is on par with those from high-throughput experiments. This suggests that allocating experimental resources to genes where the performance of EVE is lacking could be a beneficial strategy.

Predictions for 36 million variants

Our approach provides both the continuous EVE scores and classification assignments for 36 million individual amino acid variants across 3,219 disease-associated genes. Out of these variants, around 1.3 million have been observed in at least one human, as per UK Biobank [145] and gnomAD [148] (Appendix B.2.1). However, ClinVar only provides a clinical interpretation for roughly 3% of these variants (refer to Fig. 4.9a, left). Excluding the 25% most uncertain variants to maintain approximately 90% accuracy, EVE class assignments provide interpretation for about 27 million variants overall, and for over 800k (approximately 64%) of variants documented in humans to date (Fig. 4.9a, middle). The continuous EVE scores for all single amino acid variants provide a complementary picture to that of the discrete class assignments. The distribution of EVE scores within proteins reveals clusters of high pathogenicity, reflecting patterns expected by functional importance, such as hydrophobic cores, and ligand-binding and active sites. For instance, many variants in the SCN4A–SCN1B ion channel complex (PDB 6AGF [170]) with high EVE scores are located at the complex interface, the pore region of SCN4A, and the hydrophobic core of SCN1B (Fig. 4.9b, c). For the mismatch DNA repair complex MSH2–MSH6, strong EVE pathogenic signals are identified for variants located close to the bound ADP and DNA (PDB 2O8B [171]), where clinical labels are sparse but still observed in the population (Fig. 4.9d).

Combining EVE with other evidence

Since EVE provides a single source of evidence, it is easily combined with other orthogonal sources of evidence, as is typically performed by expert panels such as ClinGen [149]. To illustrate this, we combined our model class assignments with population data from gnomAD [148] and other forms of existing evidence (Fig. 4.9 and Appendix B.2.3). This process identified 256k variants with no prior clinical interpretation for potential reclassification, and another 539 variants that contradict the current ClinVar status, for which we found independent supporting evidence. Lastly, as a fully unsupervised method, EVE can use all existing labels for validation, offering another decisive advantage over supervised methods (Appendix B.2.4). This could be critical to a more refined approach in which the *strength of evidence* provided by the model may be allowed to vary on a gene-by-gene basis, in close analogy with recommendations for functional assays [172].

4.4 Predicting viral immune escape

4.4.1 Background and objectives

Viral diseases are characterized by a complex interplay between immune detection in the host and viral evasion, often leading to the evolution of viral antigenic proteins. Antibody escape mutations affect viral reinfection rates and the duration of vaccine efficacy. Consequently, predicting with sufficient lead time which viral variants are likely to avoid immune detection is crucial for creating effective vaccines and therapeutics. While experimental methods such as pseudovirus assays [173, 174] and deep mutational scans [175–190] can potentially anticipate viral immune evasion, their ability to predict immune escape early on is limited by their reliance on representative antibodies or sera, which are only available once a significant part of the population is infected or vaccinated. Furthermore, the rapid evolution of pandemic viruses makes it impractical to systematically test all emerging variants.

Building computational methods to predict viral escape is therefore of significant interest. An ideal model should be able to evaluate the likelihood of escape for unseen

variations across the full antigenic protein, guide the design of specific experiments, would be updated with pandemic information, and generate predictions early enough to influence vaccine development. However, existing computational approaches for forecasting viral fitness or immune escape heavily depend on real-time sequencing or pandemic antibody structures. This dependence hampers their ability to predict unseen variants and restricts their utility for vaccine development during the early stages of a pandemic [21, 35, 191–193].

In this section, we introduce EVEscape, a flexible framework that overcomes the limitations of current methods by combining a deep generative model trained on historical viral sequences, with structural and biophysical constraints. Unlike its predecessors, EVEscape doesn't rely on recent pandemic sequencing or antibodies, making it useful during the initial stages of a viral outbreak and for the continuous assessment of emerging strains. By leveraging functional constraints learned from past evolution, as we already thoroughly discussed in § 3.2, EVEscape can incorporate relevant epistasis [194] and thus predict mutant fitness within the context of any strain background. Furthermore, EVEscape is readily adaptable to new viruses, as we demonstrate in our validation on SARS-CoV-2, HIV, and Influenza. Our methodology allows for early detection of potentially harmful mutations, aiding in the creation of more potent vaccines and therapeutics. By serving as an early warning system, it can inform public health decisions and preparedness plans, ultimately reducing both the human and economic burdens of a pandemic.

4.4.2 Method

Viral proteins that escape immunity disrupt antibody binding while retaining protein expression, protein folding, host receptor binding, and other properties necessary for viral infection and transmission [177]. Building on that observation, we designed EVEscape to express the probability of a mutation to induce immune escape as the product of three probabilities; the likelihood that a mutation maintains viral fitness ('fitness' term), occurs in an antibody accessible region ('accessibility' term), and disrupts antibody binding ('dissimilarity' term) (Fig. 4.10a). Critically,

these three components solely rely on pre-pandemic data sources, allowing for early warning (Fig. 4.10b).

The first model component predicts fitness using the EVE Bayesian VAE trained on evolutionarily-related protein sequences as described in § 3.2.2. EVE considers dependencies across positions, capturing the changing effects of mutations as the dominant strain backgrounds diversify from the initial sequence [195–197]. We showcase the efficacy of EVE for modeling viral proteins by comparing model predictions and data from mutational scanning experiments that measure multiple facets of fitness for thousands of mutations to viral proteins [190, 197–201]. Model performance approaches the Spearman's rank correlation (ρ) between experimental replicates, including viral replication for influenza [198] ($\rho = 0.53$) and HIV [197] ($\rho = 0.48$) (Appendix B.3.2). For SARS-CoV-2, we trained EVE across broad pre-pandemic coronavirus sequences, from sarbecoviruses like SARS-CoV-1 to "common cold" seasonal coronaviruses like the Alphacoronavirus NL63, and compared predictions to measures of expression ($\rho = 0.45$) and receptor binding [190] ($\rho = 0.26$). We note that sites which express in the DMS experiments but are predicted deleterious by EVE are frequently in contact with non-assayed domains of the Spike protein or with the trimer interface – interactions not captured in the RBD yeast-display experiment.

The second model component, antibody accessibility, is motivated by the need to identify potential antibody binding sites without prior knowledge of B cell epitopes. Accessibility of each residue is computed from its negative weighted residue-contact number across available 3D conformations (without antibodies), which captures both protrusion from the core structure and conformational flexibility [202–205].

Finally, the third component is computed using differences in hydrophobicity and charge, properties known to impact protein-protein interactions [206–208]. This simple metric correlates with experimentally measured within-site escape more than individual chemical properties, BLOSUM substitution-matrix derived distance [209].

Each component is separately standardized and fed into a temperature-scaled logistic functions, and the product of the resulting three probabilities yield the overall EVEscape index (Appendix B.3.3).

4.4.3 Results

Anticipating pandemic variation with pre-pandemic data

The extensive surveillance sequencing and experimentation undertaken during the COVID-19 pandemic have provided a unique opportunity to evaluate the predictive capabilities of EVEscape in anticipating immune evasion before escape mutations are effectively observed [210, 211]. In order to assess the model's ability to make early predictions, we conducted a retrospective study using solely pre-pandemic information (training on Spike sequences across Coronaviridae that were available before January 2020). We then evaluated the method by comparing predictions against what was subsequently learned about SARS-CoV-2 Spike immune interactions and immune escape.

The top predicted escape mutations for the whole of Spike are strongly biased towards the receptor-binding domain (RBD) and N-terminal domain (NTD), coincident with the bias for antigenic regions seen in the pandemic [213, 214] (Fig. 4.11a-b). Within these domains, EVEscape scores are biased towards neutralizing regions—the receptor-binding motif of the RBD and the neutralizing supersite [212] in the NTD (Fig. 4.11c). EVEscape's ability to identify the most immunogenic domains of viral proteins without knowledge of specific antibodies or their epitopes could provide crucial information for early development of vaccines in an emerging pandemic [215]. We next compare model predictions to mutations that were subsequently observed in the pandemic as deposited in GISAID (Global Initiative on Sharing All Influenza Data) [210], which contains over 500k unique sequences with over 12k missense mutations to Spike. For this analysis we focus on the RBD of Spike as this domain has been the most extensively studied due to its immunodominance [213, 214].

About half of our top RBD predictions were seen in the pandemic by December 2022 (Fig. 4.12a; this proportion is robust to the threshold defining top escape

mutations). The more often a mutation occurred in the pandemic, the more likely it is to be predicted by our method — 57% of high frequency observed substitutions are in the top EVEscape predictions (Fig. 4.12b-c). We expect that the highest frequency mutations, seen in historical Variants of Concern (VOCs), will be enriched for escape variants that provide a fitness advantage in an immune population (whilst not expecting that all single substitutions in the VOCs will contribute to escape).

EVEscape's immune-specific components reflect important pandemic constraints and allow for mutation interpretability. For instance, VOC mutations R190S and R408S, with high EVEscape but low EVE scores, are in hydrophobic pockets that may facilitate significant immune escape [216] (Fig. B.6). Meanwhile, the few VOC mutations (i.e., A222V and T547K) with significant EVE—but not EVEscape—scores have functional improvements such as monomer packing and RBD opening but do not impact escape [217, 218]. We also see that the proportion of EVEscape predictions seen during the pandemic increased over time – from 3% in December 2020 to 49% in December 2022 (Fig. 4.12a) – and should continue to increase, an expected trend both as more variants are observed and as adaptive immune pressure increases [219] with the growing vaccinated or previously infected population.

Our model successfully predicted escape mutations that were later observed during the pandemic within the epitopes of prominent therapeutic monoclonal antibodies authorized for Emergency Use [211] (e.g., N440, E484A/K/Q, Q493R). These predictions highlight the interplay of our three model components; for instance, the high accessibility as well as mutability of E484 results in 50% of all possible mutations at this site in the top 2% of EVEscape predictions and includes E484A/K in the top 1% – notable for escape from bamlanivimab [220] (Fig. 4.12d) – because of their high dissimilarity scores. We also identify candidate escape mutations in these therapeutic epitopes that have not yet been observed at frequencies higher than 10k – for instance variants to K444 and K417, a subset of which are beginning to appear. This result suggests that escape sites can be well predicted before

a pandemic and may have concrete applications for escape-resistant therapeutic design and early warning of waning effectiveness.

EVEscape represents a significant improvement over past computational methods. EVEscape is more than twice as predictive as prior unsupervised models [221], both at predicting pandemic mutations (49% vs. 24% of top predictions observed in pandemic and 57% vs. 9% of highest frequency mutations predicted) as well as experimental measures of antibody escape (0.53 vs. 0.24 AUPRC) (Fig. 4.12a,b,e and Fig. B.7). All EVEscape components play a role in these predictions, with fitness predictions and accessibility metrics identifying sites of escape mutations while dissimilarity identifies amino acids that facilitate escape within sites (Fig B.8). Moreover, other computational methods [191, 193] focus on near term prediction of strain dominance rather than longer term anticipation of immune evasion as they rely on pandemic sequences, antibody-bound Spike structures, or both, thereby hindering the ability to assess early predictive capacity. It is therefore notable that EVEscape outperforms even supervised approaches at predicting mutations seen in the pandemic (Fig B.9).

Adapting EVEscape to reflect pandemic characteristics through its modular framework

The modular design of our framework facilitates its adaptability to the specific characteristics of a pandemic and to new data as it becomes available. To consider the effects of insertions and deletions on SARS-CoV-2 Spike immune escape [222], we replace the EVE fitness component with TranceptEVE (§ 3.4.4) – which as we have seen in § 4.2.3 achieves state-of-the-art fitness prediction performance on both substitutions and indels. The latter is particularly compelling as prior computational models and high-throughput experiments have been unable to capture indels for SARS-CoV-2. When applied to the pandemic, this model captures the most frequent single insertion and deletion, both at site 144, each in the top decile of pandemic and random indel predictions (Fig.B.10). Lastly, we retrain EVE models with the addition of 11 million new sequences collected during the pandemic, which helps improve agreement with fitness DMS experiments by 20% – confirming that our

model is not only helpful early in pandemic but that its performance continues to increase over time as more data are collected.

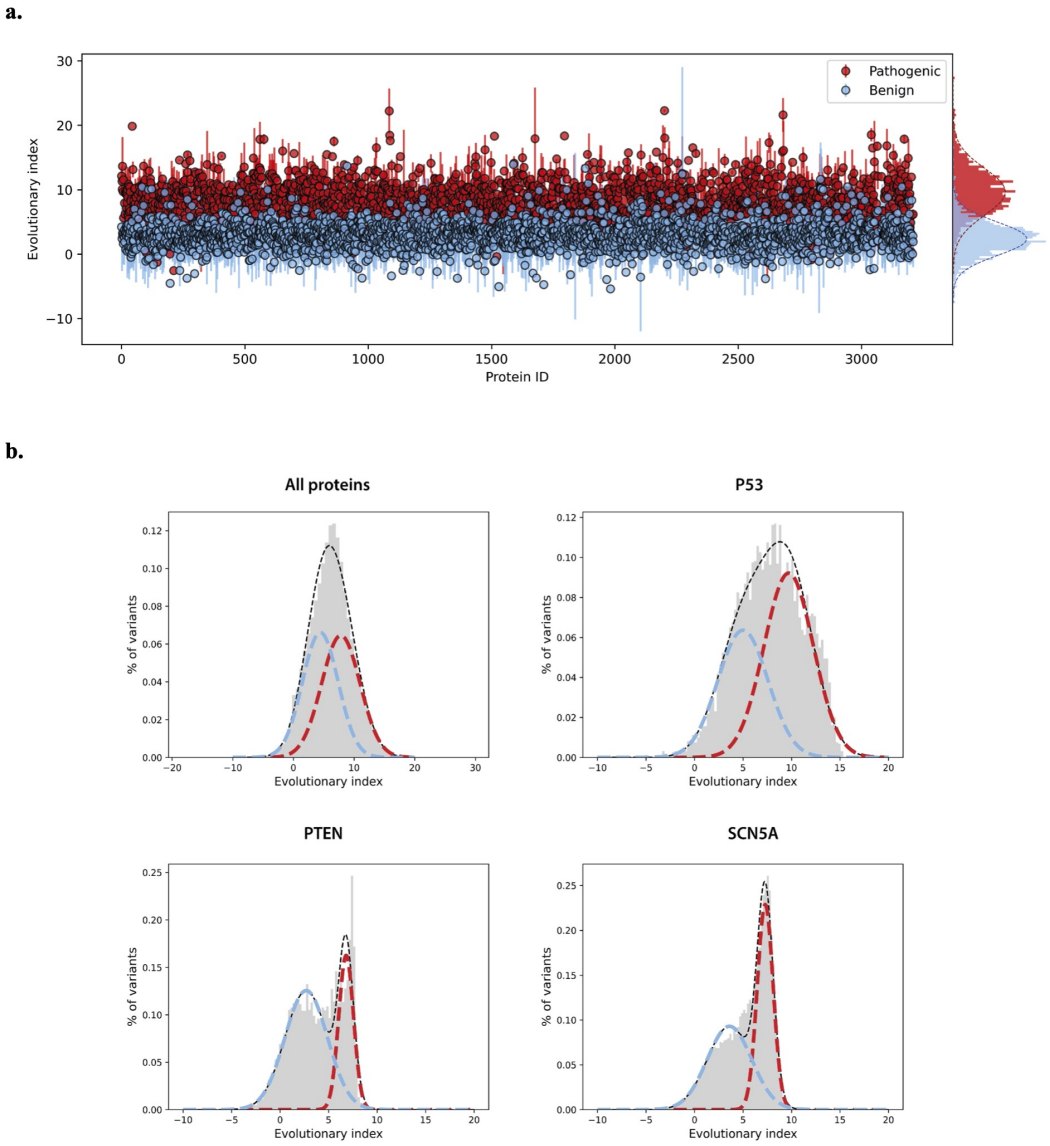

Figure 4.5: Evolutionary index separates pathogenic and benign variants **a.** Average evolutionary index per protein, and corresponding standard deviations, for variants with known Benign and Pathogenic ClinVar labels across 3,219 proteins (sorted by alphabetical order). On the right, marginal distributions of the means over the 3,219 proteins. Evolutionary index separates pathogenic and benign labels consistently across proteins. **b.** Two-component Gaussian Mixture Models (GMM) over the distributions of the evolutionary indices (histograms) for all the single amino acid variants of 3,219 proteins combined (top, left) and for P53, PTEN and SCN5A separately (top right, bottom left and right, respectively). The dashed black line is the marginal likelihood for the GMM model, i.e. the likelihood of a variant sequence after marginalizing the latent variable that corresponds to the mixture assignment; the dashed blue and red lines represent the relative share of the marginal likelihood from the benign and pathogenic clusters respectively (i.e., the product of the marginal likelihood by each cluster).

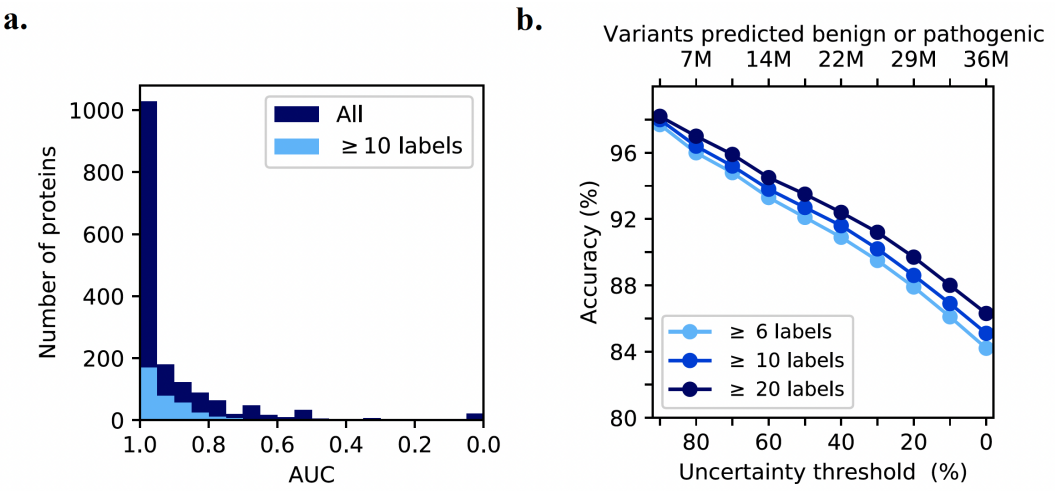

Figure 4.6: EVE – Performance on clinical labels & uncertainty a. Distribution of the AUC for EVE scores computed over known clinical labels from ClinVar, on all 3,219 proteins covered by our study (dark blue) and for the subset of proteins with at least five benign and five pathogenic known labels (pale blue). **b.** Trade-off between the accuracy of EVE and the uncertainty threshold (percentage of variants set as 'uncertain') or the total number of variants given a class assignment. Accuracy was computed over all labels for proteins with at least three (five or ten) benign labels and three (five or ten) pathogenic labels.

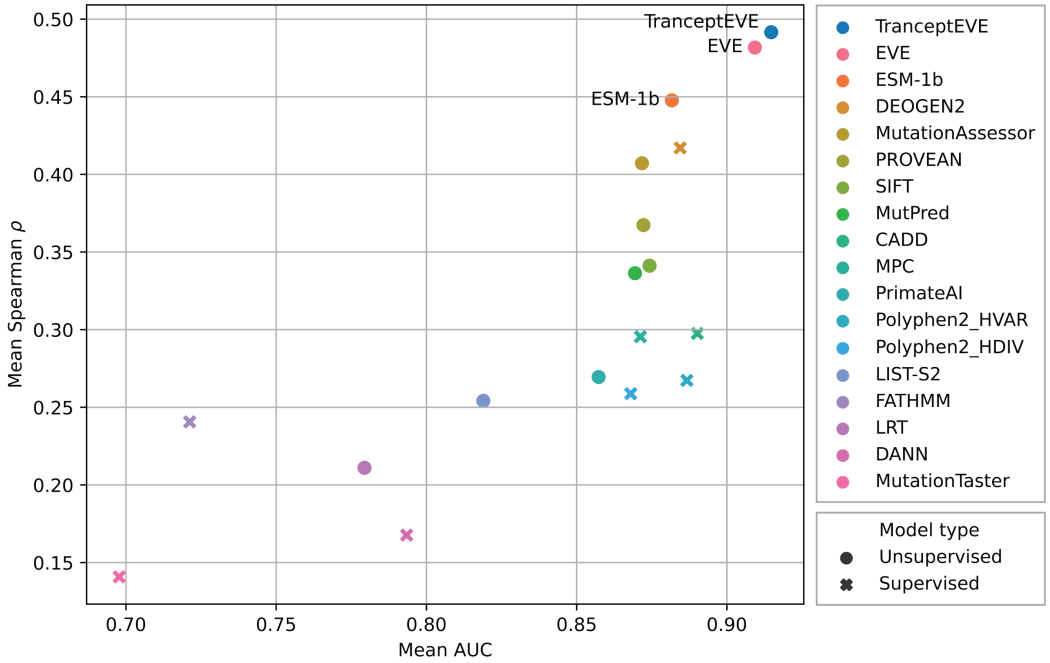

Figure 4.7: Performance of EVE and TranceptEVE compared to state-of-the-art computational variant effect predictors: eight unsupervised and eight supervised. Performance estimated against known clinical labels (avg. AUC over disease genes in ClinVar (x axis)), and high-throughput functional assays developed to assess the clinical effect of variants (avg. Spearman's rank correlation (y axis)).

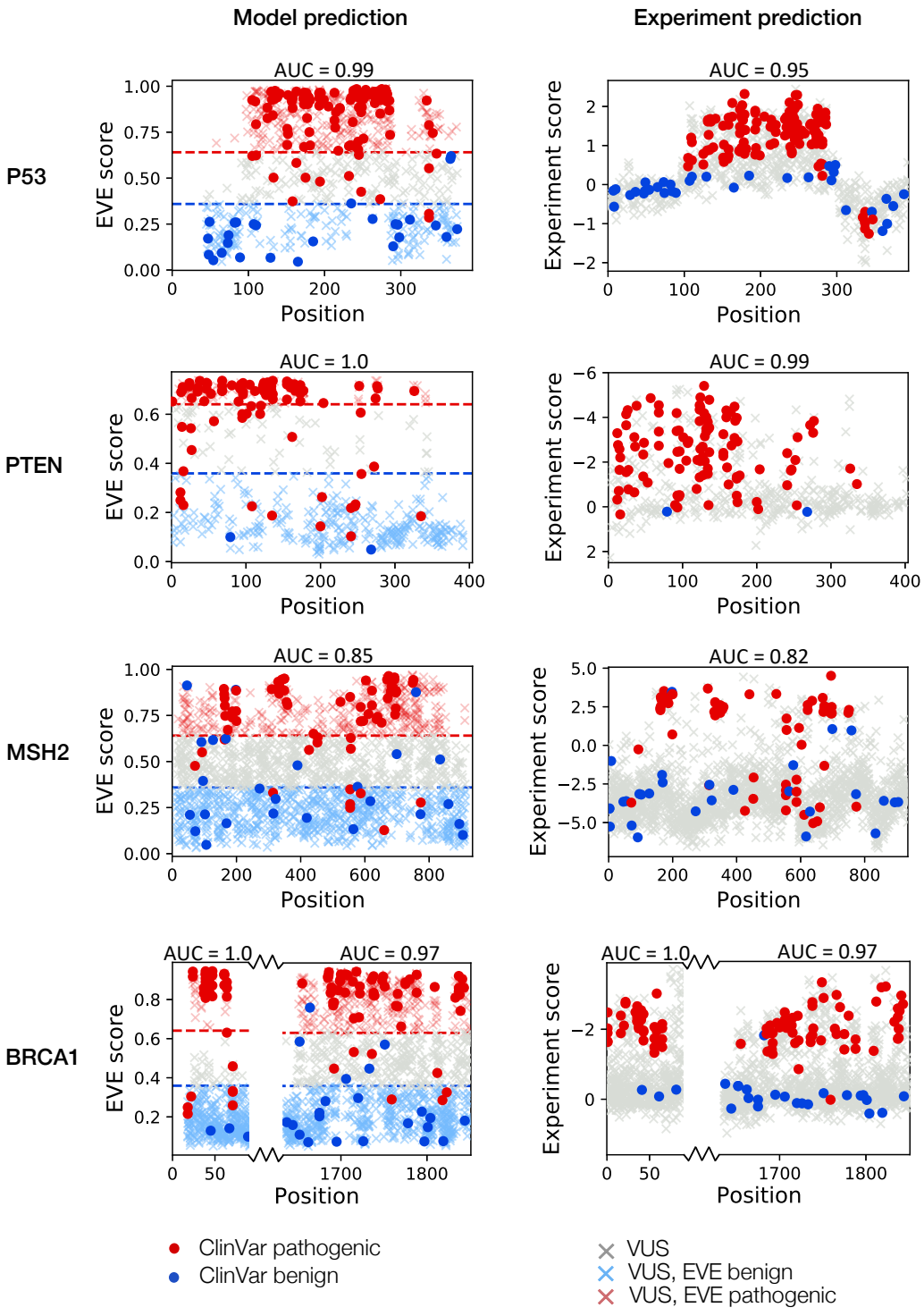

Figure 4.8: EVE is as good as functional experiments at predicting clinical interpretations of variants. Comparison of computational model predictions (left panels, y axis EVE score) and experimental predictions (right panels, y axis experimental score) to ClinVar labels (dots) and variants of unknown significance (VUS) (crosses), where pale red and pale blue crosses indicate EVE predictions; the x axis corresponds to the position in the protein. The dashed red and blue lines correspond to EVE predictions setting the 25% most uncertain assignments as uncertain (as described in § 4.3.2)

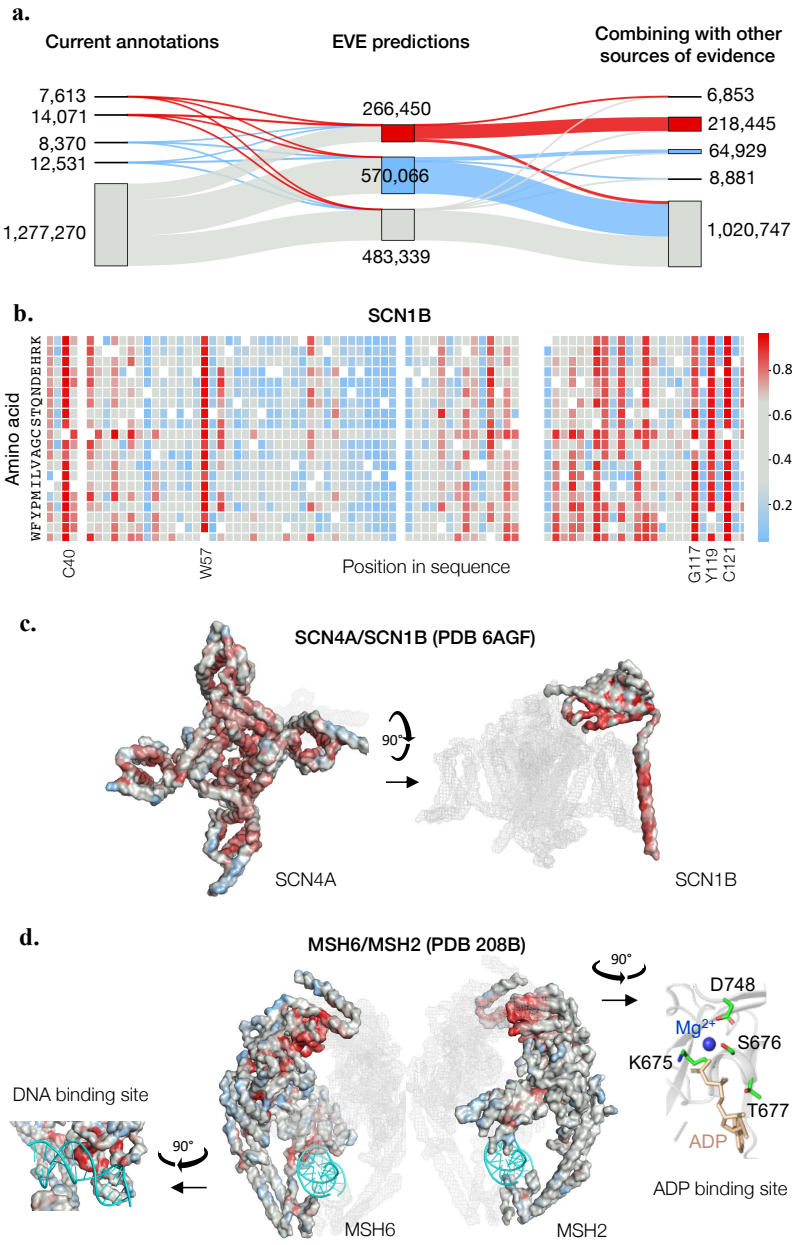

Figure 4.9: EVE predictions for variants in 3,219 genes. a. Combining EVE classifications with other sources of evidence. ClinVar labels and VUS based on gnomAD and UK Biobank (left); EVE predictions setting 25% of all possible variants as uncertain (middle); and predictions after combining EVE with other sources of evidence (right) (Appendix B.2.3). **b.** Heat map of EVE pathogenicity scores in SCN1B. **c & d.** Representations of 3D structures of SCN4A–SCN1B (PDB 6AGF [170]) (c) and MSH6–MSH2 bound to ADP and a G T mispair (PDB 2O8B [171]) (d), coloured by mean score per position (SCN4A, MSH6 and MSH2) and maximum score per position (SCN1B). Clusters of high pathogenicity in 3D include the pore region of SCN4A, the hydrophobic core of SCN1B (positions 40, 57, 117, 119 and 121), the C terminus α-helix of SCN1B and the interface with SCN4A, the ADP-binding site of MSH2 (such as D748N/V/H, K675E, S676L and T677R) and the DNA binding site of MSH6.

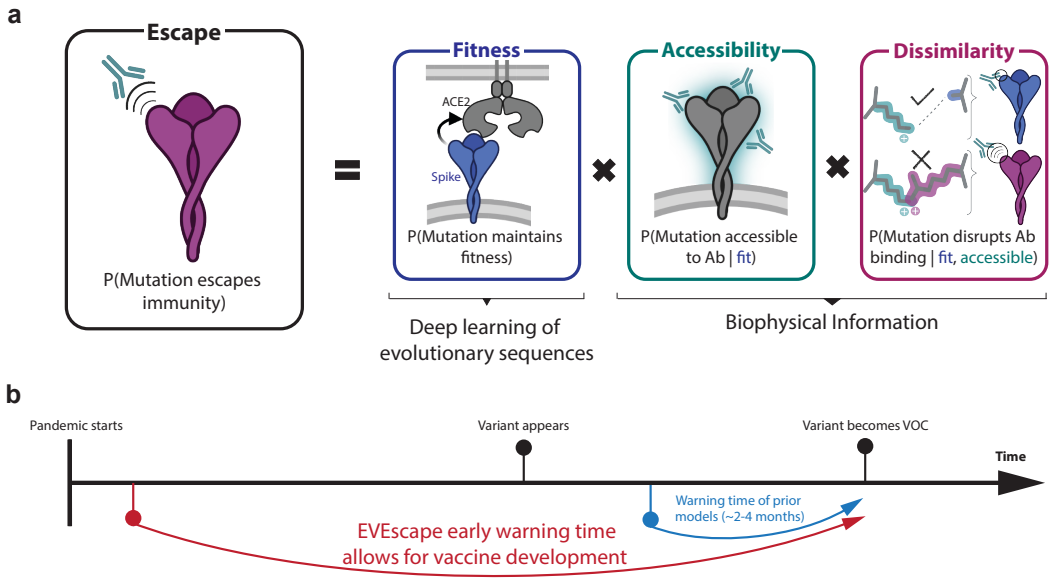

Figure 4.10: Early prediction of antibody escape from deep generative sequence models, structural and biophysical constraints. EVEscape assesses the likelihood of a mutation to escape the immune response based on the probabilities of a given mutation to maintain viral fitness, to occur in an antibody epitope, and to disrupt antibody binding. It only requires information available early in a pandemic, before surveillance sequencing, antibody-antigen structures or experimental mutational scans are broadly available.

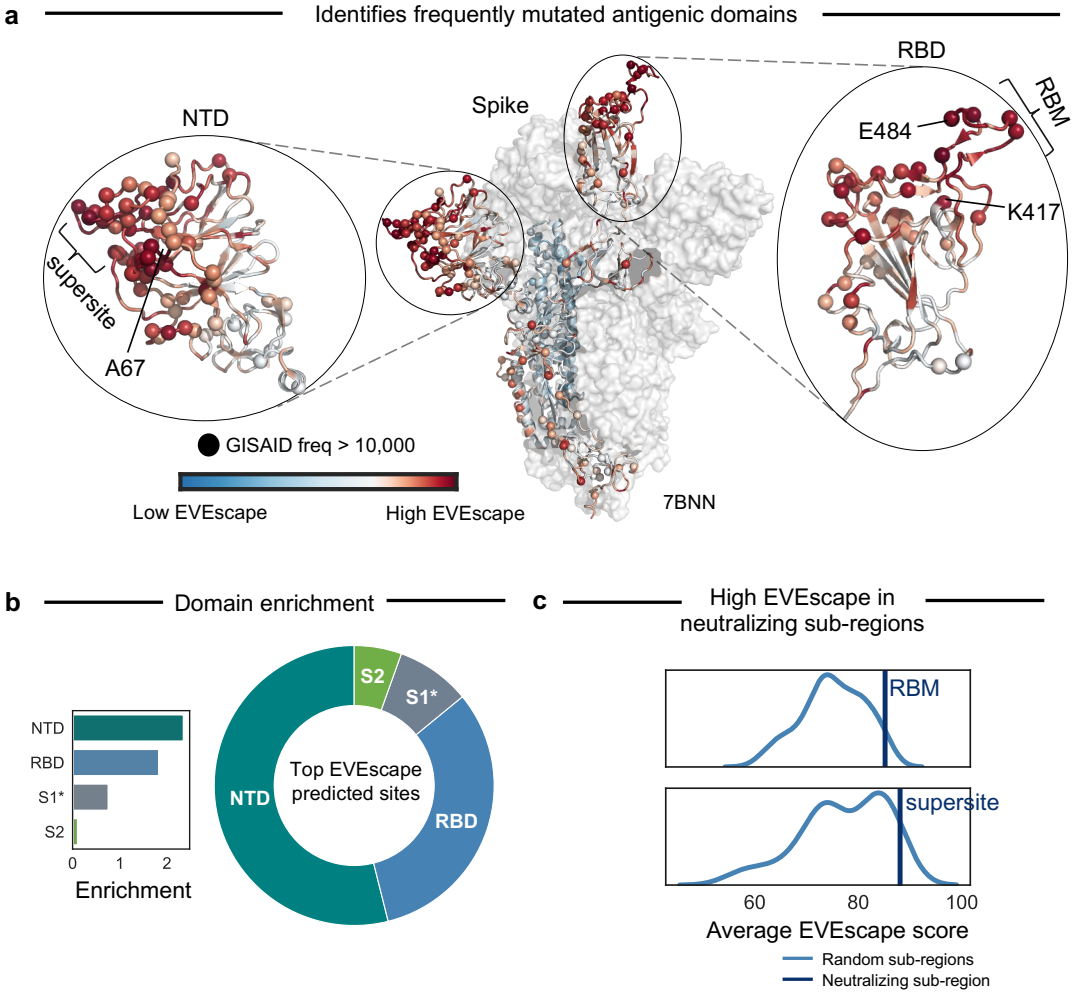

Figure 4.11: EVEscape identifies antigenic regions without antibody information. a. EVEscape scores mapped onto a representative Spike 3D structure (PDB identifier: 7BNN) highlight high-scoring regions with many observed pandemic variants, both in the RBD (receptor-binding domain) and NTD (N-terminal domain). Spheres indicate sites with mutations observed more than 10,000 times in the GISAID sequence database. **b.** The top decile of EVEscape predictions span diverse epitope regions across Spike, but the majority of predictions are in the NTD and RBD, which have a disproportionately high number of predicted EVEscape sites relative to their sequence length (enrichment). The regions considered are NTD (sequence positions 14 - 306), RBD (319 - 542), S1* (543 - 685), and S2 (686 - 1273), where S1* refers to the region in S1 between RBD and the S2. **c.** Neutralizing sub-regions – RBM (receptor-binding motif, 438-506) and NTD supersite [212] (14-20,140-158, 245-263) – have significantly higher than average EVEscape scores, relative to a distribution of 150 random contiguous regions of the same length within the RBD and NTD, respectively.

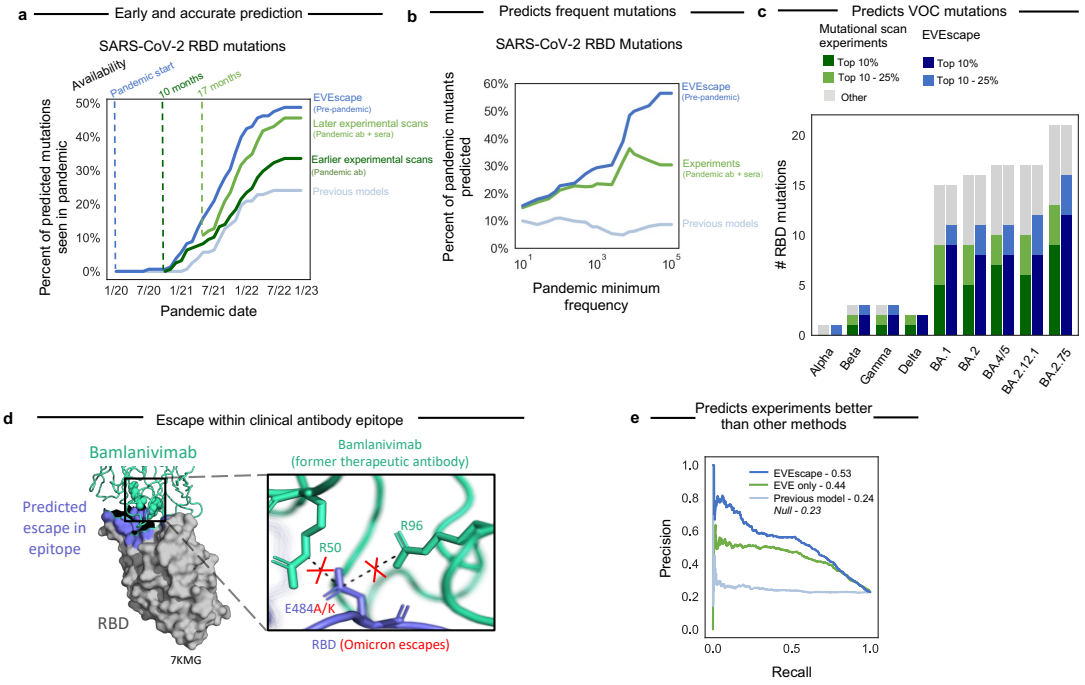

Figure 4.12: Pre-pandemic EVEscape is as accurate as intra-pandemic experimental scans at anticipating pandemic variation: retrospective analysis. **a.** Percent of top decile predicted escape mutations by EVEscape, mutational scan experiments, and a previous computational model [221] seen over 100 times in GISAID by each date since the start of the pandemic. EVEscape based on pre-pandemic sequences anticipates pandemic variation at least on par with mutational scan experiments based on antibodies and sera available 10 or 17 months into the pandemic. Analysis focuses only on nonsynonymous point mutations that are a single nucleotide distance away from the Wuhan viral sequence. RBD is the receptor-binding domain of the Spike protein. **b.** Percent of observed pandemic mutations in top decile of escape predictions by observed frequency during the pandemic. High-frequency mutations in particular are well-captured by EVEscape. **c.** The majority of RBD mutations observed in VOC strains have high EVEscape scores and somewhat lower scores in the mutational scan experiments against pandemic sera. **d.** EVEscape can predict escape mutations in the epitope of the former therapeutic antibody bamlanivimab. E484 is involved in a salt bridge with R96 and R50 of bamlanivimab, which lost FDA Emergency Use Authorization due to Omicron's emergence, wherein E484A or E484K mutations (both predicted in the top 1% of EVEscape Spike predictions) escape binding due to the loss of these salt bridges[220]. **e.** Precision-recall curve of RBD escape predictions of EVEscape, EVEscape fitness component only (EVE model) and previous computational model [221] when compared to DMS escape mutations (AUPRC reported with a comparison to a "null" model where escape mutations are randomly predicted).

5 Design

Contents

5.1 Introduction

In this chapter, we delve into various generative models that support the design of novel biomolecules. We will use drug discovery as a connecting thread for the different sections of this chapter, but we note that the majority of the methods discussed could be used as is for the design of molecules in several other application domains (e.g., new material design, sustainability).

The pharmaceutical industry currently faces considerable challenges, including high failure rates, lengthy development timelines, and escalating costs. Integrating machine learning guided decisions into the drug discovery process has the potential to address these issues by significantly enhancing efficiency, reducing costs, and accelerating the timeline from target identification to marketable therapeutics. The examples we cover in this section cover three pivotal points of that development pipeline: gene target identification (§ 5.2), de novo molecular optimization (§ 5.3), and protein design (§ 5.4).

In the first section (§ 5.2), we introduce DiscoBAX, a novel method derived from the Bayesian optimization literature (and more specifically from BAX [223]) which seeks to strike a balance between high property of the target objective and diversity of the collected objects across iterative experiment-in-the-loop cycles. We illustrate the benefits of the method in a gene target identification, an early step in the standard drug development pipeline, but the method is general and can be applied to a wide range of use cases. Furthermore, while the method discussed in this first section does not strictly require the use of generative models it may, as we will see in the following section, be leveraged in conjunction with generative models, for instance to obtain compact representations of the underlying biological objects.

In the second section (§ 5.3), de novo molecular optimization, we examine the task of generating novel small molecules that optimize an expensive black-box objective. We build on the pioneering work from Gómez-Bombarelli et al. [17] which first recast the initial discrete optimization task in the original molecular space, into a continuous optimization task in the latent space of a variational autoencoder. This has many benefits that we briefly review a the beginning of the corresponding section, but also comes with certain challenges – one of them being that optimization may explore regions of latent space far from the VAE training data, which often results in poor output from the decoder of the VAE. Our method proposes to leverage the uncertainty of the decoder network to guide the optimization in latent space and avoid regions that are likely to lead to poor decodings.

In the third and last section (§ 5.4), we focus on the engineering of novel proteins. We describe how generative models, including the models we introduced in § 3 may be used to guide iterative protein redesign cycles. This process encompasses the optimization of naturally occurring sequences for improved properties (e.g., stability, binding affinity, specificity) which is a critical step in the development of biologics and protein-based therapeutics.

5.2 Gene target identification

5.2.1 Background and objectives

Genomic experiments that investigate gene function in realistic cellular conditions are fundamental to modern early-stage drug target discovery and validation, as they can be used to identify effective modulators of one or more disease-relevant cellular processes. These experiments, for example using Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR) [224] perturbations, are both time and resource-intensive [225–228]. Therefore, an exhaustive search of the billions of potential experimental protocols covering all possible experimental conditions, cell states, cell types, and perturbations [229–233] is infeasible even for the world's largest biomedical research institutes.

To mitigate the chances of failure in subsequent stages of the drug design pipeline, it is desirable for the subset of precursors selected in the target identification stage to operate on *diverse* underlying biological mechanisms [234]. That way, if a promising candidate based on in-vitro experiments triggers unexpected issues when tested in-vivo (e.g., undesirable side effects), other lead precursors relying on different pathways might be suitable replacements that are not subject to the same issues. This two-phase maximization problem diverges from standard formulations of Bayesian optimization or active learning. In particular, the noisy measurements obtained by the experimenter do not correspond to the objective of interest, but are only correlated with this outcome via some unknown mechanism. Thus even in the limit of infinite intermediate phenotype measurements, it is not possible to identify the maximum of the objective function.

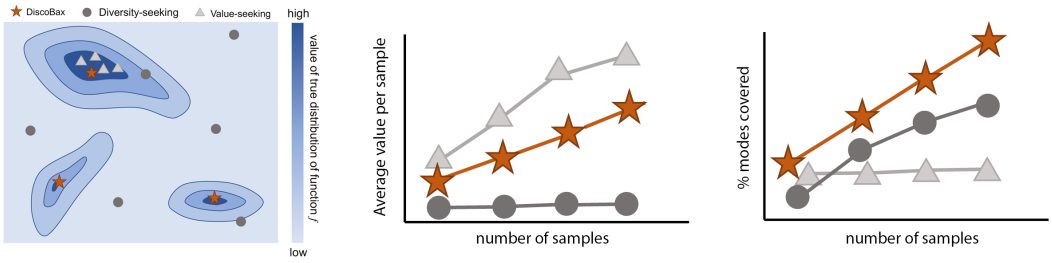

Figure 5.1: We compare DiscoBAX (orange star) to existing diversity-seeking (dark grey circle) and value-seeking (light grey triangle) batch active learning policies. DiscoBAX aims to recover a maximally diverse set of interventions with values above a pre-defined threshold from a given underlying distribution. This aim contrasts with value-seeking strategies focusing on maximizing value and diversity-seeking strategies focusing on maximizing coverage. We expect DiscoBAX to design genomic experiments yielding high value findings that maximize mode coverage. As discussed in § 5.2.1, the diversity of selected interventions is highly desirable to increase the chances that at least some of these interventions will succeed in subsequent stages of the drug discovery pipeline.

Mathematically, finding a diverse set of precursors corresponds to identifying and sampling from the different modes of the black-box objective function mapping intervention representations to the corresponding effects on the disease phenotype. Existing machine learning methods for iterative experimental design (e.g., active learning, Bayesian optimization) have the potential to aid in efficiently exploring this vast biological intervention space. However, to our knowledge, there is no existing method geared toward identifying the modes of the underlying black-box objective function to identify candidate interventions that are both effective and diverse. For instance, Bayesian optimization is concerned with finding the global optimum of a function with the fewest number of function evaluations [235, 236] and is not concerned with promoting diversity in the queried points. Active learning typically seeks to optimize the model learning process by selectively querying the most informative instances from a pool of unlabeled data points – this tends to result in selecting diverse instances, but there is no bias toward points that maximize the target function.

To this end, we introduce DiscoBAX (§ 5.2.3) - a sample-efficient BAX method (see next paragraph) for discovering genomic intervention sets with both high expected change in the target phenotype and high diversity to maximize chances of success in the following stages of drug development, which we conceptually

illustrate in Fig. 5.1. We demonstrate the benefits of the method across a diverse set of CRISPR experimental assays – the GeneDisco benchmark – which we introduce in § 5.2.2.

Bayesian Algorithm Execution (BAX). BAX [223] is a method to estimate the output, $O_{\mathcal{A}} \coloneqq O_{\mathcal{A}}(f)$, of an algorithm, $\mathcal{A}$, run on a function, f, by having access to function values $\{\mathrm{y}_{\mathbf{x}_i}\}_{i=1}^{T} \in \mathcal{X}$ for a budgeted set of input points, $\mathcal{D}_t$. BAX acquisition functions select points by maximizing the expected information gain (EIG) obtained from each point about the output of the algorithm. Crucial to the applicability of BAX is the tractability of accurate approximators of the EIG for algorithms which, like the one we will propose, return a subset S of their inputs. The exact computation of the EIG for arbitrary algorithms is not generally tractable; however, Neiswanger et al. [223] present an approximation that only requires the computation of the entropy of the distribution over function values conditioned on algorithm outputs.

$$\mathrm{EIG}_t(\mathbf{x}, \mathcal{D}_t) = H(f(\mathbf{x})|\mathcal{D}_t) - \mathbb{E}_{p(S|\mathcal{D}_t)}[H(f(\mathbf{x})|S, \mathcal{D}_t)]. \tag{5.1}$$

When the model P is a Gaussian Process (GP) both of these quantities are straightforward to compute: the first is the entropy of the GP's predictive distribution at $\mathbf{x}$, and we can estimate the second by conditioning a posterior on the values of elements in the set S. Monte Carlo approximation of this quantity is possible when the model P does not permit a closed form.

5.2.2 GeneDisco benchmark

The GeneDisco benchmark curates and standardizes two types of datasets: three standardized feature sets providing representations for interventions of interest (e.g., gene knockouts), and five in vitro genome-wide CRISPR experimental assays, each measuring an outcome of interest following a possible intervention. We describe these assays in Appendix C.1.1. Additional details about the benchmark are available in our GeneDisco paper [237].

5.2.3 Method

A significant obstacle in the drug discovery process is the inconsistency between the results of *in vitro* experiments and the outcomes observed *in vivo*. For example, in vitro experimental data can measure the impact of a gene knockout on a particular aspect of a cellular phenotype within a controlled environment, such as a petri dish, while in vivo interactions between the drug and the organism might result in reduced effect sizes or toxicity, deviating from the in vitro observations. The drug discovery pipeline involves a series of stages that begin with evaluating a broader set of candidate interventions and selecting a subset of promising candidates for further development. This subset needs first and foremost to impact the disease phenotype of interest. But it is also desirable to consider a *diverse* set of candidate interventions targeting different mechanisms within the disease phenotype, to increase the likelihood of at least one candidate succeeding in subsequent phases.

We formalize this problem as an optimization problem where the optimizer has access to a *noisy* measurements correlated with the quantity of interest. We define our search space (e.g., the set of human genes) $\mathcal{G} = \{\mathrm{g}_1, \ldots, \mathrm{g}_m\}$ and the function f_{ip} that operates a mapping *intervention representations* (or features) to phenotype changes. The subscript 'ip' stands for *intermediate phenotype* to reinforce the fact that it is not the actual clinical measurement caused by the gene knockout, but rather a measurement known to correlate with the disease pathology and is tractable in the lab setting. We assume the phenotype change is a real number $f_{\mathrm{ip}}(\mathbf{x}) \in \mathbb{R}$; however, given suitable modeling assumptions, it is possible to extend our approach to vector-valued phenotype readouts.

We also define a function called *disease outcome*, f_{out}, which is composed of f_{ip} and factors outside the biological pathway, such as toxicity of a molecule that engages with a target gene. The noise component, η, encapsulates all these extra factors.

We seek out a *set* of interventions $S \subset \mathcal{G}$ of some fixed size $|S| = k$ whose elements cause the maximum expected change (for some noise distribution) in the disease outcome and high diversity. The latter will help ensure the associated toxicity or unintended side effects will be minimally correlated, thereby maximizing

the odds that at least one will succeed in subsequent steps of the drug discovery pipeline. We thus obtain a set-valued maximization problem:

$$\max_{S \subseteq \mathcal{X}} \mathbb{E}_\eta \left[\max_{\mathbf{x} \in S} f_{\text{out}}(\mathbf{x}; \eta) \right]. \tag{5.2}$$

The general formulation of this problem is NP-hard [238]; therefore, we propose a tractable algorithm that provides a constant-factor approximation of the optimal solution by leveraging the *submodular* structure of the objective under suitable modeling assumptions. Given such an algorithm, our task is the active learning problem of optimally querying the function, f_{ip}, given a limited number of trials, T, to accurately estimate the algorithm's output on the ground-truth dataset. Importantly, this formulation allows us to decouple modeling the measured phenotype, f_{ip}, from modeling the noise η. For example, we might make the modeling assumption that we sample f_{ip} from a GP with some kernel k_1 and that η is a Bernoulli random variable indicating the safety of the compound.

Various methods exist for efficiently optimizing black-box functions; however, our problem setting violates several assumptions underlying these approaches. In particular, while we assume access to intermediate readouts f_{ip}, the actual optimization target of interest f_{out} is not observable. Further, we seek to find a *set* of interventions that maximize its expected value under some modeling assumptions. These two properties render a broad range of prior art (§ 2.4) inapplicable. Active sampling methods do not prioritize high-value regions of the input space. Bayesian optimization methods assume access to the ground-truth function outputs (or a noisy observation thereof). And Bayesian algorithm execution approaches based on level-set sampling may not sufficiently decorrelate the hidden noise in the outcome.

Algorithm

We propose an intervention set selection algorithm in a Bayesian algorithm execution procedure that leverages the modeling assumptions we characterize in the previous section. Our method, Subset Discovery via Bayesian Algorithm Execution (DiscoBAX), consists of two distinct parts. (1) a subset-selection algorithm obtaining a

$1 - 1/e$-factor approximation of the set that optimizes (5.2), and (2) an outer BAX loop that queries the phenotype readings to maximize the information gain about the output of this algorithm. We next present the idealized form of DiscoBAX and show that it attains an approximately optimal solution.

Subset maximization. We first address the problem of identifying a subset $S \subset \mathcal{X}$ which maximizes the value $\mathbb{E}_\eta[\max_{\mathbf{x}\in S} f_{\text{out}}(\mathbf{x}; \eta)]$. As mentioned previously, the exact maximization of this objective is intractable. To construct a tractable approximation, we propose a submodular surrogate objective, under which the value of an intervention is lower-bounded by zero $f^*_{\text{out}}(\mathbf{x}; \eta) = \max(f_{\text{out}}(\mathbf{x}; \eta), 0)$. This choice is motivated by the intuition that any intervention with a negative expected value on the phenotype is equally useless as it will not be considered in later experiment iterations, and so we do not need to distinguish between harmful interventions. The resulting function $f(S) = \mathbb{E}_\eta[\max_{\mathbf{x}\in S} f^*_{\text{out}}(\mathbf{x}; \eta)]$ will be submodular, and thus Algorithm 3, the greedy algorithm, will provide a $1 - 1/e$ approximation of the optimal solution.

Observation 1. *The score function $f : \mathcal{P}(\mathcal{G}) \to \mathbb{R}$ defined by*

$$f(S) = \mathbb{E}_\eta\left[\max_{\mathbf{x}\in S}\left(\max(0, f_{out}(\mathbf{x}; \eta)\right)\right] \tag{5.3}$$

is submodular.

We provide proof of this result in Appendix C.1.2. In practice, we can estimate the expected value in this objective using Monte Carlo (MC) samples over the noise distribution η. Where MC sampling is too expensive, a heuristic that uses a threshold to remove points whose values under η are too highly correlated can also obtain comparable results with a reduced computational burden.

Algorithm 3 SubsetSelect

Require: integer $k > 0$, set $\mathcal{X}$, distribution $P(\eta)$, sampled $\widehat{f}_{\text{ip}} : \mathcal{X} \to \mathbb{R}$
  $S \leftarrow \emptyset$
  $\widehat{f}_{\text{out}}(\mathbf{x}; \eta) := \widehat{f}_{\text{ip}}(\mathbf{x})\eta(\mathbf{x})$
  for $i < k$ **do**

$$S \leftarrow S \cup \{\arg\max_{x \in \mathcal{X} \setminus S} \mathbb{E}_{\eta}[\max_{y \in S \cup \{x\}} \widehat{f}_{\text{out}}(\mathbf{x}; \eta)]\}$$

  end for
Output: S

Algorithm 4 DiscoBAX

Require: finite sample set $\mathcal{X}$, budget T, Monte Carlo parameter $\ell \in \mathbb{N}$
  $\mathcal{D} \leftarrow \emptyset$
  for $i < T$ **do**
    sample $\{\widehat{f}_{\text{ip}}\}_{j=1}^{\ell} \sim P(f_{\text{ip}}|\mathcal{D})$
    $S_j \leftarrow \text{SubsetSelect}(\widehat{f_{\text{ip},j}}), \forall j = 1, \ldots, \ell$
    $\mathbf{x}_i \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} \text{EIG}^{v}(\mathbf{x}, S_{j=1}^{\ell})$
    query $f_{\text{ip}}(\mathbf{x}_i)$
    $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{x}_i, f_{\text{ip}}(\mathbf{x}_i)\}$
  end for
Output: $\mathcal{D}$

Active sampling. Because we do not assume prior knowledge of the phenotype function f_{ip}, we require a means of selecting potential interventions for querying its value at a specified input $\mathbf{x}$. In practice, running these experiments may incur a cost, and so it is desirable to minimize the number of queries necessary to obtain an accurate estimate of the optimal intervention set. BAX [223] presents an effective active sampling approach to approximate the output of an algorithm using a minimal number of queries to the dataset of interest. In our setting, this allows us to approximate the output of Algorithm 3 over the set $(\mathcal{X}, f_{\text{ip}}(\mathcal{X}))$ without incurring the cost of evaluating the effect of every knockout intervention in $\mathcal{G}$. Concretely, this procedure takes as input some probabilistic model P which defines a distribution over phenotype readings f_{ip} conditioned on the data $\mathcal{D}_t$ seen so far and from which it is possible to draw samples. In the batch acquisition setting, we form batches of size B at each cycle by selecting the B points with the highest EIG values.

We note that subset selection is a function called within each active sampling cycle. Hence, the above observation about submodularity refers specifically to Algorithm 3 rather than its incorporation in Algorithm 4. If sample efficiency is not a concern this algorithm could be run on the set of all inputs and provide the exact solution. Also note we use multiplicative noise in Algorithm 3, but the algorithm directly extends to other noise structures (e.g., additive noise).

Practical implementation in high dimensions

Our approach is easily adaptable to incorporate approximate posterior sampling methods, enabling its use with deep neural networks on high-dimensional datasets. In that setting, we typically leverage Bayesian Neural Networks in lieu of Gaussian Processes. We sample from the parameter distribution via Monte Carlo dropout (MCD) [239], and rely on Monte Carlo simulation to estimate the quantities introduced in Algorithm 4. In particular, the entropy of the posterior distribution is obtained as follows:

$$\begin{aligned} H(\mathrm{y}_{\mathbf{x}}|\mathcal{D}_t) &= \mathbb{E}_{p(\mathrm{y}_{\mathbf{x}}|\mathcal{D}_t)}\left[-\log p(\mathrm{y}_{\mathbf{x}}|\mathcal{D}_t)\right] \\ &\sim -\frac{1}{M}\sum_{s=1}^{M} \log\ p(y_x^s|\mathcal{D}_t, f_s) \end{aligned} \tag{5.4}$$

where the samples $\{y_x^s = f_s(x)\}_{i=1}^M$ are obtained by sampling from the distribution over model parameters with MCD to obtain the parameter samples $\{f_s\}_{i=1}^M$.

5.2.4 Results

Metrics & approach

We define the set of optimal interventions as the ones in the top percentile of the experimentally-measured phenotype (referred to as 'Top-K interventions'). We use the Top-K recall metric to assess the ability of the different methods to identify the best interventions. To quantify the diversity across the set of optimal interventions, we first cluster these interventions in a lower-dimensional subspace (details provided in Appendix C.1.3). We then measure the proportion of these clusters that are recalled (i.e., any of its members are selected) by a given algorithm over the different experiment cycles. The overall score of an approach is defined as the

geometric mean between Top-K recall and the diversity metric. For all methods and datasets, we perform 25 consecutive batch acquisition cycles (with batch size 32). All experiments are repeated 10 times with different random seeds. We compare DiscoBAX against 16 baselines total: the nine active learning algorithms from the original GeneDisco benchmark [237]) and seven additional methods (UCB, qUCB, qEI, qPOI, Thompson sampling, Top-K BAX, and Levelset BAX).

Results & discussion

We observe that, across the different datasets, DiscoBAX enables to identify a more diverse set of optimal interventions relative to baselines (Table 5.1). It does so in a sample-efficient manner as it achieves higher diversity throughout the different acquisition cycles (Fig.C.1). Note that sample-efficiency is an empirical observation here not a theoretical property of the algorithm since it is possible to construct adversarial datasets where a BAX method will attain no better performance than random sampling. Interestingly, it tends to recall a higher share of optimal interventions on several assays as well, which may be the result of very steep extrema in the corresponding datasets. We also find the performance of DiscoBAX to be relatively insensitive to the choice of hyperparameters (Table C.2). Lastly, we note that when the input feature space (i.e., the intervention representation) does not correlate much with the disease phenotype of interest, the model being learned tends to perform poorly and we observe no lift between the different methods and random sampling (e.g., the SARS-CoV-2 assay from Zhu et al. [240] – see Appendix C.1.3).

5.3 De novo molecular design

5.3.1 Background and objectives

We consider the task of optimizing an expensive black-box objective function taking inputs in a *high-dimensional discrete* space. This could be for example finding new molecules for drug design, or automatically generating a computer program that matches a desired output. Solving this task directly in the original space (e.g., with discrete local search methods such as genetic algorithms) may

Table 5.1: Performance comparison on GeneDisco CRISPR assays We report the aggregated performance of DiscoBAX and other methods on all assays from the GeneDisco benchmark. All other baselines and the breakdown per assay are provided in Appendix C.1.3.

Method	Category	Top-K recall	Diversity score	Overall score
Random	-	29.3% (1.4%)	4.9% (0.3%)	12.0% (0.6%)
Thompson Sampling	Bandits	27.5% (1.5%)	4.8% (0.4%)	11.5% (0.7%)
UCB	Bayesian Optim.	33.5% (2.0%)	5.9% (0.5%)	14.1% (1.0%)
Coreset	Active learning	39.3% (1.9%)	5.5% (0.3%)	14.7% (0.8%)
Levelset BAX	BAX	35.4% (2.2%)	6.3% (0.4%)	15.0% (0.9%)
Top-K BAX	BAX	38.8% (2.3%)	6.8% (0.6%)	16.2% (1.2%)
DiscoBAX (ours)	BAX	**44.1% (2.2%)**	**7.8% (0.5%)**	**18.6% (1.1%)**

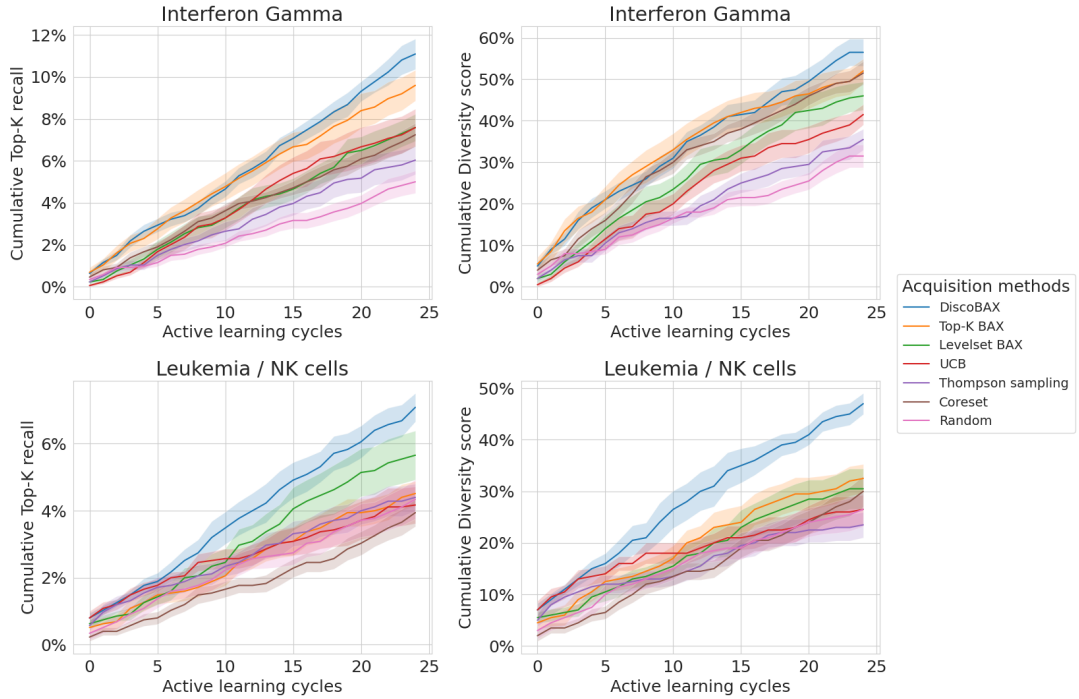

Figure 5.2: Top-K recall and Diversity score Vs acquisition cycles The two top plots are for the Interferon γ assay [241], and the two bottom plots are based on the Leukemia assay [242].

be challenging given the complex structure and high dimensionality of the data. Recently, Variational autoencoders (VAEs) [24, 36] have been successfully leveraged to model a wide range of discrete data modalities — from natural language [108], to arithmetic expressions [42], computer programs [243] or molecules [17]. By learning a lower-dimensional continuous representation of objects in their latent

space, VAEs allow to transform the original discrete optimization problem into a simpler *continuous* optimization one in latent space. For example, this can be achieved via Bayesian Optimization in the latent space, or via gradient ascent with a jointly-trained neural network predicting the black box property from the latent space representation [17, 244]. Initial methods in this area have suffered from the fact that the search in latent space may explore areas for which no data was available at train time, and therefore where the decoder network of the VAE will be unreliable [245]: seemingly good candidate points in latent space may be decoded into objects that are invalid, unrealistic or low quality.

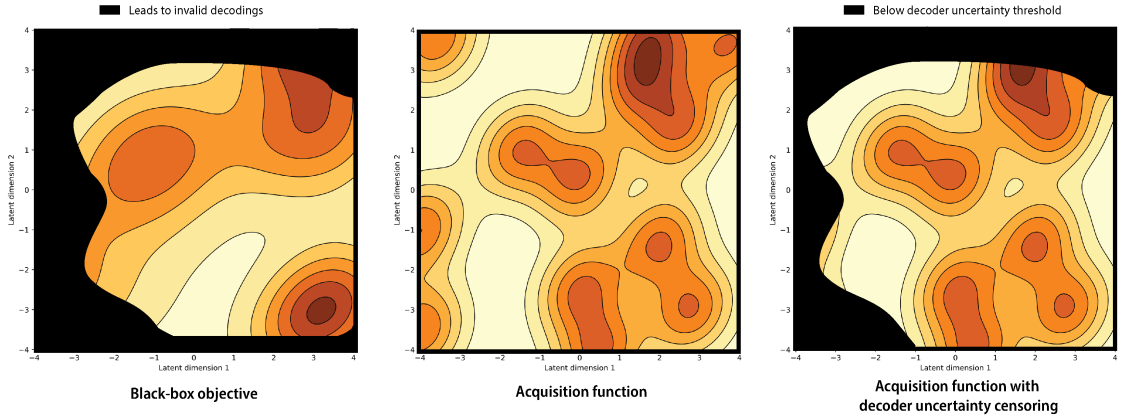

Figure 5.3: Uncertainty-guided optimization in VAE latent space The goal of black-box optimization in latent space is to attain regions with high values of the back-box objective after decoding, while avoiding the regions that lead to invalid decodings (left). Standard Bayesian Optimization in latent space may query these suboptimal areas (e.g., regions on left hand side, center). High decoder uncertainty regions overlap with regions leading to invalid decodings (right), so that censoring high uncertainty points helps guiding the optimization towards the most promising latent points.

While several methods have been introduced to promote validity of decoded objects, they either focus on modifying the generative model learning procedure or adapting the decoder architecture to satisfy the syntactic requirements of the data modality of interest. To improve the validity of decoded sequences, Kusner et al. [42] and Dai et al. [243] develop task-specific grammar rules into the VAE decoder, focusing on use cases in molecular and computer program generation. However, crafting the corresponding rules requires domain-specific knowledge, needs to be designed from scratch for each new task, and may not be straightforward

to elicit in the first place. Griffiths et al. [246] and Liu et al. [247] propose instead to formulate the problem as a constrained Bayesian Optimization and chance-constrained optimization task respectively to simultaneously optimize the target property as well as the probability to generate valid sequences. These two approaches require nonetheless access to a function that quantifies the validity or realism of objects in the training data, which is not readily available in many practical applications. A different line of research has focused on representing high-dimensional structured objects as graphs instead [248, 249]. The Junction Tree VAE (JT-VAE) [250] generates systematically valid molecular graphs, by first generating a tree-structured scaffold over a finite set of molecular clusters, and then assembling these clusters back into molecules with a message passing network. The MolDQN [251] casts the optimization problem as a reinforcement learning task (double Q-learning), which allows in turn to more naturally extend to simultaneous optimization of different objectives. GraphAF [252] combines the strengths of autoregressive and flow-based approaches to efficiently generate realistic and valid molecular graphs. Lastly, Tripp et al. [253] show that the black-box optimization performance can be further enhanced by iteratively retraining the generative model on the points selected during optimization, with weights that depend on their objective function value.

We propose instead to quantify and leverage the uncertainty of the decoder network to guide the optimization process in latent space (Fig. 5.3). Our approach deviates from all the above in that it is is representation-agnostic (works with sequences or graphs), does not require domain-knowledge to craft custom rules or constraints, does not change the model architecture nor the learning procedure, can easily be integrated within several optimization frameworks and can be combined with several of the approaches discussed above to reach even stronger optimization performance (§ 5.3.3). It results in a better trade-off between the values of the black-box objective and the validity of the newly generated objects, sometimes improving both simultaneously.

To be effective, this method requires robust estimates of model uncertainty for high dimensional structured data. Existing methods for uncertainty estimation in this domain often rely on heuristics or make independence assumptions to make computations tractable (§5.3.2). We demonstrate that such assumptions are not appropriate in our setting, and propose new methods for uncertainty estimation in high dimensional structured data instead.

5.3.2 Method

Estimating model uncertainty in high dimensions

We consider discrete output points y that belong to a high-dimensional structured object space $\mathcal{S}$ (e.g., long sequences, large graphs), of cardinality $|\mathcal{S}|$. An exact estimation of the Mutual Information between outcomes y and model parameters θ (Eq. 2.20) is impractical because the expectation involves a sum over exponentially many possible outcomes for $y \in \mathcal{S}$.

In lieu of the heuristics previously discussed, we obtain a principled approximation to the Mutual Information via Monte Carlo estimation using *importance sampling*, with an adequately chosen importance distribution.

We denote $q(\theta) \approx P(\theta|\mathcal{D})$ the learnt approximation to the posterior, and assume we can approximate expectations over model parameters by sampling M independent samples from $q(\theta)$. We can then re-write the Mutual Information $\mathcal{M}$ in Eq. 2.20 as follows:

$$\mathcal{M}(x) \approx -\sum_{s=1}^{|\mathcal{S}|} p_s \log p_s + \frac{1}{M} \sum_{m=1}^{M} \sum_{s=1}^{|\mathcal{S}|} p_{s,m} \log p_{s,m} = \sum_{s=1}^{|\mathcal{S}|} \underbrace{\left[\frac{1}{M} \sum_{m=1}^{M} p_{s,m} \log p_{s,m} - p_s \log p_s \right]}_{h(y_s)} \tag{5.5}$$

where p_s and $p_{s,m}$ are shorthands, respectively, for $P(y = y_s|x, \mathcal{D})$ — the posterior predictive distribution — and $P(y = y_s|x, \theta = \theta_m)$ — the probability of a given output $y_s \in \mathcal{S}$ given x and a sample θ_m from the approximate posterior distribution over model parameters $q(\theta)$.

We can then obtain a tractable approximation to Eq. 5.5 via importance sampling:

$$\mathcal{M}(x) = \sum_{s=1}^{|\mathcal{S}|} h(y_s) \cdot \frac{1}{\bar{p}(y_s)} \cdot \bar{p}(y_s) = \mathbb{E}_{\bar{p}}\left[h(y) \cdot \frac{1}{\bar{p}(y)}\right] \approx \frac{1}{N} \sum_{s=1}^{N} \left[h(\tilde{y}_s) \cdot \frac{1}{\bar{p}(\tilde{y}_s)}\right] \quad (5.6)$$

with $\tilde{y}_s \sim \bar{p}(.)$, where $\bar{p}$ is the importance distribution.

We choose the importance distribution to be the approximate posterior predictive defined over the outputs. We generate an outcome $\tilde{y}_s$ by first sampling a set of parameters $\tilde{\theta}_0$ from the approximate posterior, and then generating $\tilde{y}_s$ from a model defined by that set of parameters $\tilde{\theta}_0$. This distribution will sample mostly from regions in $\mathcal{S}$ with high probability under the true posterior predictive for input x. This is in contrast to a naive sum over all possible outcomes y, many of which will have a negligible contribution to the sum. This gives rise to an estimator of Mutual information (obtained with Algorithm 1) with lower variance than its naive Monte Carlo counterpart (see Appendix C.2.1).

Algorithm 5 Importance sampling estimator of MI

for $s = 1$ **to** N **do**
  Sample $\tilde{\theta}_0 \sim q(\theta)$; $\tilde{y}_s \sim P(y|x, \theta = \tilde{\theta}_0)$;
  for $m = 1$ **to** M **do**
    Sample $\tilde{\theta}_m \sim q(\theta)$; Compute $p_{s,m} = P(y = \tilde{y}_s|z, \theta = \tilde{\theta}_m)$;
  end for
  Compute $p_s = \frac{1}{M}\sum_{m=1}^{M} p_{s,m}$; $h_s = \frac{1}{M}\sum_{m=1}^{M}(p_{s,m}\log p_{s,m}) - p_s \log p_s$;
end for
Return $\mathcal{M}(x) = \frac{1}{N}\sum_{s=1}^{N}\left[h_s \cdot \frac{1}{p_s}\right]$

Uncertainty-guided optimization in VAE latent space

Black box optimization in VAE latent space. We want to optimize the black-box objective $\mathcal{O}$ over a high dimensional discrete object space $\mathcal{S}$. We train a VAE, with encoder g and decoder f, to learn a continuous lower-dimensional embedding of objects in $\mathcal{S}$. The optimization of $\mathcal{O}$ is then performed in latent space and the best candidates are subsequently decoded into the original space. As discussed in § 5.3.1, this may lead to invalid or unrealistic decodings when the decoder f operates in regions different from the ones seen during training. We propose to detect this regime by quantifying the epistemic uncertainty of the decoder for latent

points z (note the notation change compared to the previous subsection where we used x to denote inputs in the general setting): avoiding regions with high epistemic uncertainty for the decoder will make the overall optimization process more efficient by avoiding invalid decodings. We next discuss how we can leverage the uncertainty of the decoder to guide the optimization process for two approaches commonly used in latent optimization settings.

Bayesian Optimization with an uncertainty-aware surrogate model or uncertainty censoring. We first train a surrogate model, e.g., a Gaussian Process [254], to predict $\mathcal{O}(x)$ based on its latent representation z. We then perform Bayesian Optimization using an appropriate acquisition function (e.g., Upper Confidence Bound or Expected Improvement heuristic). There are two main ways to incorporate the decoder uncertainty to guide this process. The first approach consists in training the surrogate model on an objective that penalizes points with high uncertainty (e.g., optimizing $\mathcal{O}(x) - \alpha \cdot \mathcal{M}(z)$). Another method is to censor proposal points z that would have a Mututal Information $\mathcal{M}(z)$ above a predefined uncertainty threshold $\mathcal{T}$ (e.g., highest value observed on the training data) at each step of a batch Bayesian Optimization process (see Algorithm 5.3.2).

Algorithm 6 Bayesian Optimization with uncertainty censoring

Uncertainty threshold T, number of new points to generate N.
Sample M points uniformly at random from the train set, with latent tensor Z and property vector P.
for $i = 1$ **to** N **do**
    Train single task GP on (Z, P) and generate B candidate points $(z_k)_{k\in[\![1,B]\!]}$ with predicted properties $(f_k)_{k\in[\![1,B]\!]}$ by sequentially maximizing the Expected Improvement.
    Compute decoder uncertainty $\mathcal{M}(z_k)$ for $k \in [\![1, B]\!]$.
    if $\exists k \in [\![1, B]\!]$ s.t. $\mathcal{M}(z_k) \leq T$ **then**
        Set new candidate index $k^* = \arg\max_k(f_k)$ s.t. $\mathcal{M}(z_k) \leq T$;
    else
        $k^* = \arg\min_k(\mathcal{M}(z_k))$.
    end if
    Decode new candidate z_{k^*} and obtain true property p_{k^*} of decoded candidate.
    $(Z, P) \leftarrow$ Concatenate (z_{k^*}, p_{k^*}) and (Z, P).
end for

Uncertainty-constrained gradient ascent. A common architecture design when performing black-box optimization in latent space is to jointly train the VAE with an auxiliary network h (Fig.C.3) that predicts the value of the black box objective $\mathcal{O}(x)$ from the encoding z of x in latent space [17, 244, 250]. This construct is particularly useful in constrained optimization settings in which we want to perform a local search in latent space to maximize $\mathcal{O}$ while remaining close to a known input object. The joint training consists of optimizing the sum of the VAE loss (i.e., the ELBO) and the loss from the black-box objective prediction (e.g., MSE loss for a continuous output $\mathcal{O}$) via gradient descent, backpropagating gradients through the entire architecture. To optimize objects under this framework (see Algorithm 5.3.2), we start from a set of initial points in latent space z — either a random sample of latent points, or a subset of points x that we encode in the latent space ($z = g(x)$). We then compute the gradient $\nabla_z h$ of the auxiliary network with respect to z and perform gradient ascent $z \leftarrow z + \alpha \cdot \nabla_z h$. We repeat this process a few times until satisfying a stopping criteria (e.g., threshold on predicted values $h(z)$ or after a fixed number of gradient updates). Finally, we decode the latest latent positions to obtain the set of candidates $\tilde{x} = f(z)$ and measure their actual properties $\mathcal{O}(\tilde{x})$. We can further improve the quality of the candidate set by censoring the moves in latent during gradient ascent that would result in a value of uncertainty above a predefined threshold $\mathcal{T}$.

Algorithm 7 Uncertainty-constrained gradient ascent

Uncertainty threshold T, number of gradient updates N, gradient scale α.
Sample M points from the train set, with latent tensor $Z = (z_k)_{k \in [\![1,M]\!]}$ and property vector P.
Compute $\nabla_z h(Z)$, where h is the auxiliary network predicting P from Z.
for $i = 1$ **to** N **do**
  for $k = 1$ **to** M **do**
    if $\mathcal{M}(z_k + \alpha \nabla_z h(z_k)) \leq T$ **then**
      $z_k \leftarrow z_k + \alpha \nabla_z h(z_k)$.
    end if
  end for
end for
Decode final positions Z^*.
Obtain true properties P^* of decoded points.

5.3.3 Results

Experimental setup

Molecular generation for drug design seeks to identify new molecules satisfying desired chemical properties. Molecules are typically either represented as sequences of characters, using their SMILES representation [81], or as graphs of atoms [248]. We demonstrate the effectiveness of the approach described in § 5.3.2 for these two different representations: experiments with the 'Character VAE' (CVAE) for molecules [17] leverage the SMILES representation, while experiments with the JT-VAE [250] are based on a graph representation of molecules. For both architectures, we trained our models on a set of 250k drug-like molecules from the ZINC dataset [255].

Uncertainty estimators and baselines Across all experiments, we quantify the Mutual Information between outputs and decoder parameters with both the Importance sampling estimator (IS-MI) described in §5.3.2 and based on the token independence approximation (TI-MI) described in §2.3. Sampling from model parameters is achieved via Monte Carlo dropout [239]. We compare optimization results with two baselines: the standard approach that fully ignores decoder uncertainty, and an approach in which we censor proposal points with low probability under the prior distribution (standard normal) of the VAEs in latent space (referred to as 'NLLP').

Optimization We perform uncertainty-guided optimization in latent space as per the two approaches described in §5.3.2. For Bayesian Optimization, we train a single task Gaussian Process as our surrogate model based on a random subset of training points embedded in latent space and their corresponding black-box objective values. We then perform several iterations of batch Bayesian Optimization using the Expected Improvement heuristic as our acquisition function. At each iteration we select a batch of 20 latent vectors by sequentially maximizing the acquisition function. We select the point with the highest predicted target value for which the decoder uncertainty is below a predefined threshold (e.g., 99^{th} percentile of decoder uncertainty values observed on the training set) or the one with lowest uncertainty

if no point in the batch is below the threshold. We re-train the surrogate model with the newly generated point at each step. For gradient ascent, we randomly sample points from the training set, embed them in latent space, and use these as our starting positions. We then compute the gradient of the auxiliary property network with respect to latent positions and accept proposal moves along these directions if the decoder uncertainty at the corresponding position in latent is below a predefined threshold (e.g., 99^{th} percentile of decoder uncertainty on the training set). For practical considerations, we suggest to always start with a high value for the threshold to not unnecessarily constrain the optimization, and move to stricter uncertainty constraints if the validity or quality is not high enough for the particular use case considered (see Table C.6 for an analysis on the impact of that hyperparameter choice). All optimization experiments reported below are carried out 10 times independently with different random seeds.

Character VAE (CVAE)

Setup We jointly train a CVAE model, which learns to encode and decode molecules SMILES strings, along with an auxiliary property network that predicts a target property of these molecules. Following prior work [42, 243, 250], we define the black-box objective as the octanol-water partition coefficient penalized by the synthetic accessibility score and the number of long cycles, (Appendix C.2.2) and we refer to it as 'Penalized logP' for brevity. Since the SMILES representation of molecules follows a strict syntax that determines whether a given expression is valid or not, we are interested in generating molecules that simultaneously maximize the target property and represent valid SMILES expressions.

Results We first verify that our estimator is able to discriminate points in-distribution (low uncertainty) vs out-of-distribution (high uncertainty). We consider 4 distinct sets of points in latent: embeddings into latent space of a random sample from the train and test sets, random samples from the VAE prior (standard normal) and random samples "far from the prior" (we sample from an isotropic gaussian with standard deviation equal to 10). As can be seen on Fig.5.5a, uncertainty

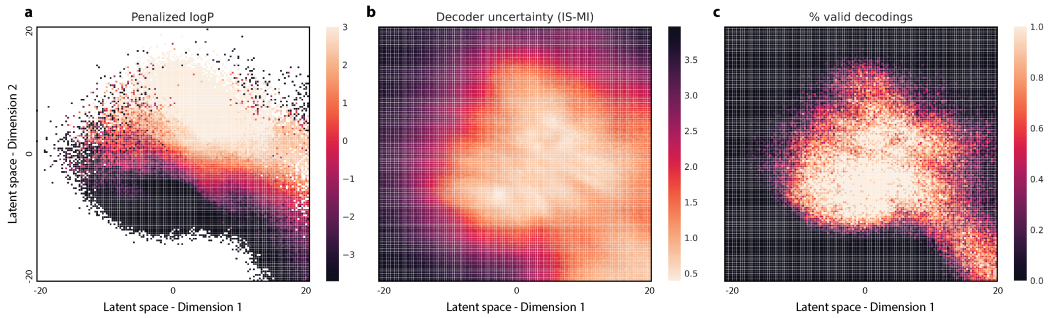

Figure 5.4: CVAE latent space visualization. We apply Principal Component Analysis on the embedding of the full training data and keep the first 2 components. We then create a grid on the resulting 2D-space and measure the penalized logP (a), decoder uncertainty (b) and the proportion of valid decodings in that region (c) (a & c averaged over 300 decodings; b measured via IS by sampling 100 times from the importance distribution, and averaging over 100 samples of model parameters.; for a, white squares correspond to regions where none of the 300 decodings are valid). We observe a strong overlap between decoder uncertainty (b) and validity of decodings (c).

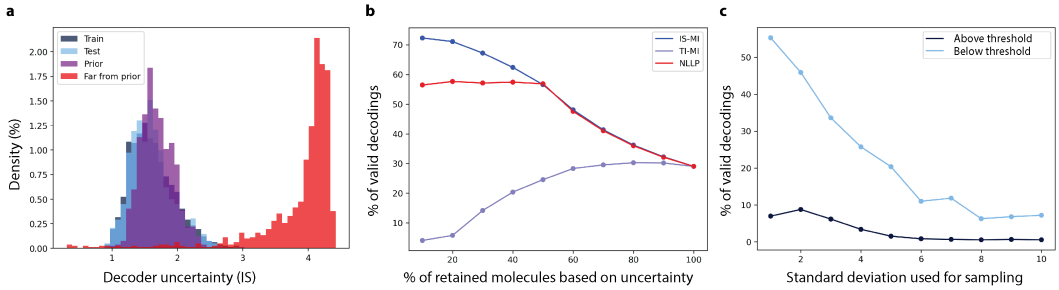

Figure 5.5: Uncertainty estimator. a) Distribution of decoder uncertainty values (IS-MI) for 1k samples for 4 distinct sets (train & test set samples embedded in latent space; samples from the prior; samples far from the prior). b) Valid decodings (%) as a function of the proportion of samples kept based on their uncertainty — eliminating points with high uncertainty first (dataset comprised of 50% samples from test set & 50% of samples far from the prior). The IS-MI estimator has superior ability to identify points leading to invalid decodings. c) Valid decodings (%) for samples from a normal distribution with increasing standard deviation. Samples with decoder uncertainty below a predefined threshold (maximum IS-MI value observed on training data) have a much higher rate of valid decodings. Points above the threshold are very likely to lead to invalid decodings.

estimates for the first 3 sets strongly overlap while being disjoint from the estimates corresponding to points far from the prior. Furthermore, we observe a strong correlation between low decoder uncertainty and regions that lead to valid SMILES decodings (Fig. 5.4). This is corroborated by the analysis described in Fig. 5.5c: when considering latent points "far from the prior", points for which the decoder uncertainty is lower than a predefined threshold (e.g., maximum value observed

on training data) will lead to a significantly higher proportion of valid decoded molecules compared to latent points with uncertainty above the threshold. This is critical as it allows to censor points that will likely lead to invalid decodings, even when we move *far from the prior* in latent space.

For the Bayesian Optimization experiments, we investigate the impact of different bounds on the space we optimize within, as well as different uncertainty thresholds. As we increase the bounds, we typically reach higher optima, at the cost of a higher fraction of invalid decodings during search. We obtain higher validity % and penalized logP values when leveraging the decoder uncertainty (Table 5.2). In this setting, the token-level independence assumption (TI-MI) leads to poor performance compared to the importance sampling-based estimator (IS-MI). Results are also robust to the choice of decoder uncertainty thresholds (Table C.6).

Table 5.2: CVAE - Bayesian Optimization results. Censoring proposal points with high decoder uncertainty values with the IS-MI estimator helps increase validity across experiments. As we increase the bounds on the Bayesian Optimization search space, validity % generally decreases but remains 5-10x higher when leveraging IS-MI compared to baselines. This is critical as is helps uncover molecules with very high penalized logP values.

Search bounds	**Decoder uncertainty**	**Penalized logP**		**Validity (%) ↑**
		Top 1 ↑	**Avg. top 10 ↑**	
5	None	4.0 ± 0.2	2.5 ± 0.2	$22\% \pm 1.4\%$
	NLLP	4.2 ± 0.2	2.7 ± 0.1	$30\% \pm 1.3\%$
	TI-MI	4.1 ± 0.3	2.3 ± 0.1	$21\% \pm 0.8\%$
	IS-MI	$\mathbf{4.5 \pm 0.2}$	$\mathbf{3.0 \pm 0.1}$	$\mathbf{33\% \pm 1.8\%}$
10	None	3.9 ± 1.2	-2.3 ± 2.8	$1\% \pm 0.4\%$
	NLLP	2.9 ± 0.8	0.5 ± 0.8	$3\% \pm 0.7\%$
	TI-MI	5.9 ± 3.6	1.1 ± 1.5	$2\% \pm 0.4\%$
	IS-MI	$\mathbf{6.6 \pm 0.6}$	$\mathbf{1.6 \pm 0.8}$	$\mathbf{11\% \pm 0.8\%}$
15	None	10.3 ± 4.3	5.0 ± 2.6	$1\% \pm 0.3\%$
	NLLP	3.9 ± 2.5	0.8 ± 1.2	$1\% \pm 0.3\%$
	TI-MI	6.7 ± 3.8	6.4 ± 3.9	$1\% \pm 0.3\%$
	IS-MI	$\mathbf{27.6 \pm 2.2}$	$\mathbf{9.9 \pm 1.3}$	$\mathbf{5\% \pm 0.7\%}$

Junction Tree VAE (JT-VAE)

Setup We train a Junction Tree VAE model (JT-VAE) [250] using the same dataset of 250k molecules (ZINC) and black-box objective (penalized logP) as for the CVAE experiments. All molecules generated by the JT-VAE are valid by design. However, not all generated molecules will be of high *quality*, which we assess with the quality filters proposed by Brown et al. [256] that aim at ruling out "compounds which are potentially unstable, reactive, laborious to synthesize, or simply unpleasant to the eye of medicinal chemists." We show that it is straightforward to attain state-of-the-art performance in terms of penalized logP values with the basic optimization approaches described in § 5.3.2 by moving sufficiently 'far away' in latent, but that in doing so we tend to generate molecules that never pass quality filters. Factoring in decoder uncertainty during optimization helps generate new molecules with both high penalized logP values and high quality.

Using notations from § 5.3.2, sampling a new object $\tilde{y}_s$ is achieved by successively decoding from the junction tree decoder and then the graph decoder. We then decompose $\log p_{s,m}$ – the log probability of the sampled graph molecule – as the sum of the log probabilities corresponding to the different predictions made by the junction tree decoder and graph decoder, namely the topology and node predictions in the junction tree decoder, and the subgraph prediction in the graph decoder.
We replicate the analysis described in 5.3.3 with the 4 distinct datasets in latent space (i.e., train, test, prior and far from prior) and observe similar results: the histogram of decoder uncertainty values for points "far from the prior" is disjoint from the other three histograms (Appendix C.2.3), confirming the ability of the estimator to identify out-of-distribution points.

Results Both gradient ascent (Table 5.3) and Bayesian Optimization (Appendix C.2.3) allow to generate new molecules with state-of-the-art performance in terms of penalized logP (Table C.8). However, the majority of these molecules do not pass quality filters. Leveraging decoder uncertainty leads to the generation of high logP and high quality molecules. Using likelihood under the prior (NLLP) to achieve the same is detrimental to optimization performance.

Table 5.3: JT-VAE - Gradient ascent results. We obtain state-of-the-art performance in terms of penalized logP via gradient ascent. However, most generated molecules are of very low quality (only $\sim 1\%$ pass the quality filters from Brown et al. [256]). Leveraging the uncertainty of the decoder (IS-MI) during optimization helps generating molecules with high penalized logP and high quality. NLLP constraints help maintain high quality but lead to suboptimal black-box objective values. Results with different hyperparameters and threshold values for each method are reported in Table C.9.

Decoder uncertainty	Penalized logP - Before filters Top 1 ↑	Penalized logP - Before filters Avg. top 10 ↑	Quality top 10 (%) ↑	Penalized logP - Passing filters Top 1 ↑	Penalized logP - Passing filters Avg. top 10 ↑
None	$\mathbf{23.7 \pm 1.3}$	$\mathbf{17.0 \pm 0.6}$	$1\% \pm 1\%$	1.2 ± 1.2	0.3 ± 0.3
NLLP	3.0 ± 0.1	2.5 ± 0.1	$82\% \pm 6\%$	3.0 ± 0.1	2.0 ± 0.2
IS-MI	8.4 ± 10.8	6.0 ± 0.3	$\mathbf{89\% \pm 3\%}$	$\mathbf{7.7 \pm 0.7}$	$\mathbf{5.3 \pm 0.3}$

5.4 Protein design

5.4.1 Background: Machine learning for functional protein design

Proteins fulfill a wide range of functions in nature, with that diversity encoded in the underlying amino acid sequences. The goal of protein design is to create new proteins by discovering sequences with functions that enhance or extend those of existing proteins, with the potential to address our most pressing problems in healthcare, agriculture, and sustainability. However, the potential design space is massive and sparsely functional. Given the impossibility to exhaustively list out all combinations — let alone quantify their properties experimentally or computationally, one of the first challenges that protein design is faced with is narrowing the search to a tractable space. A multitude of strategies have emerged to address this challenge, from methods manually selecting the most promising mutants based on a deep understanding of a given protein structure and function (e.g., rational design [257]), to experimental methods testing a broader range of variants (e.g., directed evolution [32], combinatorial libraries [258]) to biophysics-based models of the protein structure, folding and interactions (e.g., computational design [259]). Machine learning methods have recently emerged as another strategy to efficiently explore the massive design space, with unique strengths that are quickly redefining what is possible in protein design.

Objectives of machine learning for protein design

The design objectives supported by machine learning methods can broadly be classified in three groups, depending on whether we start from scratch or a known protein and, in the latter case, whether we change or enhance the existing function (Fig. 5.6). We review these different design paradigms and connect them with the machine learning approaches discussed in the next subsection (§ 5.4.1).

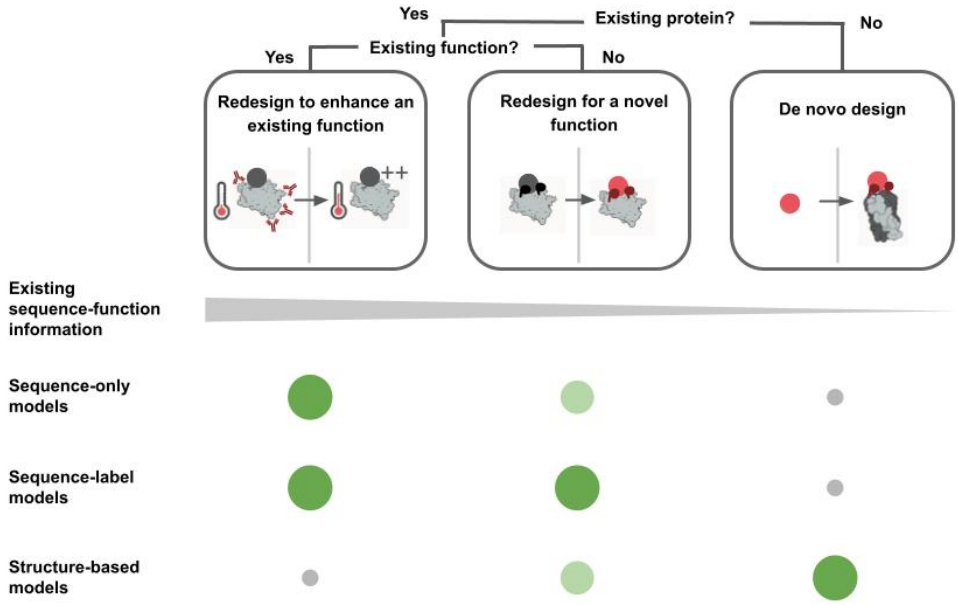

Figure 5.6: Objectives of machine learning for protein design. The various protein design objectives can be broadly grouped in 3 categories based on how far the design target is from existing functional proteins. For each group, different model classes may be more appropriate depending on available sequence-function information. The size of the circle indicates how suited a given approach is to a particular design task.

Redesign to enhance an existing function. The goal of protein enhancement is to start from a protein (natural or otherwise) that already possesses the desired function and introduce mutations to improve its properties or achieve the original biological function in different conditions. The enhanced property could be the main function of the protein (e.g., catalytic activity, binding affinity to a specific target) or another of its attributes. For instance, the objective may be to enhance the thermostability or solubility of the protein while maintaining its original function [260], or mitigating undesirable interactions with other molecules, such

as reducing the immunogenicity of therapeutics by altering epitopes [261]. One way to enhance intrinsic properties such as stability is to sample high probability sequences from a sequence-only model [262, 263] or sequence-structure model [260]. Sequence-only models can also be used to generate libraries of 'new family members' to pan for secondary properties (e.g., chorismate mutase functionality in E. coli [264] or antibody binding affinity [265]). When mutation-phenotype data exists for the property of interest, sequence-label models can prove useful, e.g. for stability prediction [266] or prediction of immune epitopes to guide design [267].

Redesign for a novel function. In this scenario, the objective is to design a protein with a novel function by working from an existing protein with a related function (e.g., shifting a binder or enzyme to act on a new target [268]). This requires either a detailed understanding of the mechanism of function, or ample data relating sequence to substrate or product and, consequently, most approaches have relied on sequence-label models. These data can be obtained by selecting sequences according to the reaction of interest, either via measured phenotypes of natural proteins, deep mutational scans, or next-gen sequencing (NGS) of directed evolution experiments [269].

De novo design. Machine learning design of sequences with de novo folds focuses on sequence-structure models. These methods can generate sequences with diverse 3D folds and multimer arrangements with a high success rate of stable expression [270, 271]. The motivation to design sequences based on 3D structure results from the large role of structure in our understanding of protein function; 3D structure enables us to make assertions about physicochemical interactions and is a convenient representation for inferring or specifying constraints on function. De novo design requires function constraints on sequence and structure, often derived from other existing proteins, such as metal-binding sites and fragments of protein-protein complexes [272]. Excitingly, it is becoming possible to design a de novo protein from the target protein alone for both protein-protein binding [19] and small-molecule binding [273]. Recently, luciferase enzymatic activity has been achieved by a de

novo protein, albeit templated on an existing family of small-molecule binding proteins [274]. Although some applications require thousands of designs to be assayed, the capabilities of de novo design are growing rapidly.

Typology of protein design models

The rich diversity of machine learning models for protein design can broadly be categorized in three groups, based on the type of representation of protein objects and the data leveraged during training (Fig. 5.7).

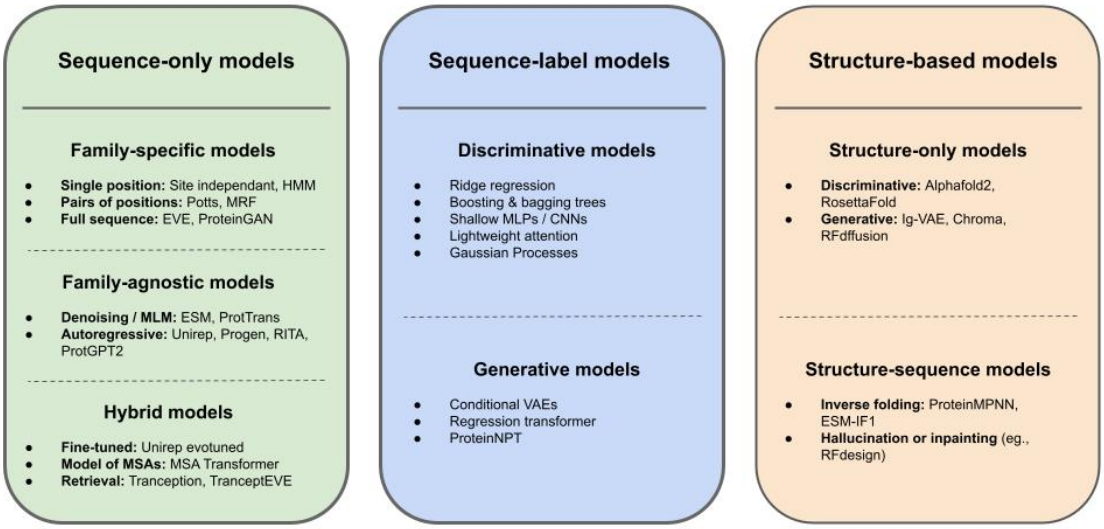

Figure 5.7: Typology of protein design models. Machine learning methods for protein design can be categorized in 3 grouped depending on the data modalities used to train them.

Sequence-only models. We discussed this model class extensively in Chapter 3. The objective is to learn a generative model solely based on the primary structure of a large collection of proteins, with the aim of implicitly capturing the biochemical constraints that characterize the proteins present in the training set. Models in that category were classically 'family-specific' and trained on a set of homologous sequences obtained with a Multiple Sequence Alignment (MSA) [38, 107, 112, 275]. Building on the intuition that certain amino-acid constraints or patterns may

generalize across protein families, 'family-agnostic' models trained on unaligned sequences across protein families then emerged as a practical alternative covering all proteins with a single model [39, 40, 118, 120, 127, 128]. However, without relying more explicitly on homology, even the best family agnostic models would barely match the fitness prediction abilities of classical family-specific model (e.g., ESM-1v being on par with Potts models in the zero-shot setting, and matching the performance of DeepSequence only when fine-tuned on MSAs [41]). This observation subsequently gave rise to a multitude of 'hybrid models' that sought to combine the relative strengths of each approach, e.g., MSA Transformer [130], Tranception (§ 3.4.2), TranceptEVE (§ 3.4.4).

Sequence-label models. When a sufficiently large number of labels for the property of interest are available, it becomes possible to train discriminative supervised models learning a mapping between a representation of the protein and the corresponding property. Labels are typically measurements collected in the lab via next-gen sequencing (NGS) of directed evolution campaigns, mutation libraries [276], or partial measurements that supplement natural sequence data [277]. The representation can be either a simple one-hot encoding of the sequence, physico-chemical properties, or other hand-crafted features or, increasingly, the embeddings extracted from sequence-only models giving rise to label-efficient semi-supervised architectures. The trained regressor is typically lightweight to avoid overfitting, e.g., ridge regression, shallow CNN or dense network, Gaussian Process (GP) [142, 144, 278]. While the majority of supervised models has relied on representation of the primary structure, some architectures have also been proposed based on the tertiary structure [279].

Structure-based models. Two broad types of structure-based models may be used for design. 'Sequence-structure models' [103, 271, 280] are trained on known or predicted sequence-structure pairs. On the other end, 'structure-only models' may generate 3D backbones without input sequences [281, 282]. To eventually produce novel sequence designs, these models must be paired with sequence-structure models.

Choosing a model architecture. The main drivers in the selection of a particular model are the desired design objective (as discussed in section 1) and available data (e.g., a sufficiently large number of labels to train sequence-label models). Another consideration is how the model will be effectively used in practice. Generative models — whether they are sequence-only or structure-based — provide a way to sample novel proteins that resemble the data they have been trained on. A sound sampling process producing natural-like proteins coupled with a robust experimental pipeline to measure the actual properties of generated objects is key. Additional controls (e.g., taxonomic labels [118], enzyme classification [283]) with which we can condition the sampling process may help in increasing the usefulness of each sample. Alternatively, sequence-label architectures provide diverse ways to prioritize a subset of variants for subsequent experimental validation. For instance, if the model outputs both predictions along with the corresponding uncertainty (e.g., Gaussian Process, Bayesian Neural Network), one can frame the iterative design approach under a batch Bayesian optimization framework. In the next subsection, we illustrate this approach by leveraging the ProteinNPT model introduced in § 4.2.4.

5.4.2 Iterative protein redesign with ProteinNPT

Setup

In this section, our objective is to demonstrate that the ProteinNPT architecture (§ 4.2.4) is effective for iterative protein design. This design paradigm falls under the "Redesign to enhance an existing function" category described in § 5.4.1. It typically starts from a sequence with the desired function, and iteratively makes mutations to that sequence until the desired property level is achieved or the experimental budget is exhausted. Since no or very few labels are usually available in the early stages of design, doing well on that task requires the underlying machine learning model to be data efficient. Intuitively, ProteinNPT is well suited to these label-scarce settings given the high quality protein embeddings it relies on and thanks to its additional denoising objective which promotes regularization.

Although all experiments and evaluation in this section are carried out in silico, our aim is to mirror a real-world design scenario in which several rounds of experiments are conducted iteratively, with the knowledge gained in earlier rounds being utilized to inform and steer the direction of subsequent experiments. As discussed in § 5.4.1, this iterative task can be effectively addressed with Bayesian optimization (§ 2.4.1) and we can leverage the outcomes of DMS experiments as the oracle providing property values for query points. However, unlike a real life setting, we are constrained to select the mutants to query from the set of mutants that were experimentally measured in the corresponding DMS assays. While newly queried points can continuously enhance protein function in real-life experiments, favorable mutations eventually run out in the setup outlined in this section. Consequently, the model has to choose from what can be described as the 'best among the worst' options. Despite this constraint, our approach should still allow us to assess the capability of various models to select the best points based on limited information during the early stages of iterative design. We choose a representative set of 24 assays from ProteinGym (§ 4.2.2), encompassing a range of MSA depth, taxa, and number of labels available (Appendix C.3.2). We compare ProteinNPT with the same supervised baselines discussed in § 4.2.5.

Method

We describe the approach followed in our in silico iterative design experiments in Algorithm 8. We first select an initial labeled data $\mathcal{D}_L$, drawing points at random from the set $\mathcal{D}$ of all mutants in the corresponding DMS assay, and keep all other points as our unlabeled pool set $\mathcal{D}_U = \mathcal{D} \backslash \mathcal{D}_L$. At each cycle, we first train the considered model (ProteinNPT or baselines) on $\mathcal{D}_L$. We then predict the property and quantify our prediction uncertainty for all possible variants in $\mathcal{D}_U$. We do so by performing Monte Carlo (MC) dropout [239] and extracting the mean $\mu(x)$ and standard deviation $\sigma(\boldsymbol{x})$ across each forward pass for each point $\boldsymbol{x} \in \mathcal{D}_U$. We then sequentially acquire a batch of B points by greedily optimizing the Upper

Confidence Bound (UCB) acquisition function $\alpha(\boldsymbol{x}; \lambda) = \mu(\boldsymbol{x}) + \lambda\sigma(\boldsymbol{x})$, where λ is an hyperparameter controlling the exploration/exploitation trade-off.

Algorithm 8 In silico iterative design experiment

Input: Initial labeled data $\mathcal{D}_L$; Initial unlabeled data $\mathcal{D}_U$; Batch size B; Batch set $S = \emptyset$; Acquisition function $\alpha(\boldsymbol{x}; \lambda)$
for $t \in 1, 2, \ldots, 10$ **do**
  Train model model on $\mathcal{D}_L$
  for $t \in 1, 2, \ldots, B$ **do**
    Select $\boldsymbol{x}_{\text{new}} = \arg\max_{\boldsymbol{x} \in \mathcal{D}_U} \alpha(\boldsymbol{x}; \lambda)$
    Obtain measurement y_{new} for $\boldsymbol{x}_{\text{new}}$ from DMS assay results
    $S \leftarrow S \cup \{(\boldsymbol{x}_{\text{new}}, y_{\text{new}})\}$, $\mathcal{D}_U \leftarrow \mathcal{D}_U \backslash \{\boldsymbol{x}_{\text{new}}\}$
  end for
  $\mathcal{D}_L \leftarrow \mathcal{D}_L \cup S, S \leftarrow \emptyset$
end for
Output: $\mathcal{D}_L$, Trained model on $\mathcal{D}_L$

Results

We first confirm that MC dropout provides sensible uncertainty estimates. For the different cross-validation schemes described in § 4.2.5, we plot the Mean Squared Error (MSE) by censoring an increasing proportion of test set points with the highest uncertainty. We observe that the MSE monotonically decreases on more confident subsets of points, validating the proper calibration of our uncertainty estimates (Appendix C.3.1).

We then proceed to the various iterative design experiments as per the experimental setup described above. For each assay, we plot the proportion of acquired points at each cycle with fitness in the top 3 deciles for the corresponding DMS assay. We observe that the ProteinNPT model markedly outperforms all baselines in aggregate performance over all assays (Fig. 5.8), and outperforms other baselines on the majority of individual assays (20 out of 24 assays; see detailed results in Appendix C.3.2).

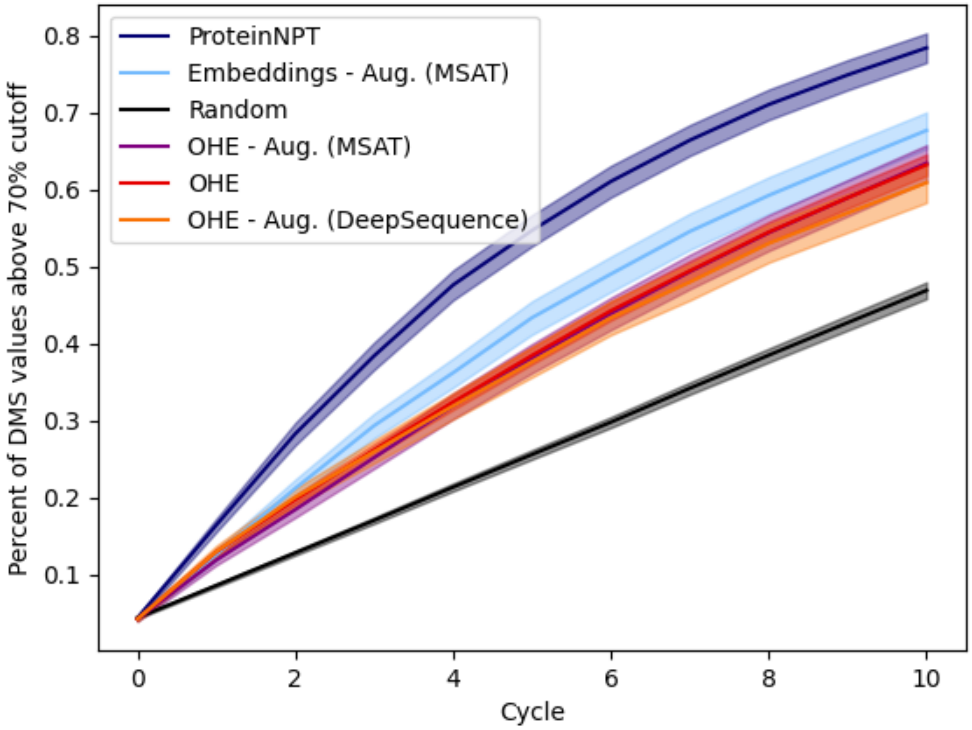

Figure 5.8: In silico iterative protein redesign experiments. We compute the proportion of acquired points at each cycle for which the fitness is in the top 3 deciles for the DMS assay. Results are then averaged across the 24 DMS assays used in this experiment.

6 Conclusion and Future Directions

Contents

We conclude this thesis by summarizing our findings related to the key scientific questions highlighted in introduction (§ 1) and discussing future research directions.

6.1 Represent

Q1. How can we best learn general-purpose representations of protein sequences across families? Transformers excel at large-scale training due to their parallelizable nature and superior handling of long-range dependencies [1]. We found that, in zero-shot fitness predictions, autoregressive transformer models consistently surpassed Masked-Language Model (MLM) transformer alternatives. For example, the autoregressive models RITA [127], Progen2 [126], and Tranception [137] all outperformed the MLM models ESM-1b [40] and ESM-1v [41] on the ProteinGym substitution benchmark when comparing similar-sized, non-fine-tuned networks without retrieval, with Tranception markedly outperforming all architectures in that setting. The performance edge of autoregressive models may stem from their ability

to capture important epistatic effects between several mutated positions, which are overlooked in the masked-marginal approach typically used with MLM architectures. However, this leads to the trade-off where embeddings from autoregressive models depend solely on the left context, in contrast to the bidirectional context dependencies in MLMs. This limitation can be partially mitigated in zero-shot fitness prediction by scoring sequences from N-terminus to C-terminus and vice versa, and then averaging the results. Recently-developed learning paradigms such as fill-in-the-middle (FIM) modeling [46] or meet-in-the-middle (MIM) modeling [284] may eventually offer further improvements. Additionally, autoregressive architectures naturally handle insertions and deletions, unlike MLM approaches for which the masked-marginal heuristics necessitate the position to exist in the wild-type sequence. Interestingly, when it comes to using protein language models as the backbone for semi-supervised architectures, MLMs like ESM-1v tend to provide superior embeddings compared to autoregressive models like Tranception, as evidenced in the ablation studies in Appendices B.1.4 and B.1.4. In these supervised experiments, we further note that the best results were consistently achieved using the MSA Transformer [130], which brings us to the following question.

Q2. Do the corresponding model architectures perform on par with family-specific methods? Despite recent progresses in family-agnostic protein language models, both MLM and autoregressive architectures fall short in terms off zero-shot fitness prediction performance compared to leading family-specific models such as GEMME [285] and EVE [135]. For instance, an ESM model (650M parameters) without fine-tuning performs on par with a site-independent model, and an ensemble of five non-fine-tuned ESM models equals the performance of a Potts model trained on retrieved MSAs [41]. This finding is also observed in protein structure predictions, where AlphaFold2 [16] - a model heavily reliant on MSAs - outperforms ESMFold [286], a top-performing model using single-sequence input. As of today, the only way for protein language models to perform on par with family-specific models is to explicitly leverage homology, whether it is via standard

fine-tuning [41], prompt tuning [127], learning a model of MSAs across families [130] or, as we have advocated in this work, via retrieval (§ 3.4.2).

Q3. Do family-specific and family-agnostic models capture redundant information, or could they be used synergistically? As we have demonstrated in § 4.2.3, TranceptEVE 3.4.4 efficiently combines together the family-specific EVE with the family-agnostic Tranception to reach state-of-the-art zero-shot fitness prediction performance. In particular, it surpasses the individual performance of each of the underlying models, further demonstrating that each class of model captures different yet complementary information.

Future directions

Retrieval at Training Time. Our work has shown that retrieval at inference time can be a flexible and modular approach for including information from homologous sequences into our architecture (§ 3.4.3). However, end-to-end retrieval, wherein the model is trained to retrieve the right sequences, weigh them appropriately, and learn how to use these sequences for fitness prediction or generation, could further enhance predictive performance.

Hybrid Models of Structure and Sequence. Another exciting direction is the development of hybrid models that combine information from both sequence and structure. The models we have focused on in this thesis have been primarily sequence-only, with no structural information explicitly provided. By combining sequence models with structural ones, we can leverage the granular understanding offered by structure models while taking advantage of the larger coverage of protein space conferred by sequence-only models [104].

Model and data scaling. Preliminary analyses into the scaling laws for protein language models [127] have highlighted the benefits from scaling model size and, perhaps, hinted to some of its limits [126]. If trends observed in NLP [287] hold for proteins, leveraging larger datasets of protein sequences during training will yield improved generative models.

6.2 Predict

Q4. How can we robustly assess the capability of various models to predict protein fitness, in the zero-shot and supervised settings? The main challenge when attempting to compare different fitness predictors in the zero-shot setting is the significant variation in relative model performance across different DMS assays. Tackling this issue requires both scale (in terms of number of assays) and diversity (in terms of taxa, alignment depths, mutational depths, functions). This observation has led to the development of ProteinGym 4.2.2, the largest collection of DMS assays for fitness prediction benchmarking. In the supervised regime, attention must be given to potential data leakage when assessing model generalization to unseen positions. For instance, a random Cross Validation (CV) scheme may result in training and test sets that include labelled instances with mutations at identical positions. Since predicting the effects of mutations at positions observed in training is substantially easier than generalizing to unseen positions, solely focusing on a random CV scheme may lead to an over-optimistic performance evaluation. To mitigate this issue, we introduced two additional CV schemes (§ 4.2.5) that provide more robust generalization claims in the supervised regime by carefully separating the mutated positions across training and test datasets.

Q5. What is the performance lift provided by supervised methods over their zero-shot counterparts when predicting protein fitness? The semi-supervised approaches introduced in § 4.2.5 and § 4.2.5, in particular the ProteinNPT, confer a significant performance boost over zero-shot methods across the various CV schemes (Table 4.5). The performance lift is substantial when

considering the random CV scheme, and is halved when considering the modulo and contiguous CV schemes, further underscoring the importance of points made in the previous paragraph.

Q6. Can we predict the effects of missense mutations on human disease risk in a fully unsupervised manner? Predicting the effects of missense mutations on human disease is challenging in practice since the number of manual annotations that could be used for training are very limited for the large majority of genes, and subject to various biases (§ 4.3.1). We nonetheless demonstrated in § 4.3.3 that a fully unsupervised approach based on generative models of evolutionary data (EVE) could yield accurate predictions of genetic mutation effects. Interestingly, this method outperformed prior supervised methods (Fig. 4.7), and performed on par with experimental approaches (Fig. 4.8).

Q7. Can we predict the propensity of viral mutations to induce immune escape, early in a pandemic? The ability to predict viral variants that can evade immune detection ahead of time is crucial for effective pandemic surveillance and optimal vaccine and therapeutic development(§ 4.4.1). EVEscape (§ 4.4.2) is an unsupervised approach that relies entirely on information available early in a pandemic, such as evolutionary sequences and viral protein structures – before surveillance sequencing, antibody-antigen structures or experimental mutational scans become broadly available. As observed in § 4.4.3, this method not only outperforms prior computational methods, but also matches the accuracy of predictions from high-throughput experiments.

Future directions

Unified Multi-Modal Architecture. Our work on fully unsupervised (e.g., EVE, Tranception, TranceptEVE) and semi-supervised (e.g., ProteinNPT) model architectures has demonstrated the value of learning from both unlabeled and labeled sequences. However, our semi-supervised approaches rely on assay-specific

training, and each model does not use labels from other experiments and protein families. An exciting area of future research would consist in building a unified semi-supervised architecture across protein families, property types, assay datasets and other modalities, such as gene ontologies [288] and biomedical text [289].

Full Genome Modeling. Our focus in this thesis has been on protein-coding variants. The next logical step is to build holistic genomic models that capture similarities and dependencies across the entire genome, potentially unlocking new insights on the mutations affecting regulatory regions controlling gene expression and responsible for numerous pathologies.

6.3 Design

Q8. In the process of identifying gene targets for new therapeutics, can we guide the experiments towards selecting interventions that are both diverse and optimize the phenotype of interest? Identifying gene targets for novel therapeutic design demands sequential decision making methods that can efficiently explore a vast experimental space. The selected candidates should be both effective and diverse to maximize the chances of success in subsequent stages of the drug development pipeline (§ 5.2.1). To overcome the limitations of standard sequential methods such as Bayesian optimization or active learning (§ 2.4), we developed DiscoBAX (§ 5.2.3), a general-purpose and data-efficient sequential decision making algorithm which promotes the selection of interventions that are both diverse and effective.

Q9. During the de novo optimization of small molecules with variational autoencoders, how can we ensure the optimization in latent space eventually yields high-quality decodings? De novo molecular design can be conceptualized as a complex optimization problem over linear sequences [81] or molecular graphs [248] (§ 5.3). Recent work has explored the use of Variational Autoencoders (VAEs) to convert this discrete optimization task into a continuous

one in the latent space [17]. While this transformation can make optimization more manageable, it also introduces new challenges, such as the possibility of optimization to explore areas in the latent space that are far from the training data and thus difficult for the VAE to decode from. To mitigate this issue, we proposed to leverage the uncertainty of the decoder to guide the continuous optimization in latent space, and avoid regions that will likely lead to poor decodings (§ 5.3.2).

Q10. When optimizing protein properties iteratively, how can we best leverage the scarce labels we progressively collect to enhance the overall design performance? Efficiently optimizing the properties of a target protein necessitates the use of models that accurately predict the relevant properties. Considering the extensive cost of experimentation and the vastness of the design space, it is imperative for these methods to be sample-efficient. In § 5.4.2, we illustrated the robust performance of the ProteinNPT architecture in this regard, showing a marked improvement over prior semi-supervised methods.

Future directions

Uncertainty in Transformers. In § 5.3.2, we introduced an importance-sampling-based estimator for high-dimensional spaces. Future research will involve adapting this estimator to large transformer models, allowing us to assess the uncertainty of their predictions effectively and at scale. This will be critical to their use in practical applications, whether it is for more efficient exploration of protein design spaces or to enable their use in sensitive contexts such as disease prediction.

Self-driving laboratories. Finally, we are witnessing the emergence of self-driving laboratories [290, 291], which represent an exciting frontier in the field of ML-driven computational biology. These laboratories are built on a sophisticated integration of uncertainty-aware models with experimental processes. This leads to systems that can autonomously design, carry out, and learn from their experiments, drastically reducing the time required to test hypotheses or develop new products.

As generative models continue to evolve and improve, they are poised to radically transform the field of computational biology, by allowing us to seamlessly transition between genotype and phenotype via an intuitive interface, likely natural language [289, 292]. It holds the potential to transform modern medicine via personalized treatments, predict the course of viral evolution in future pandemics, and design the novel biomolecules that will help us address key challenges in healthcare, agriculture, and sustainability.

"Imagination will often carry us to worlds that never were. But without it we go nowhere." — Carl Sagan.

Appendices

A

Appendix to Chapter 3 – Represent

Contents

A.1 EVE model details

A.1.1 Scope and model training data

We focus on genes known to be involved in one or several diseases, that we define as the set of genes in ClinVar with at least one missense variant labeled as Pathogenic. There are 3,851 such genes in ClinVar (as per the April 2021 release). To each gene we associate a single protein, by picking the canonical transcript according to Uniprot/Swissprot [87]. To train EVE, we build multiple sequence alignments for each protein family by performing five search iterations of the profile HMM homology search tool jackhmmer [92] against the UniRef100 database of non-redundant protein sequences [162], downloaded on April 20th 2020. Following the protocol of Hopf et al. [38] and Riesselman et al. [107], we retrieve sequences that align to at least 50% of the target protein sequence, and columns with at least 70% residue occupancy. The software used is publicly available on GitHub at the following address: `https://github.com/debbiemarkslab/EVcouplings`.

We explore a range of bit score thresholds, using 0.3 bits per residue as a reference, and select the best possible multiple sequence alignment based on the criteria of maximal coverage of the target protein sequence and sufficient, but not excessive, number of sequences in the alignment (the latter implying an alignment that is too lenient). Specifically, we prioritize alignments with coverage $L_{\text{cov}} \geq 0.8L$, where L is the length of the target protein sequence, and with a total number of sequences N such that $100,000 \geq N \geq 10L$. If these requirements cannot be met, we sequentially relax them down to $L_{\text{cov}} \geq 0.7L$ and $N \leq 200,000$. These criteria are met for 97% of alignments. For the remaining 3%, we drop the coverage constraint entirely. Following this procedure, we have so far obtained a set of 3,219 clinically relevant proteins with corresponding evolutionary training data.

A.1.2 Model architecture

We perform several ablations to optimize the underlying model architecture and the choice of training hyperparameters (Fig. 3.1). The main changes we propose

over the DeepSequence architecture [107] are as follows:

- Symmetrization of the encoder and decoder network architectures;
- Increased number of layers and increased layer width for the encoder and decoder networks ($2,000$ - $1,000$ - 300 and 300 - $1,000$ - $2,000$ respectively);
- Increased size of the latent space (50);
- Larger number of training steps to train the more complex architecture ($400,000$ training steps);
- Lower learning rate to stabilize learning process (10^{-4});
- Removal of the group sparsity priors, responsible of significant performance drops for certain proteins.

We summarize in Fig. A.1 the performance gains achieved with the changes above compared with DeepSequence, by comparing Spearman correlation of the two models with the output of 38 different benchmark MAVEs, following the same protocol as described in Riesselman et al. [107].

A.1.3 Codebase

We have made the EVE codebase fully open source and uploaded it to the following GitHub repository: `https://github.com/OATML-Markslab/EVE`.

A.2 Tranception model details

A.2.1 Model architecture

Tranception is an autoregressive transformer architecture designed to explicitly promote head specialization and extraction of contiguous protein subsequence patterns, building on ideas introduced in Primer [123] and Inception [121]. We performed thorough ablations when developing Tranception and summarize the main variants tested in Table A.1. Our largest transformer model, Tranception L, has 700M parameters and is trained on UniRef100 [124]. In early iterations we also

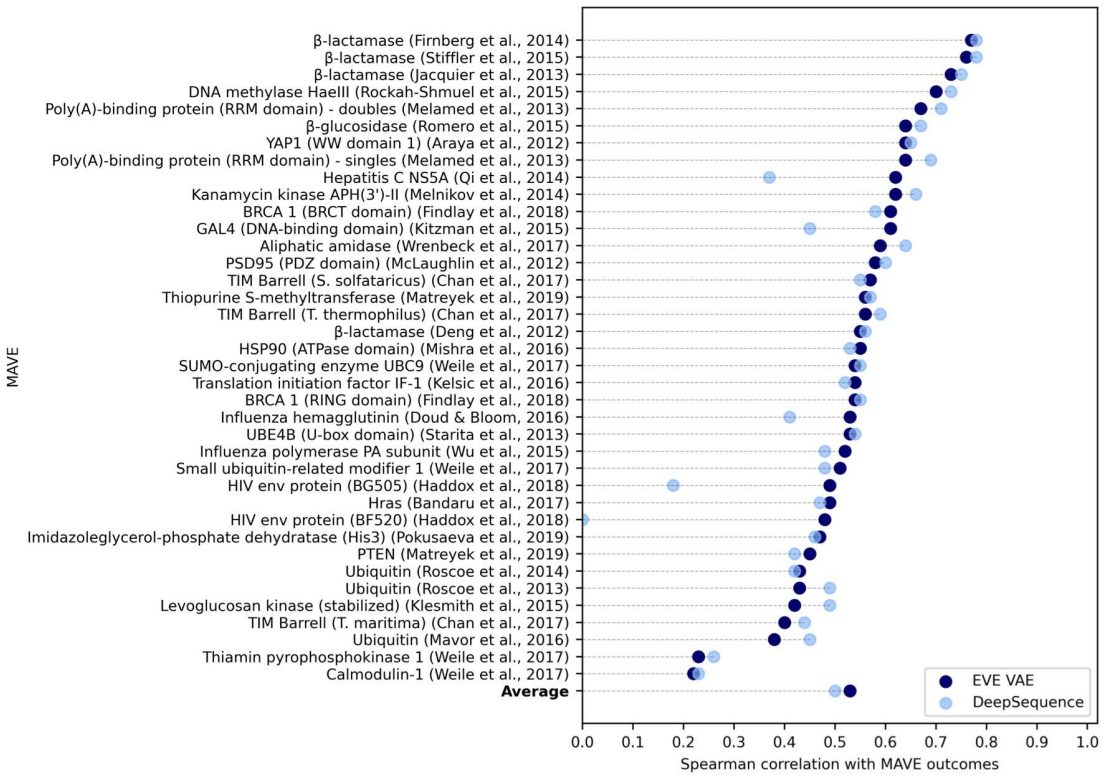

Figure A.1: Comparison of performance at mutation effect prediction of our Bayesian VAE implementation and DeepSequence. Comparison between the performance of the Bayesian VAE architecture in EVE and the one from DeepSequence [107] which achieves state-of-the-art performance on the protein function prediction task. The spearman correlations between model and experiment were evaluated for 38 multiplexed assays of variant effects. "Evolutionary indices" were computed using the same protocol as Riesselman et al. for both models, i.e. by sampling 2k times from the approximate posterior distribution and by ensembling the obtained indices over 5 independently trained VAEs.

experimented training our architecture on UniRef90 and UniRef50, clustered versions of UniRef100 at 90% and 50% similarity levels respectively, but observed superior performance from training on UniRef100 (see Appendix A.2.3 and Table A.1).

Hyperparameter	**GPT2 S**	**Primer S**	**Tranception LS**	**Tranception S**	**Tranception M**	**Tranception L**
Parameters	85M	85M	85M	85M	300M	700M
Attention heads	12	12	12	12	16	20
Layers	12	12	12	12	24	36
Embedding size	768	768	768	768	1,024	1,280
Activation function	GELU	Squared ReLU	Squared ReLU	Squared ReLU	Squared ReLU	Squared ReLU
Position encoding	Learned embedding	Learned embedding	Learned embedding	Grouped ALiBi	Grouped ALiBi	Grouped ALiBi

Table A.1: Characteristics of different model variants used in ablations. All models had a max context length of 1024 tokens, and we use the default dropout value of 0.1 in all variants. Tranception LS differs from Tranception S by the use of learned position embeddings instead of Grouped ALiBi.

To decide between different architecture options while not overfitting these decisions to our benchmark, we selected a small yet representative subset of DMS assays in the ProteinGym substitution benchmark (10 out of 87 substitution DMS assays):

- BLAT ECOLX [293]
- CALM1 HUMAN [294]
- CCDB ECOLI [295]
- DLG4 RAT [296]
- PA I34A1 [297]
- Q2N0S5 9HIV1 [197]
- RL401 YEAST [298]
- SPG1 STRSG [299]
- SPIKE SARS2 [190]
- TPOR HUMAN [300]

Together, these 10 assays cover the different taxa (3 viral proteins, 4 human and other eukaryote proteins, 3 prokaryote proteins), mutation depths groupings (3 low, 4 medium and 3 high as per the classification described in Table 4.2) and include one assay with multiple mutants (which matches the overall proportion of assays with multiple mutants within ProteinGym). Downstream performance of the different ablations on this validation set, and the overall substitution set are reported in Table A.2.

Model variant	Training data	Position encoding	Spearman validation set	Spearman full set
GPT2 S	Uniref100	Learned embedding	0.324	0.320
Primer S	Uniref100	Learned embedding	0.314	0.315
Tranception LS	Uniref100	Learned embedding	0.330	0.333
Tranception S	Uniref100	Grouped ALiBi	0.344	0.335
Tranception S	Uniref90	Grouped ALiBi	0.264	0.275
Tranception S	Uniref50	Grouped ALiBi	0.248	0.247
Tranception M	Uniref100	Grouped ALiBi	0.358	0.376
Tranception L	Uniref100	Grouped ALiBi	**0.399**	**0.404**

Table A.2: Performance of the different model variants in ablation studies. Performance is measured via Spearman's rank correlation ρ between model scores and experimental measurements, following the approach discussed in B.1. Retrieval inference is excluded from this analysis. Model selection is performed on the validation set described in Appendix A.2.1.

A.2.2 Data processing

Except for the two ablations focusing on UniRef50 and UniRef90, all models are trained on UniRef100. We perform very mild filtering steps of the data to remove fragments and low quality sequences, and preserve as much sequence diversity as possible. For each UniRef100 sequence cluster, we map the corresponding UniRef50 cluster which pools together sequences within 50% similarity from one another [1]. We use 99% of the data ($\sim$ 249 million sequences) for training and set aside 1% of the data for validation ($\sim$ 2.5 million sequences). All singletons at the UniRef50 cluster level are removed (eg., isolated fragments). We further exclude from the training and validation datasets all sequences that contained the infrequent Pyrrolysine (O) or Selenocysteine (U) amino acids, or with two or more consecutive indeterminate amino acids X to remove lower quality sequences. The remaining indeterminate amino acids (X, B, J, Z) are kept at train time and randomly imputed as follows: X is imputed to any of the 20 standard amino acids, B to either D (Aspartic acid) or N (Asparagine), J to either I (Isoleucine) or L (Leucine), Z to either E (Glutamic acid) or Q (Glutamine). All sequences with indeterminates are excluded from the

[1] More precisely, UniRef100 is first clustered at the 90% identity to generate UniRef90. Cluster representatives in UniRef90 are then clustered at 50% identity to yield UniRef50.

Metric	Value
Number of sequences	249M
Max sequence length	40,921
95^{th} percentile of length	939
75^{th} percentile of length	470
Median sequence length	314
25^{th} percentile of length	198
5^{th} percentile of length	92
Min sequence length	12

Table A.3: High level statistics of protein sequences in UniRef100 after preprocessing. About 98% of sequences in UniRef100 have length lower than 1,024.

Hyperparameter	Value
Training steps	150k
Batch size	1,024
Peak learning rate	$3 * 10^{-4}$
Weight decay	10^{-4}
Optimizer	AdamW

Table A.4: Model training hyperparameters.

validation set. Table A.3 recapitulates key statistics of sequences in UniRef100 after applying these filtering criteria. The observed distribution of sequences guided our choice for the maximum context length of 1,024 tokens for our transformer models, as it allows to handle 98% of protein sequences in UniRef100 without truncation.

A.2.3 Model training

All model variants are trained for 150k steps, with a batch size of 1,024 sequences. During training, we reverse sequences at random and truncate sequences if longer than the maximum context size as per the scoring scheme details described in Appendix A.2.4. We train with the AdamW optimizer [301], with a learning rate schedule annealed over the first 10k steps up to the maximum value ($3 * 10^{-4}$), and then linearly decreased until the end of training. Other training hyperparameters are summarized in Table A.4. In terms of computing resources, small architectures are trained on 8 V100 GPUs for ∼1 week, medium architectures with 32 V100 GPUs for ∼1 week, and our largest model, Tranception L, is trained on 64 A100 GPUs for ∼2 weeks.

We provide the training loss curves of the different architecture variants in Fig. 3.3 and Fig. A.2. While trained beyond the optimal number of steps for transformer models used in NLP tasks [302], our larger networks still appear to

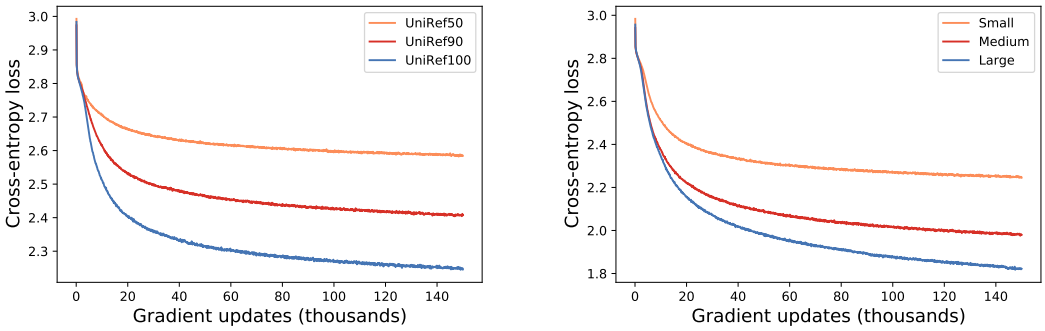

Figure A.2: Cross-entropy loss Vs number of gradient steps Left: Tranception S architecture trained on different UniRef datasets (UniRef50, UniRef90, UniRef100). Right: Small (S), Medium (M) or Large (L) Tranception architectures trained on UniRef100.

be undertrained. We note that cross-entropy is not necessarily a good indicator of downstream performance when comparing datasets with very different characteristics. For instance, UniRef50 is by design less redundant than UniRef100, thus we expect the loss to be typically lower on the latter (as can be seen in Fig.A.2). It so happens that, for the particular task we are interested in (fitness prediction), the more granular UniRef100 both leads to lower cross-entropy loss and higher downstream task performance (Table A.2), hence we trained our larger models on that dataset.

A.2.4 Scoring protein sequences

Mirrored sequences While protein sequences are not strictly invariant to mirroring (ie., the same sequence of amino acids assembled in the left to right order may not lead to the same 3D structure if assembled from right to left), prior autoregressive transformer architectures [118] have proposed to augment their training dataset by including all sequences and their reverse. In this work we further investigate the benefits of using each sequence and its reverse *at inference* when predicting fitness (§ 4.2.3). To remove potential discrepancies between training and inference, we apply a data augmentation similar to that of Madani et al. [118] at training time and randomly reverse a subset of sequences in each batch (each sequence in the batch is flipped with a probability 0.5). We find that averaging the log-likelihood ratios obtained by scoring each sequence and its mirror image leads to superior downstream task performance, with or without retrieval (Table A.5).

DMS set	Model	Unidirectional scoring	Bidirectional scoring
Validation set	Tranception (w/o retrieval)	0.376	0.399
	Tranception (w/ retrieval)	0.447	0.452
Full set	Tranception (w/o retrieval)	0.376	0.401
	Tranception (w/ retrieval)	0.432	0.445

Table A.5: Comparison of unidirectional Vs bidirectional scoring Performance is measured via Spearman's rank correlation ρ between model scores and experimental measurements. Analysis is performed with Tranception L (with and without retrieval), and scores are reported on both the validation DMS set and the full DMS set. The bidirectional scoring is the average of log-likelihood scores obtained by traversing the sequence in the canonical direction (left to right) and the reverse direction (right to left).

Scoring window When scoring protein sequences that are longer than the maximum context length of the model (1,024 amino acids in our model), we have the choice to either truncate the sequence or combine predictions from slided (overlapping or non-overlapping) windows of the sequence. We observe very little performance difference between the different approaches in terms of average Spearman's rank correlation with validation DMS assays measurements, and therefore leveraged the former for simplicity. For single mutations, the optimal scoring window is selected so as to maximize the context available around that mutation for prediction when scoring the sequence from left to right and right to left. When scoring multiple mutants, we applied a very similar approach, maximizing context around the barycenter of the various mutant positions.

A.2.5 Retrieval at inference

Augmenting the autoregressive predictions at each position via retrieval inference only occurs at the positions covered (even partially) by the MSA – outside of theses positions, we fully rely on autoregressive predictions. We compute the pseudocounts at each position of the alignment via weighted Laplace smoothing [131], with a small smoothing parameter (10^{-5}). As discussed in § 3.4.2, sequence are weighted as per the procedure described in [106] and we fully ignore gaps in the MSA when computing the pseudocounts.

We optimize the retrieval inference weight α from equation 3.6 via linearly-spaced grid search between 0.0 (no retrieval) and 1.0 (full retrieval on covered positions,

DMS set	Retrieval inference weight α										
	0.0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
Validation set	0.399	0.412	0.424	0.435	0.444	0.450	**0.452**	0.449	0.443	0.431	0.397
Full set	0.401	0.410	0.419	0.428	0.435	0.441	0.445	**0.446**	0.442	0.432	0.404

Table A.6: Retrieval inference weight optimization. We perform a linearly-spaced grid search for α on our validation DMS set and obtain an optimal rate of 0.6.

autoregressive only otherwise). As per the results in table A.6, the optimal α value on the validation DMS set is 0.6. Except for the analysis discussed in this section, whenever retrieval is used in this paper,it is with this 0.6 retrieval inference weight.

A.2.6 MSA Filtering analyses

We report additional DMS level results for the MSA filtering analysis described in § 3.4.3 in Fig. A.3. We observe that the performance of Tranception is less sensitive to MSA depth compared to EVE or MSA Transformer.

Codebase

We have made the Tranception codebase fully open source and uploaded it to the following GitHub repository: `https://github.com/OATML-Markslab/Tranception`.

A.3 TranceptEVE model details

A.3.1 Inference details

We chose Tranception and EVE as they are respectively the best family-agnostic and family-specific mutation effect predictors at the time of developing the corresponding models (based on correlation with experimental results from the ProteinGym benchmarks). Furthermore, in early experiments, these two models showed the best complementary when performing naive ensembling between models (testing all possible pairs from baselines in Notin et al. [303]).

We obtain the 'EVE log prior' as follows:

- We train an EVE model (or, in the case of the ProteinGym analyses, an ensemble of 5 EVE models with different random initializations) on an MSA

retrieved from a wild type sequence that is representative of the protein family of interest (e.g., canonical sequence in Uniprot);

- We encode that same wild type sequence in the latent space of the trained bayesian VAE from EVE, and then decode it a large number of times (200k samples);
- We average the resulting log softmax probabilities (i.e., the decoder outputs) across samples;
- If we have trained an ensemble of 5 EVE models, we also average across the models in the ensemble;
- The resulting tensor represents a log probability over the amino acid vocabulary as each position in the sequence (i.e., $\log P_E(x)$), akin to the empirical distributions obtained in the MSA retrieval of Tranception (i.e., $\log P_M(x)$).

Similarly to what we do in Tranception (Equation 3.4), we score sequences from both directions (i.e., N→C and C→N), then take the arithmetic average of each log probability:

$$\log P(\mathbf{x}) = \frac{1}{2}[\log P(\mathbf{x}_{N\to C}) + \log P(\mathbf{x}_{C\to N})] \tag{A.1}$$

A.3.2 Aggregation coefficients based on MSA depth

Instead of the constant aggregation coefficient used in Tranception, we use aggregation coefficients (i.e., α and β in Equation 3.4.4) which depends on the depth of the retrieved MSA for a given protein family (see Table A.7). This enables TranceptEVE to seamlessly adapt to all possible protein families by leaning more heavily on the most relevant mode of inference depending on the situation: if the protein family of interest has no or very few homologs, we rely exclusively on the autoregressive transformer; if the MSA is deeper, we give more importance to the MSA and EVE log priors.

	MSA depth (Nb. sequences)				
	< 10	$< 10^2$	$< 10^3$	$< 10^5$	$\geq 10^5$
α	0.0	0.3	0.6	0.7	0.8
β	0.0	0.1	0.3	0.4	0.5

Table A.7: Aggregation coefficient for EVE log prior (α) and MSA log prior (β) based on MSA depth.

A.3.3 Indels scoring details

When scoring insertions & deletions ('indels'), we adapt the log prior distributions by re-aligning the protein to be scored to the family-specific MSA: deletions in the protein sequence translate to deletions of the corresponding positions in the priors; insertions lead to the addition of dummy columns at these positions in the priors, which are then ignored in the final weighted average (i.e., in Equation 3.4.4) such that the predictions at these positions rely solely on the autoregressive transformer.

A.3.4 Recalibrating models probabilities

Since the transformer in Tranception and the VAE in EVE are trained independently on different data domains, their output probabilities are not identically calibrated. After factoring in the relative weights with respect to MSA depth as discussed in Appendix A.3.2, we would like each model to have the same overall importance in the predictions from Equation 3.4.4. We thus iteratively recalibrate the EVE log probabilities via temperature scaling so that the two models have the same mean output log softmax when given the wild type sequence as input.

A.3.5 Scope of the EVE log prior

In the original EVE architecture [135], we were only leveraging positions that were sufficiently-covered in the MSA. As a result, the corresponding EVE models were only able to score mutations at these well-covered positions. There is however no strict constraint to model only well-covered positions, and we have observed in practice that also including the non well-covered positions in the EVE models was 1) not detrimental to the predictive performance when scoring mutations at

well-covered positions only (e.g., same overall performance on the ProteinGym benchmark) 2) would allow to score mutations at non well-covered positions as well. In TranceptEVE, we therefore by default leverage EVE models modeling all positions in the alignment and report performance in § 4.2 accordingly. However, the TranceptEVE codebase fully adapts to the characteristics of the underlying EVE model if one would rather train the EVE model on well-covered positions only: if the EVE model was trained on well-covered positions only, it bases its predictions for the non well-covered positions on the autoregressive transformer and retrieved MSA only (as in Tranception with retrieval); otherwise it leverages the autoregressive transformer and the two log priors.

A.3.6 Comparison with standard model ensembling

The aggregation described in Equation 3.4.4 can be perceived as performing something analogous to model ensembling between Tranception and EVE (we do not perform a strict 'model ensembling' since the EVE log prior is independent from the particular sequences to score and is instead protein-family specific). While we observe comparable overall performance between our TranceptEVE scheme and a standard ensembling of Tranception and EVE (e.g., average of standardized log-likelihood ratios from Tranception and standardized delta ELBOs from EVE) on the ProteinGym substitution benchmark (eg., same average Spearman and AUC), our proposed scheme however presents several significant advantages:

- The TranceptEVE scoring scheme preserves autoregressiveness during decoding, which allows it to support novel sequence generation for protein design (unlike with standard ensembling where we only obtain a single score for the full sequence with the delta ELBOs output by EVE);

- TranceptEVE can be used to score indels as per the extensions described in Appendix A.3.3 (unlike naive ensembling since EVE is unable to score insertions or deletions given its fixed-size MLP encoder);

- Standard ensembling of Tranception and EVE requires the scores to be standard scaled before averaging (since the Tranception and EVE models scores are on different scales). This operation requires scores from both models for a representative set of sequences (e.g., scores from all singles) to compute the relevant score statistics (i.e., mean and variance). It is not clear which representative set to use in practice (e.g., should we use the same set for mutations at different mutation depths?) and represents a potentially large computational overhead. TranceptEVE is not impacted by any of these issues;
- In TranceptEVE we only score the wild type sequence with EVE and use the same EVE log prior to score all mutated sequences of interest. With standard ensembling we need to score each sequence with both Tranception and EVE.

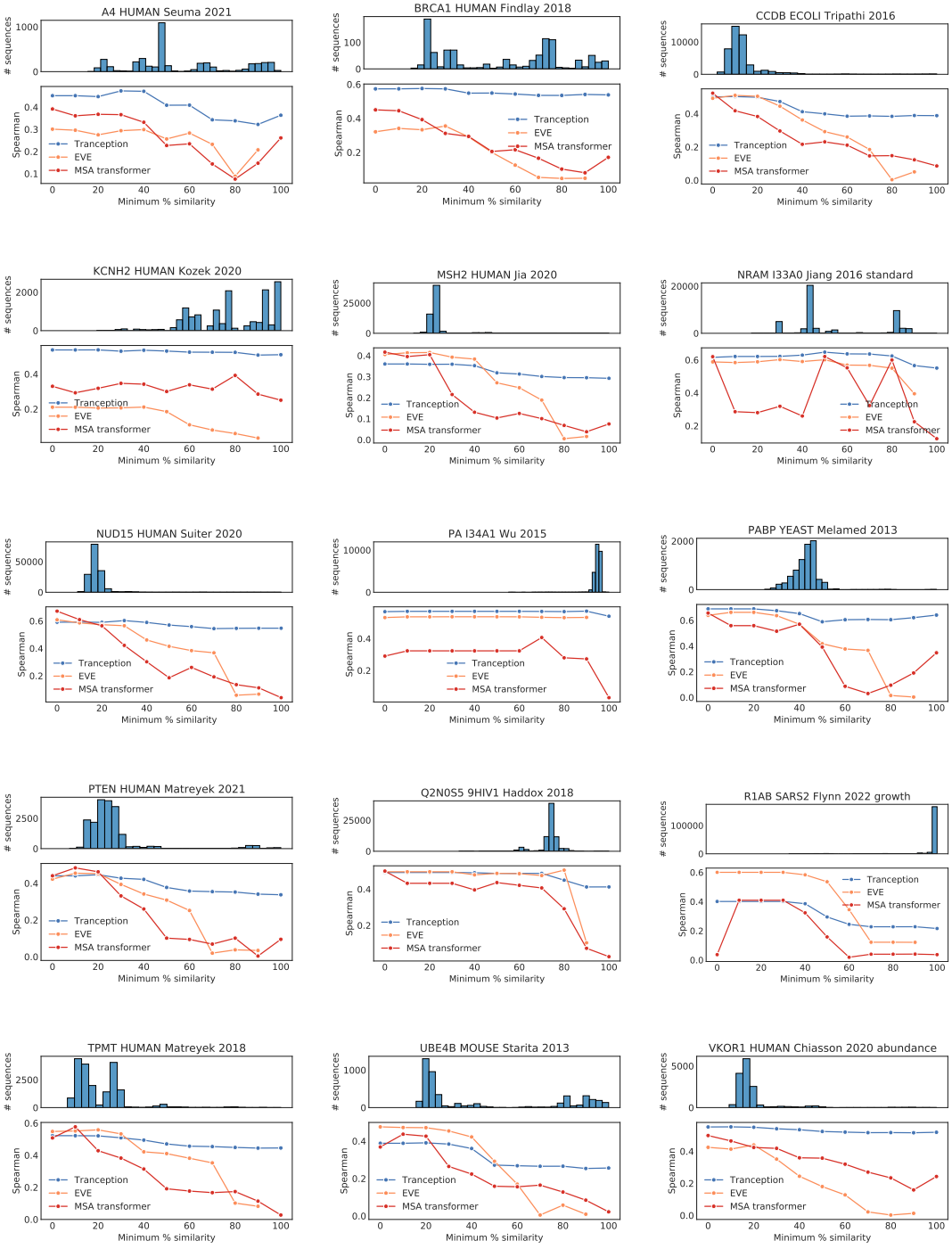

Figure A.3: Robustness to alignment depth for several DMS assays in the ProteinGym substitution benchmark. We measure the Spearman's rank correlation between model score and experimental measurement for Tranception, EVE and MSA Transformer as we progressively exclude sequences in the corresponding MSAs based on their similarity to the sees sequence used to create the alignment.

B

Appendix to Chapter 4 – Predict

Contents

B.1 Protein fitness prediction

B.1.1 ProteinGym – DMS assays curation

To build the ProteinGym benchmark, we initially collected a set of 137 deep mutational scanning assays. We then filtered out 43 of these assays based on the

following criteria: Data had not been made publicly available (9), non-protein assays (UTR, tRNA, promoter, etc.; 7), synthetic proteins (3), insufficient number of measurements (3), outdated (ie., a more recent improved assay on the same protein and same property was found; 4), majority of data hitting experimental floor (6), low dynamic range (6), assay not relevant to fitness prediction (5).

The final version of the ProteinGym consists of the experimental measurements of 94 deep mutational scanning experiments (87 substitutions assays and 7 indels assays) from the following 77 publications: [132, 156, 164–167, 190, 197, 198, 201, 293–300, 304–360].

The ProteinGym benchmarks are made publicly available on our GitHub repository, and on our dedicated website.

B.1.2 ProteinGym – MSA creation

For every assay in ProteinGym, we build MSAs of the corresponding protein by performing five search iterations of the profile HMM homology search tool Jackhmmer [92] on the UniRef100 database of non-redundant proteins [124], downloaded on November 24 2021. We explore a range of 9 bit score thresholds, from 0.1 to 0.9. We subsequently select the alignment with the highest number of significant Evolutionary Couplings (ECs) [160]. To be consistent in our approach across all assays, we build a single MSA for each protein, and do not investigate domain-specific alignments where relevant. As noted in § 3.4.3, crafting domain-specific alignments, when these domains are known, may help further increase performance of the different models. Characteristics of the different MSAs (eg., number of sequences, selected bit score) we obtained with the above process are provided in the ProteinGym reference file available in our GitHub repository.

B.1.3 Zero-shot setting – Detailed performance analysis

Performance metrics

We report performance based on the 3 metrics described in § 4.2.2:

- **Spearman's rank correlation ρ between model scores and DMS experimental measurements.** Since certain DMS assays are relatively difficult to model resulting in very low (and sometimes negative) ρ values, we report the signed ρ value instead of the absolute ρ values reported in prior works [41, 107]. To that end, we adjust the signs of measured phenotypes where needed, to ensure consistency in the directionality across assays. In ProteinGym, a higher DMS score is always associated with higher fitness;
- **AUC between model scores and DMS experimental measurements.** This metric is particularly relevant when focusing on assays with a bimodal distribution of the measured phenotype. We binarize DMS measurements by setting the threshold manually when the assay is clearly bimodal and there is no ambiguity about the correct threshold value to select. We use a median cutoff in all other instances. We report the numerical value of the cutoff and the binarization method (median or manual) in the ProteinGym reference file (available in our GitHub repository);
- **Matthew's correlation coefficient (MCC) between model scores and DMS experimental measurements.** This complements the analysis obtained with AUC. DMS measurements are binarized with the same thresholds as for AUC. Predictions are binarized by using their median value as threshold.

DMS assays preprocessing

In order to standardize measured outcomes as much as possible across assays, we further preprocess the raw DMS measurements as follows:

- **Silent mutations**: certain assays include nucleotide substitutions resulting in silent mutations. We remove these from our benchmark;
- **Duplicate mutations**: certain assays include duplicate mutants – either nucleotide substitutions resulting in mutant repeats in the experiments, or indels resulting in identical protein sequences. We remove duplicates by averaging all DMS measurements across duplicate mutants;

- **Missing measurements**: mutants with missing assay measurement are dropped.

Finally, ProteinGym contains several DMS assays for the same protein, for a handful of proteins (eg., we include 4 assays for BLAT ECOLX, 4 assays for P53). While all different and important in their own right, these different assays have experimental measurements that are strongly correlated. Consequently, models that tend to do well for one of these assays relative to others, tend to perform well across all of the assays for the same protein. In order to remove the potential biases resulting from these correlated assays, we first aggregate all performance metrics at the Uniprot ID level and then compute the different average results in § 4.2.3 (ie., when there 4 DMS assays for the same protein, each of these assays carries a weight of 0.25 in the aggregate results).

Zero-shot baselines details

Except for the enhancements we discuss in this section, we use the official codebases for all baselines included in § 4.2.3. For the Site independent and EVmutation models we use the EVcouplings library. For Wavenet, we use the code made available in the SeqDesign repository. All alignment-based models are trained on the MSAs we obtain via the process discussed in Appendix B.1.2, and the same MSAs are used at inference time with MSA transformer or for retrieval in Tranception and TranceptEVE. Except for Wavenet, all family-specific models are unable to score indels since they rely on the fixed coordinate system from the original MSA they have been trained on.

EVE and DeepSequence For both DeepSequence and EVE, we use the Pytorch implementation available in the official EVE github repository (we use the optimal parameters for EVE as per Frazer et al. [135], and as per Riesselman et al. [107] for DeepSequence). EVE and DeepSequence models are trained on the protein-specific MSAs discussed in Appendix B.1.2. Sequences with more than 50% of gaps in the MSA are removed from training. In order to score all mutations in the ProteinGym benchmarks, we relax the constraint on minimum column coverage

from the original DeepSequence and EVE architectures and consider all residues during training and inference. As discussed in Appendix A.3.5, this helps increase scope with no impact on performance.

ESM-1b, ESM-1v and MSA Transformer For ESM-1v and MSA Transformer, we start from the official ESM codebase, and extend it as follows to support:

- the scoring of multiple mutants – as discussed in Meier et al. [41] this is achieved by independently summing the effects of each single mutations that comprise the multiple mutant;
- the scoring of sequences that are longer than the context size of the ESM-1v and MSA Transformer models. We leverage the same routine we developed for Tranception to select the optimal scoring window (see Appendix A.2.4);
- the weighted sampling of sequences of an input MSA in the MSA Transformer. We compute sequence weights with the same procedure used for retrieval inference in Tranception, ie. based on the procedure described in Hopf et al. [38].

When scoring with ESM-1v or MSA Transformer, we use the masked-marginals approach introduced in Meier et al. [41]. This scoring heuristics has two main limitations: 1) as discussed above, scores for multiple mutants are computed as the sum of the effects from the individual single mutants comprising the multiple mutation, which leads to ignoring potentially important epistatic effects 2) masked scoring implies a fixed frame of reference in which the mutated position exists in the original wild-type sequence. Consequently these two models are unable to score indels with the masked-marginals heuristic. We use the most recent checkpoints made publicly available for each model (resp. esm1v_t33_650M_UR90S_[1-5] and esm_msa1b_t12_100M_UR50S at the time of writing). For MSA transformer, we follow [41] and first filter sequences in the input MSA with HHFilter [361] to ensure minimum coverage of 75% and maximum sequence identity of 90%, and then sample 384 sequences (with the weights discussed above). For ESM-1b, we use the official ESM repository for model checkpoint and mutation effects predictions. We further

extend the codebase to handle sequences longer than the model context size (i.e., longer than 1,022 amino acids) as per the procedure described in Brandes et al. [136].

Other baselines We use the official github repositories for the implementations of Unirep [39], RITA [127] and Progen2 [126]. Evo-tuning for Unirep is performed on the relevant protein MSAs from ProteinGym. For all RITA and Progen2 model variants, we score sequences from both directions (similar to what is described in equation 3.4 for Tranception). For these two baselines, the ensemble version is obtained by ensembling the different sizes of the corresponding model (e.g., RITA S, RITA M, RITA L and RITA XL).

Detailed results

In addition to the results provided in § 4.2.3, we also report aggregated performance results by taxon (Spearman) in Table B.1. The conclusions from the analyses based on AUC and MCC are strictly identical to those based on Spearman's rank correlation, and we refer the reader to the ProteinGym codebase or the Tranception paper [137] for the corresponding results. The Spearman's rank correlation coefficients between model scores and DMS experimental measurements for Tranception, EVE, ESM-1v and MSA Transformer for each DMS assay in the substitution benchmark are shown in Fig. B.1. Detailed performance (Spearman) at the assay level for Tranception, TranceptEVE and the best baselines (Wavenet and RITA) on the ProteinGym indel benchmark are provided in Table B.2.

Statistical significance Given the high correlation between baselines across assays (e.g., certain assays are difficult to predict for all models, others easy for all models), we assess the statistical significance of the results reported in Table 4.2 via the bootstrap standard deviation of the *difference* between the best method (TranceptEVE) and other baselines (Table B.3).

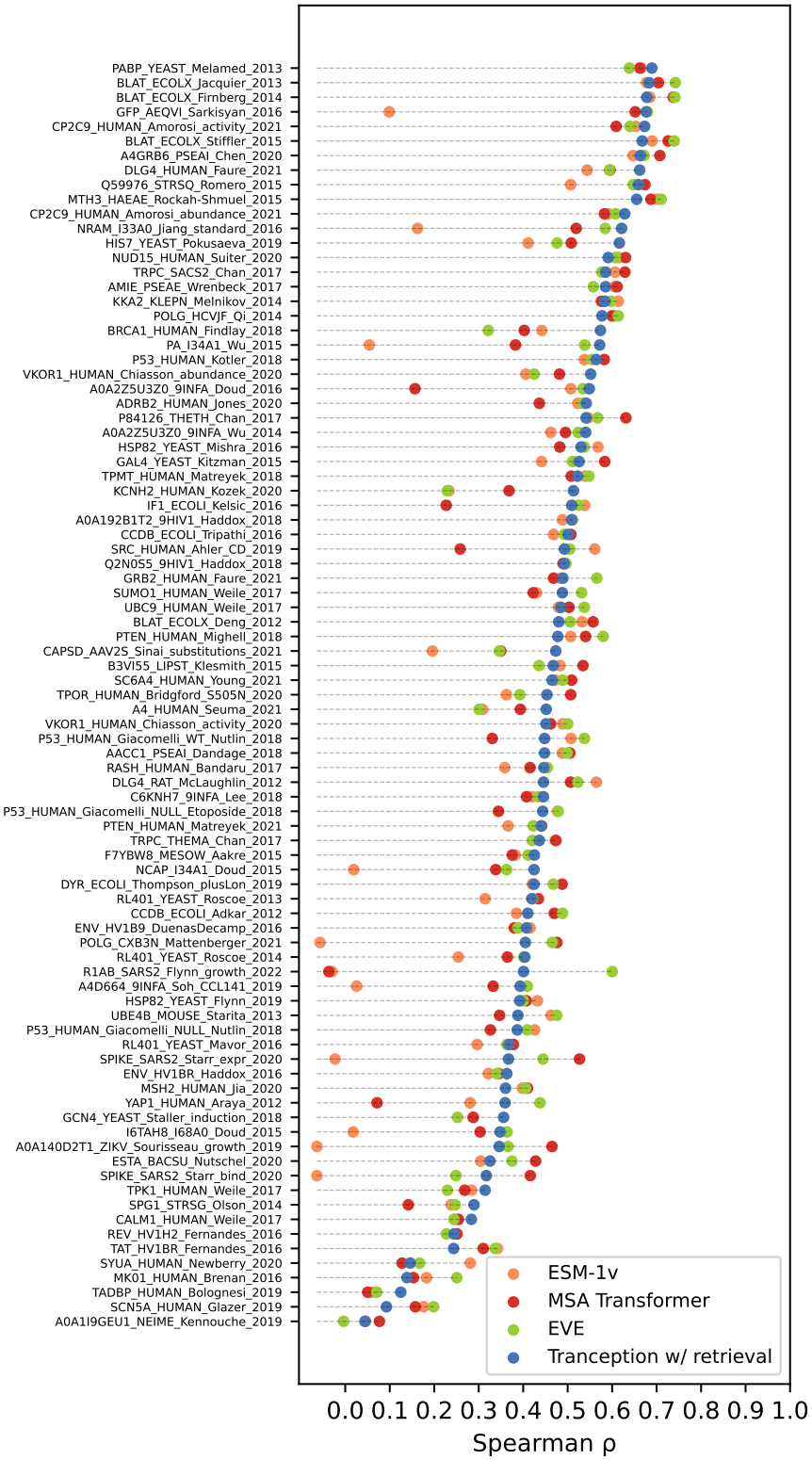

Figure B.1: Zero-shot model performance on the ProteinGym substitution benchmark. We report the DMS-level performance (measured by the Spearman's rank correlation ρ between model scores and experimental measurements) of Tranception with retrieval, ESM-1v [41], MSA Transformer [130] and EVE [135] for each DMS in the ProteinGym substitution benchmark.

Model type	Model name	Spearman by Taxon ↑ Human	Other eukar.	Prokaryote	Virus	All
Alignment-based models	Site independent	0.366	0.417	0.324	0.412	0.375
	Wavenet	0.378	0.446	0.472	0.308	0.391
	EVmutation	0.385	0.448	0.472	0.381	0.413
	DeepSequence (ens.)	0.400	0.493	0.492	0.348	0.421
	EVE (ens.)	0.408	0.499	0.500	0.435	0.449
Protein language models	Unirep	0.254	0.226	0.166	0.01	0.166
	ESM-1b	0.391	0.442	0.485	0.153	0.358
	RITA (ens.)	0.378	0.387	0.381	0.410	0.388
	ESM-1v (ens.)	0.425	0.441	0.502	0.257	0.401
	Progen2 (ens.)	0.399	0.459	0.462	0.364	0.413
	Tranception (w/o retrieval)	0.361	0.436	0.45	0.395	0.401
Hybrid models	Unirep (evotuned)	0.334	0.338	0.372	0.351	0.348
	MSA Transformer (ens.)	0.394	0.487	0.498	0.398	0.432
	Tranception	0.417	0.503	0.478	0.429	0.446
	TranceptEVE	**0.440**	**0.518**	**0.514**	**0.454**	**0.472**

Table B.1: Average Spearman's rank correlation ρ between model scores and experimental measurements by taxon.

DMS assay	Wavenet	RITA (ens.)	Tranception	TranceptEVE
A0A1J4YT16 9PROT [350]	0.117	0.159	**0.214**	0.212
B1LPA6 ECOSM [351]	0.385	0.359	0.416	**0.421**
BLAT ECOLX [352]	**0.546**	0.446	0.336	0.342
CAPSD AAV2S [353]	0.457	0.406	0.586	**0.594**
HIS7 YEAST [315]	0.680	0.678	0.692	**0.695**
P53 HUMAN [132]	0.001	**0.414**	0.398	0.399
PTEN HUMAN [166]	**0.699**	0.657	0.605	0.602
Average	0.412	0.446	0.464	**0.466**

Table B.2: Average Spearman's rank correlation ρ between model scores and experimental measurements on the ProteinGym indel benchmark.

B.1.4 ProteinNPT model details

PNPT architecture details

Model architecture. The ProteinNPT is a semi-supervised pseudo-generative model for protein property prediction, taking as input raw protein sequences and associated labels, as well as 'auxiliary labels' as described in § 4.2.4. Instead of learning a parametric model that makes protein property predictions based on the features of a single input protein sequence, our model makes predictions both based on these features but also based on similarity with a mini-batch of already labeled instances via self-attention. Using similarity between protein sequences is a

Model type	**Model name**	**Average Spearman**	**Standard error of diff. to TranceptEVE**
Alignment-based models	Site independent	0.375	0.012
	Wavenet	0.391	0.010
	EVmutation	0.413	0.005
	DeepSequence (ensemble)	0.421	0.009
	EVE (ensemble)	0.449	0.005
Protein language models	Unirep	0.166	0.024
	ESM-1b	0.358	0.019
	RITA (ensemble)	0.388	0.014
	ESM-1v (ensemble)	0.401	0.019
	Progen2 (ensemble)	0.413	0.011
	Tranception (w/o retrieval)	0.401	0.009
Hybrid models	Unirep (evotuned)	0.348	0.014
	MSA Transformer (ensemble)	0.432	0.012
	Tranception (w/ retrieval)	0.446	0.005
	TranceptEVE	**0.472**	-

Table B.3: Average Spearman's rank correlation between model scores and experimental measurements on the ProteinGym substitution benchmark, and standard error of the difference to TranceptEVE. The standard error reported is the non-parametric bootstrap standard error of the difference between the Spearman performance of TranceptEVE and that of a given baseline, computed over 10k bootstrap samples from the set of proteins in the ProteinGym substitution benchmark.

cornerstone of computational biology and protein modeling via the use of Multiple Sequence Alignments (MSAs). The MSA Transformer showed how self-attention across the sequences of an MSA is a very effective way to learn compact protein representations in an unsupervised setting, and the idea of performing self-attention across labeled instances has first been proposed by [139] – we discuss how the ProteinNPT relates to these two works in more details in Appendix B.1.4.

We summarize our architecture in Fig. 4.1. We embed the input mini-batch of protein sequences with a pre-trained protein language models. We experimented with various models in our ablations (see 'Ablations' below), obtained superior performance with the MSA Transformer and used this model in all subsequent analyses discussed in this work. Assuming the input sequence is of length L, the resulting embedded mini-bath of B sequences is of size $B.L.D$, where D is

the dimension for the embedding of a single amino acid token [1] . To prevent overfitting when training a ProteinNPT model on a relatively small labeled dataset, the embedding dimension d used in subsequent self-attention layers of the network is smaller than D (e.g., $D = 768$ for the MSA Transformer, and $d = 200$ in our final architecture based on ablations). We apply linear projections to reduce dimensionality accordingly.

In what follows, we assume the input target labels and auxiliary labels are both unidimensional continuous inputs, but everything seamlessly extends to continuous or categorical inputs of arbitrary dimensions (our codebase handles both input types). In our final ProteinNPT architecture the auxiliary labels are zero-shot fitness predictions obtained with the MSA Transformer, following the procedure described in Meier et al. [41]. To disambiguate masked entries from true input zeros, we append to continuous vectors a binary flag indicating whether the corresponding label is masked (for categorical input, a separate category corresponds to masked targets). After applying standard scaling on the input targets, we use linear layers (a separate one for each target and each auxiliary label) to project them into a d-dimensional vector. We then concatenate the protein sequence embeddings with the input target embeddings and auxiliary label embeddings.

We also experiment with various transforms of the resulting embeddings: Convolutional layer (CNN), ConvBERT layer [362], or no transform. We find in our ablations that applying a CNN layer delivers the best performance on downstream tasks performance.

The transformed embeddings are subsequently fed into 5 consecutive ProteinNPT layers (value chosen based on ablations). Each ProteinNPT layer applies successively row-attention, column-attention and a feedforward layer. Each of these transforms is preceded by a LayerNorm operator and we add residual connections to the output of each step. As proposed in the MSA Transformer, we leverage tied row attention,

[1] Depending on the underlying pre-trained protein language models used, additional beginning of sequence (BOS) and end of sequence (EOS) tokens are also added in practice. For instance, in the case of the MSA Transformer, a BOS token is prepended to the beginning of the input sequence. The size of the embedded sequence is thus $B.(L+1).D$

Table B.4: ProteinNPT - Architecture details

Hyperparameter	Value
Nb. ProteinNPT layers	5
Embedding dimension (d)	200
Feedforward embedding dim.	400
Nb. attention heads	4
CNN kernel size (post embedding)	7
Weight decay	5.10^{-3}
Dropout	0.0

Table B.5: ProteinNPT - Training details

Hyperparameter	Value
Training steps	10k
Learning rate warmup steps	100
Peak learning rate	3.10^{-4}
Optimizer	AdamW
Gradient clipping norm	1.0
Learning rate schedule	Cosine
Training batch (masked)	64
Training batch (unmasked)	361

ie., the attention maps in row attention are averaged across rows, resulting in lower memory footprint. This is a sensible design for the applications studied in our work, in which labelled sequences are relatively similar to one another and share a common three-dimensional contact structure.

Finally, we make predictions for the targets of interests by feeding the corresponding target embeddings into a L2-penalized linear projector (a separate linear map is learned for each target). We summarize all architecture details from our final architecture in Table B.4.

Ablations We carried out thorough ablations to develop the ProteinNPT. Following [41, 137], to support theses various ablations, we set aside a relatively small set of DMS assays (8 assays out of the 100 assays in the extended ProteinGym substitution benchmark), which are used to decide between different model architectures while not overfitting these decisions to the benchmark:

Table B.6: ProteinNPT ablations - Use of auxiliary labels Spearman's rank correlation between model predictions and experimental measurements. The auxiliary label used here is the zero-shot fitness prediction with the MSA Transformer.

CV scheme	No aux. label	With aux. label
Random	0.683	**0.684**
Modulo	0.527	**0.531**
Contiguous	0.439	**0.501**
Average	0.549	**0.572**

Table B.7: ProteinNPT ablations - Protein sequence embeddings Spearman's rank correlation between model predictions and experimental measurements.

CV scheme	Tranception	ESM1v	MSAT
Random	0.609	0.659	**0.684**
Modulo	0.416	0.468	**0.531**
Contiguous	0.370	0.380	**0.501**
Average	0.465	0.502	**0.572**

- BLAT ECOLX [293]
- CALM1 HUMAN [294]
- DYR ECOLI [341]
- DLG4 RAT [296]
- P53 HUMAN [165]
- REV HV1H2 [306]
- RL401 YEAST [298]
- TAT HV1BR [306]

Together, these 8 assays cover diverse protein families in terms of taxa, mutation depths, sequence lengths and MSA depth.

We report results for these ablations in this subsection, in particular varying the model used to obtain protein sequence embeddings (Table B.7), the use of auxiliary labels (Table B.6), and the number of training points used at inference (Table B.8). Based on these analyses, we use for our final model architecture the MSA Transformer [130] to obtain protein sequence embeddings, use auxiliary labels (ie. zero-shot fitness predictions with MSA Transformer), and use batches with 1k training instances to make predictions at inference.

Model training At train time, we feed as input to our model a batch of 425 protein sequences and the corresponding labels. Since we mask at random 15% of labels, this equates to having on average about 64 masked training instances – which

Table B.8: ProteinNPT ablations - Nb. labelled sequences used at inference Spearman's rank correlation between model predictions and experimental measurements.

CV scheme	Nb. labelled sequences sampled at inference 0	100	200	500	1000	2000
Random	0.398	0.677	0.678	0.679	0.684	0.685
Modulo	0.299	0.533	0.531	0.531	0.531	0.531
Contiguous	0.254	0.496	0.504	0.502	0.501	0.500
Average	0.317	0.569	0.571	0.571	**0.572**	**0.572**

we want to accurately predict the values of, and about 361 unmasked instances – which are used to support the predictions of masked instances. We also mask at random 15% of input protein sequence tokens. As discussed in § 4.2.4, our training loss is the weighted average of two different losses: the Cross-Entropy (CE) loss over masked tokens predictions (akin to a standard masked-language modeling objective [37]) and the Mean Squared Error (MSE) over the masked targets predictions. Throughout training, we vary the weight used to balance out these two losses, starting with equal weights at the beginning of training, and annealing progressively the amino acid token prediction objective with a cosine schedule over the course of training.

We train our model with the AdamW optimizer [301]. We experimented with various early stopping criteria to train our model by setting aside a validation set, monitoring the validation MSE or Spearman's rank correlation between prediction and targets, varying the patience parameter (e.g., 3, 5 and 10) but obtained systematically better performance on the downstream tasks from our 'validation benchmark' when instead training for a fixed number of training steps (10k gradient steps). All training hyperparameters are summarized in Table B.5.

Lastly, before training a ProteinNPT model on a given assay, we compute and persist to disk the sequence embeddings for all mutated proteins in that assay. During training, we load from disk the embeddings corresponding to each mini batch. Since the parameters of the model used to extract sequence embeddings are frozen, this procedure is strictly equivalent to computing embeddings on the fly. However, training is much faster as embeddings are computed only once and

loading from disk is relatively fast. Additionally, we can work with larger training batches as memory requirements are lower – computing embeddings with large protein language models can necessitate substantial amounts of GPU memory.

Inference During the inference process, our objective is to make predictions for unlabeled protein sequences. These sequences are handled in our model using the same approach as we employed for sequences with masked targets during training. The input mini-batch is also comprised of labelled sequences from the training set, which are used to support the predictions of unlabelled instances. Ideally we would fit the entire training set when making predictions. However, this approach is hindered by GPU memory bottlenecks in practice when the training set is too large. To that end, we set an upper bound M on the maximum number of training sequences we can leverage at inference time: if the training dataset is smaller than M, we use all training sequences with no filtering; if the training dataset is larger, we sample at random M sequences with no replacement. As seen in ablations above, we choose a value of $M = 1k$ in practice as larger values only yield marginal gains on downstream tasks performance. We also experimented with a sampling scheme with replacement in earlier ablations, but obtained systematically comparable performance with the scheme described above. We note that in the case where the number of training points used at inference is set to zero, the ProteinNPT does not make use of column attention and reduces to standard parametric transformer. In that case, it is analogous to the recently introduced Regression Transformer [363], which can be seen as a special case of our architecture.

Difference with axial transformer, MSA Transformer and Non-parametric transformers

Our work builds on prior work from the self-attention literature, in particular the Axial Transformer [54], MSA Transformer [130] and Non-Parametric Transformer [139].

Focusing on image and video modeling tasks, the Axial Transformer introduced the concept of axial attention, alternating in a given self-attention layer between row-wise attention and column-wise attention. We use this concept throughout our

architecture to operate self-attention across residues (in embedding layers), across residues and labels (in ProteinNPT layers), across sequences of a retrieved MSA (in embeddings layers) and across labelled observations (in ProteinNPT layers).

The MSA Transformer leveraged axial attention to learn protein sequence representations by training across a large number of Multiple Sequence Alignments (MSAs). Axial attention is used to alternatively learn dependencies across residues in the protein sequences, and across MSA sequences at a given position. The MSA Transformer is a fully unsupervised architecture and does not leverage available labels. Authors also show that a regression trained on attention maps of the MSA transformer achieve strong performance in predicting the residues that are in contact in the tertiary structure of the protein. It is an integral part of our ProteinNPT architecture, allowing us to obtain high-quality embeddings for each residue, which are then leveraged in the subsequent ProteinNPT layers.

Focusing on tabular datasets, Non-Parametric transformers (NPTs) leveraged axial attention to learn dependencies across features and labels of the tabular input, and across labelled instances. ProteinNPT differs from and extends NPTs in the following ways:

- In lieu of tabular inputs, we focus on a **different data modality** (protein sequences);
- Given the scarcity of labels involved with respect to the complexities of the underlying objects and their relations with the targets of interest, our architecture is **semi-supervised** and we leverage protein embeddings pre-trained on large quantities of unlabelled natural sequences;
- In our final architecture, we operate **tri-axial** self-attention as described above (instead of bi-axial self-attention in MSA Transformer and in standard NPTs);
- We introduce the concept of **auxiliary labels** which is critical for us to achieve strong performance, but it also broadly applicable beyond the protein-specific use cases we focus on in this work (§ 4.2.4);

- As discussed in Appendix B.1.4, we leverage **tied row attention**;
- We explore the **multi-task learning** setting (§ 4.3);
- We investigate **uncertainty quantification** to support the Bayesian optimization tasks (Appendix C.3.1).

Baselines details

Model architecture As discussed in § 4.2.5, our first baseline is the top performing model that was introduced in Hsu et al. [140], which learns a ridge regression on top of One-Hot-Encoded (OHE) representations of input protein sequences and unsupervised fitness predictions from DeepSequence [107]. We then sought to improve upon this initial baseline in two ways: 1) superior zero-shot fitness predictors 2) improved protein representations. For the former, we ran ablations in which we replaced the zero-shot fitness predictions from DeepSequence with more recent alternatives: ESM-1v [41], MSA Transformer [130], Tranception (§ 3.3.2) and TranceptEVE (§ 3.4.4. Results are provided in Table B.11. We generally see that gains in zero-shot fitness predictions translate into increased performance of the corresponding semi-supervised 'OHE - augmented' models. For the latter, we build on the extensive literature [39, 133, 142–144, 278] of semi-supervised models for protein property prediction. These works typically first extract embeddings for labelled protein sequences with a large pre-trained protein language model. Given the dimensionality of the resulting full sequence embeddings – relatively large in practice compared with the size of the training data, a dimensionality reduction technique is typically used. The most commonly used approach consists in applying a *mean pooling* [39] operator across all the tokens of the input sequences. After pooling, a non-linear transform (e.g., MLP, CNN) is used to learn complex dependencies between the corresponding protein representations and the targets of interest. Albeit simple, this approach has lead to strong performance across various benchmarks [142]. We then propose to combine these ideas with the zero-shot fitness prediction augmentations from [140] to obtain superior baselines. The

Table B.9: Embeddings Augmented - Architecture details

Hyperparameter	Value
Embedding dimension	768
CNN kernel size (post embedding)	9
Weight decay (fitness pred.)	10^{-8}
Weight decay (others)	5.10^{-2}
Dropout	0.1
Max context length	1024

corresponding baselines are referred to as 'Embeddings - Augmented'. We carry out extensive ablations to optimize this new baseline further, and present a summary of the main variants tested in the next paragraph. Our final top-performing baseline ('Embeddings - Augmented (MSA Transformer)') is based on embeddings obtained with the MSA Transformer, fed into a CNN layer with kernel size 9, followed by a ReLU non-linear activation, mean-pooled across sequence length, concatenated with zero-shot fitness predictions, and then passed to a linear projection for the final prediction. Following Hsu et al. [140], the L2-penalty on the zero-shot fitness prediction parameter is set to a much lower value (10^{-8}) compared with other parameters (5.10^{-2}). We summarize details of the final architecture in Table B.9.

As seen in Table 4.5, our improved baseline ('Embeddings - Augmented (MSA Transformer)') outperforms all OHE-based variants across all cross-validation schemes, including the ones augmented with the current best-in-class zero-shot fitness predictors. The lift is particularly important on the 'Contiguous' and 'Modulo' cross-validation schemes, for which the non-augmented OHE baseline performs just as poorly as random guessing, and the augmented equivalents perform on par with zero-shot fitness predictors.

Ablations We present in this section the main ablations conducted to give rise to our various baselines. We use the same subset of 8 DMS assays as described in Appendix B.1.4.

The first analysis (Table B.11) looks at the effect of the model used for the zero-shot fitness predictions in OHE baselines. We find that the better the underlying

Table B.10: Embeddings Augmented - Training details

Hyperparameter	Value
Training steps	10k
Learning rate warmup steps	100
Peak learning rate	3.10^{-4}
Optimizer	AdamW
Gradient clipping norm	1.0
Learning rate schedule	Cosine
Training batch	64

Table B.11: Baselines ablations - Augmented OHE Spearman's rank correlation between model predictions and experimental measurements.

CV scheme	No Aug.	DS	ESM1v	MSAT	Tranception	TranceptEVE
Random	**0.572**	0.510	0.508	0.494	0.499	0.521
Modulo	0.033	0.403	0.405	0.396	0.390	**0.423**
Contiguous	0.105	0.411	0.418	0.402	0.396	**0.434**
Average	0.237	0.442	0.444	0.431	0.428	**0.459**

zero-shot fitness predictor, the higher the performance of the corresponding OHE augmented model.

The second analysis (Table B.12) investigates various pre-trained protein language model, and concludes that representations obtained with the MSA Transformer are the most robust across the various cross-validation schemes.

The third analysis (Table B.13) compares several of the non-linear transforms that have been suggested by the prior literature to predict protein properties based on the sequence embeddings. More specifically, we investigated L2-penalized regression [142], shallow Multi-Layer Perceptron (MLP) [120], Convolutional Neural Network (CNN) [142], ConvBERT [143] and Light Attention [144]. We obtain our best results with a CNN layer with a kernel of size 9.

Model training For all baselines ('OHE' and 'Embeddings'), the loss we optimize is the MSE loss between property predictions and true labels. Similarly to what we discussed in Appendix B.1.4, we train our various baselines for 10k steps, using the AdamW optimizer with a cosine learning rate schedule. All training hyperparameters are summarized in Table B.10. At train time we find it helpful to

Table B.12: Baselines ablations - Protein sequence embeddings Spearman's rank correlation between model predictions and experimental measurements.

CV scheme	Tranception	ESM1v	MSAT
Random	0.584	0.633	**0.624**
Modulo	0.436	0.534	**0.553**
Contiguous	0.402	0.487	**0.509**
Average	0.474	0.551	**0.562**

Table B.13: Baselines ablations - Non-linear transform post embedding Spearman's rank correlation between model predictions and experimental measurements. We perform this ablation on the Random CV scheme only.

Transform	Spearman
Light attention	0.405
Linear	0.480
MLP	0.509
ConvBERT	0.617
CNN	**0.624**

pre-compute the MSA Transformer embeddings of all labelled examples to prevent GPU memory bottlenecks and speed up training. Following Rao et al. [130], the embeddings for each DMS sequence are computed by first filtering sequences in the corresponding MSA with HHFilter [361] to ensure minimum coverage of 75% and maximum sequence identity of 90%, and then performing a weighted sampling of the MSA to extract 384 natural sequences. The sampling weights are based on the procedure described in Hopf et al. [106].

B.1.5 Supervised setting – Detailed performance analysis

Extended ProteinGym benchmark

The 13 additional assays that are part of the extended ProteinGym substitution benchmark are as follows:

- Q8WTC7 9CNID, D7PM05 CLYGR, Q6WV13 9MAXI are three GFP proteins [364]
- SPG1 STRSG, human protein GB1 [365]
- CAPSD BPMS2, a capsid protein useful for possible vector delivery [366]

- PARE ECOLX, an anti-toxin [367]
- SPCAS9 [368] under positive and negative selection pressures
- BAC7, an antimicrobial synthetic sequence [369]
- A0A247D711 LISMN and A0A220GHA5 9CAUD, two artificial Cas9 sequences [370]
- HXK4 HUMAN, a human glucokinase [371]
- ANCSZ, a reconstructed ancestral Syk-family kinase [372]

Property prediction experiments for single mutants

Experiment details In this analysis we seek to assess the ability of various models to accurately predict the properties of single amino acid substitutions. We performed this analysis on all single mutants from the 100 assays from the extended ProteinGym substitution benchmark, and considered the 3 cross validation schemes described in § 4.2.5.

We compared the performance of the following models:

- **Zero-shot (MSA Transformer)**: zero-shot predictions obtained with the MSA Transformer (ensembling over 5 weighted samples from the corresponding MSA, as described in Meier et al. [41]);
- **OHE**: ridge regression on trained one-hot-encodings of the input protein sequences;
- **OHE - Augmented (DeepSequence)**: ridge regression on trained one-hot-encodings of the input protein sequences and zero-shot predictions from DeepSequence [107];
- **OHE - Augmented (MSA Transformer)**: ridge regression on trained one-hot-encodings of the input protein sequences and zero-shot predictions from the MSA Transformer [130];
- **Embeddings - Augmented (MSA Transformer)**: ridge regression on the concatenation of mean-pooled embeddings from the MSA Transformer and zero-shot predictions from the MSA Transformer (as described in Appendix B.1.4);

- **ProteinNPT**: final ProteinNPT architecture as described in Appendix B.1.4

For brevity, we use the following shorthands in the Figures and Tables throughout this subsection:

- **DS** stands for DeepSequence
- **MSAT** stands for MSA Transformer
- **PNPT** stands for ProteinNPT
- **Embed.** stands for Embeddings
- **Aug.** stands for Augmented

Detailed performance results We report the DMS-level and average performance for each cross validation scheme in Fig B.2 (Random cross validation scheme), Fig. B.3 (Modulo cross validation scheme) and Fig. B.4 (Contiguous cross validation scheme). We observe that ProteinNPT outperforms all baselines across the different cross validation schemes. The largest performance lift is observed with the Random cross validation as the model fully benefits from its column attention modules by learning from mutants that occurred at the same positions in the training set.

Property prediction experiments for multiple mutants

Experiment details In this analysis we investigate the ability of the different models listed in Appendix B.1.5 to predict the effects of multiple mutations. We carry out this analysis on the subset of 17 DMS assays with multiple mutants from the ProteinGym substitution benchmark. As discussed in § 4.2.5, since multiple mutants occur across several positions simultaneously, we only conduct this analysis with the 'Random' cross validation scheme.

Detailed performance results We report aggregated performance and corresponding standard error for all models in Table B.14. Performance is reported in terms of Spearman's rank correlation and Mean Squared Error (MSE) between model predictions and experimental measurements. In the zero-shot setting, a fitness predictor is only able to rank order mutants based on relative fitness, but is unable to predict the actual value of the experimental phenotype. Thus, we do

Figure B.2: Single mutants fitness prediction - Random cross validation scheme We report the DMS-level performance (measured by the Spearman's rank correlation ρ between model scores and experimental measurements) of the ProteinNPT and other baselines listed in Appendix B.1.5.

Figure B.3: Single mutants fitness prediction - Modulo cross validation scheme We report the DMS-level performance (measured by the Spearman's rank correlation ρ between model scores and experimental measurements) of the ProteinNPT and other baselines listed in Appendix B.1.5.

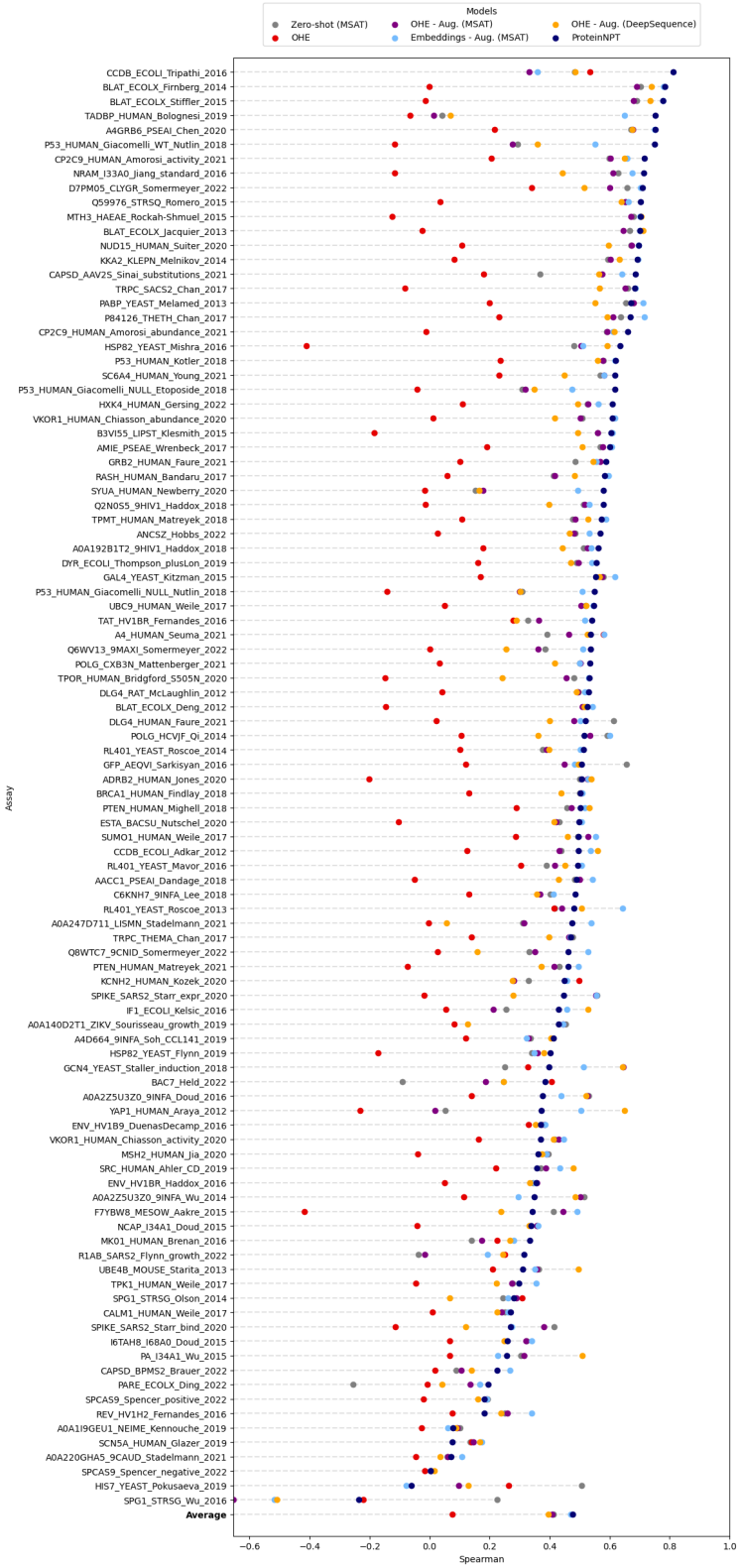

Figure B.4: Single mutants fitness prediction - Contiguous cross validation scheme We report the DMS-level performance (measured by the Spearman's rank correlation ρ between model scores and experimental measurements) of the ProteinNPT and other baselines listed in Appendix B.1.5.

Table B.14: Multiple mutants fitness prediction - Performance summary We report the aggregated performance of the ProteinNPT and other baselines listed in Appendix B.1.5, measured by the Spearman's rank correlation ρ and Mean Squared Error (MSE) between model scores and experimental measurements. The standard error of the average performance is reported in parentheses.

Model name	**Spearman ↑**	**MSE ↓**
Zero-shot (MSAT)	0.367 (0.002)	–
OHE	0.717 (0.002)	0.541 (0.008)
OHE - Augmented (DeepSequence)	0.719 (0.001)	0.658 (0.106)
OHE - Augmented (MSAT)	0.722 (0.001)	0.516 (0.008)
Embeddings - Augmented (MSAT)	0.695 (0.002)	0.479 (0.008)
ProteinNPT	**0.788 (0.002)**	**0.259 (0.001)**

not report MSE performance in the zero-shot setting and only provide Spearman performance. We also report DMS-level performance (Spearman) in Fig. 4.2 (left).

We find that the ProteinNPT markedly outperforms all other baselines both in terms of Spearman's rank correlation and Mean Squared Error. As shown in Fig. 4.2 (right), this is the case both for aggregate performance and across all mutation depths.

Property prediction experiments for multiple properties

Experiment details We seek to evaluate the ability of the models to predict several properties simultaneously for the same mutated sequences. This is often highly important in practice as the majority of real-life protein design use cases call for the joint optimization of several properties, or the optimization of one property under constraints over other properties.

To that end, we extract from the ProteinGym substitution benchmark a subset of proteins for which several phenotypes have been measured. The seven proteins with at least two distinct measurements we considered were as follows:

- RL401 YEAST
- CCDB ECOLI
- VKOR1 HUMAN
- BLAT ECOLX
- P53 HUMAN
- PTEN HUMAN
- CP2C9 HUMAN

The three proteins with at least three measurements were as follows:

- RL401 YEAST
- BLAT ECOLX
- P53 HUMAN

We focus on the 'Random' cross validation scheme, and consider all possible mutants across the different assays for the same protein, which sometimes result in certain target values being missing for a subset of mutants.

For the various baselines we compare against, a separate linear head is used to predict each target based on the concatenation of the mean-pooled embeddings and zero-shot predictions. The loss that we optimize is the sum of the MSE losses for each target. When a subset of target values are missing for a given observation, we ignore the corresponding missing values in the loss computation and only leverage the loss for non-missing target values for these training instances.

In the ProteinNPT, we simply add as input as many target columns as required, using the same masking procedure as described in Appendix B.1.4. Each target is then predicted by inputting the corresponding last-layer target embedding into a target-specific linear projection. The loss we optimize is the same composite loss which combines cross-entropy loss for the masked token prediction and the sum of MSE losses for each target prediction. When a subset of target values are missing for a given observation, we simply mask the corresponding input values. This leads to an implicit imputation scheme internally which contributes to further boosting the predictive performance.

Detailed performance results We report the performance of the ProteinNPT and other baselines for the simultaneous predictions of two protein properties in Table B.15, and three properties in Table B.16. Consistent with the results from the above analyses, the ProteinNPT outperforms all baselines across the different settings, both in terms of Spearman's rank correlation and MSE.

Table B.15: Multiple properties prediction - Performance summary We report the aggregated performance of the ProteinNPT and other baselines listed in Appendix B.1.5, measured by the Spearman's rank correlation ρ and Mean Squared Error (MSE) between model scores and experimental measurements.

Model name	Spearman ↑		MSE ↓	
	Assay 1	Assay 2	Assay 1	Assay 2
OHE	0.609	0.601	0.917	0.917
OHE - Aug. (DS)	0.541	0.553	0.678	0.702
OHE - Aug. (MSAT)	0.524	0.533	0.687	0.718
Embeddings - Aug. (MSAT)	0.648	0.668	0.490	0.516
ProteinNPT	**0.743**	**0.746**	**0.337**	**0.411**

Table B.16: Multiple properties prediction - Performance summary We report the aggregated performance of the ProteinNPT and other baselines listed in Appendix B.1.5, measured by the Spearman's rank correlation ρ and Mean Squared Error (MSE) between model scores and experimental measurements.

Model name	Spearman ↑			MSE ↓		
	Assay 1	Assay 2	Assay 3	Assay 1	Assay 2	Assay 3
OHE	0.694	0.600	0.720	2.718	2.792	2.721
OHE - Aug. (DS)	0.564	0.566	0.584	2.064	2.156	2.088
OHE - Aug. (MSAT)	0.534	0.538	0.539	2.087	2.172	2.147
Embeddings - Aug. (MSAT)	0.707	0.720	0.762	1.361	1.409	1.368
ProteinNPT	**0.786**	**0.759**	**0.821**	**0.988**	**1.150**	**0.998**

B.2 Predicting the effects of human genetic mutations

B.2.1 Obtaining clinical significance labels from ClinVar

To obtain variants with clinical labels we make use of the ClinVar public archive [34], April 2021 release, which contains reports of the relationships between human genetic variation and phenotypes, with supporting evidence. Of particular relevance for this work is the classification of single nucleotide variants (SNVs) into five categories: Benign, Likely Benign, Uncertain Significance, Likely Pathogenic and Pathogenic. In addition, the quality of evidence provided is ranked according to a four star system, which can be summarized as follows:

- No Stars – Interpretation provided but criteria not met.
- One Star – Criteria provided, single submitter.

- Two stars – Criteria provided, multiple submitters, no conflicts.
- Three Stars – Reviewed by expert panel.
- Four Stars – Practice guideline.

Of $\sim$ 78k missense variants labeled (Likely) Benign or (Likely) Pathogenic in ClinVar, $\sim$ 63k have one star or more. In most of this work, with the exception of Extended Data Fig. 6, we only consider labels with quality rating of one star or higher.

While ClinVar contains clinical labels of SNVs, our model provides evidence at the amino acid variant level. We therefore require a procedure that selects a single label whenever more than one SNV is present in ClinVar for the same amino acid substitution. For these cases we pick the label with the most stars, and if two SNVs have labels with equal star-rating, we pick the most recent. Variants which do not match the transcript used as reference in our model are dropped from the analysis.

Finally, in order to obtain a high quality set of labels for validation, we assign all no-star labels as Uncertain and collapse the remaining (Likely) Benign and (Likely) Pathogenic into just two classes, Benign and Pathogenic, respectively. Unless stated otherwise, this is the set of benign and pathogenic labels used for benchmarking throughout the text. In total, we have $\sim$ 43k such labels across 3,219 proteins. Beside all variants labeled with Uncertain Significance and with no-star rating in ClinVar, we also define as Uncertain all variants observed in gnomAD [148] and UK Biobank [145] which do not feature in ClinVar. In total we find $\sim$ 1.3M 'variants of unknown significance' across the 3,219 explored in this work.

For the purpose of comparison to predictions from high-throughput experiments (§ 4.3.3), we find the above label definitions too restrictive, with only 5 genes having both a substantial number of these high quality labels, as well as high-throughput experimental data. To expand our analysis to include more experiments, we therefore define a second more lenient label policy. We define 'Lenient' labels as the set of all labels in Clinvar – including no-star rating ones – as well as defining as Benign all variants which are more frequent in the population than the most frequent Pathogenic label in the same gene (frequencies estimated from gnomAD). This policy is similar to the one used in Refs [166, 373]. It is important to stress that we expect

these labels to be less trustworthy. Consistent with this, our model performance improves as we consider sets of labels with more stringent quality controls – our average AUC over all 3,219 proteins improves from 0.83 to 0.91 to 0.94 as we compute it against ClinVar lenient, 1-star or higher and 2-star or higher labels, respectively.

B.2.2 Performance against other models of variant effects

We compare the performance of EVE models with that of 16 of the most popular computational methods (8 supervised, 8 unsupervised) [150, 154, 155, 374–384]. Scores for these models were obtained from dbNSFP [385]; we excluded models explicitly trained on population frequency data in order to avoid circularity when validating on known labels, as frequency is currently heavily used for classification of variants.

Performance is assessed by two metrics: first, we compute AUCs and classification accuracies over high quality labels as defined above. Since we do not account for the train-validation split used by supervised models and meta-predictors at train time (as some of these methods did not make this data publicly available), the reported performance for variant effect prediction is to be interpreted as an upper bound on their true performance. Secondly, we compute the correlation with 40k experimentally measured variants across 10 proteins [156, 165–167, 312, 330, 344, 347, 349, 386] – a benchmark that we argue is less sensitive to the biases and circularity issues with ClinVar. We chose these experiments based on the *clinical relevance* of the assay. Since these experiments are independent of the ClinVar labeling process, performance metrics on this benchmark provide a less biased estimate of true generalization performance, at the cost of only being available for a limited number of genes. We note nonetheless that there may be residual training bias when assessing the performance of supervised methods on assays carried out on proteins with many labels in ClinVar (e.g., BRCA1).

Benchmarking EVE's performance against performance of high-throughput functional assays

We compare the predictions of EVE to the predictions of a number of high-throughput experiments [132, 156, 165–167, 294, 305, 340, 344, 347, 348, 386],

which were also designed with the intention of predicting the pathogenicity of variants. Not all assays are equally relevant for human disease phenotype. We compute AUCs on the intersecting set of variants common to both the assay and EVE, using multiple label definitions as described above, as well as older versions of ClinVar, to avoid comparing EVE and experiment on labels which were established by making use of data from that same experiment. On average we find EVE outperforms the experiments regardless of this choice.

In Fig. 4.8 we present results using high quality labels only – with at least 1-star quality rating – as they are sufficient for performance estimates for P53, BRCA1 PTEN, SCN5A and MSH2. We use ClinVar 2021 data only, for simplicity. This means there is some circularity potentially inflating the estimate performance for P53 and BRCA1. Reported AUCs are computed using all available labels (as opposed to intersecting set). In Fig. B.5 we use "lenient" labels as well as labels with at least 1-star quality rating (defined in Data Acquisition section).

B.2.3 Combining EVE predictions with other sources of evidence

The 2015 American College of Medical Genetics and Genomics–Association for Molecular Pathology (ACMG-AMP) guidelines [158] present steps towards a systematic approach to variant classification which can be used consistently across independent groups. They propose a classification scheme consisting of 28 criteria to classify variants into one of five categories – Benign, Likely Benign, Uncertain, Likely Pathogenic and Pathogenic. Each criteria corresponds to a different form of evidence, such as population data, or functional data, and the strength of evidence provided by a given criteria falls into one of four categories – Supporting, Moderate, Strong and Very Strong. Finally, a set of rules determines how criteria are to be combined to determine the category of a given variant. For instance, one Strong pathogenic criterion, combined with two or more Supporting criteria, would result in a variant being classified as Likely Pathogenic.

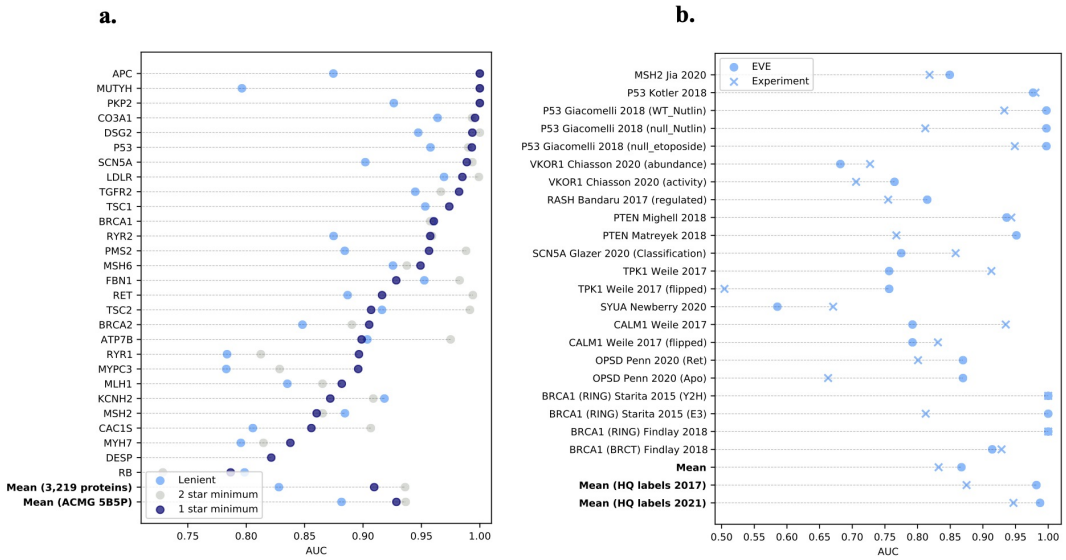

Figure B.5: Comparison of label policies, and comparison of EVE and experimental predictions of clinical labels. a. The y-axis is the subset of the ACMG actionable protein list with at least 5 benign and 5 pathogenic labels with at least a one-star review status in ClinVar, mean for the 3,219 proteins and mean for this subset. x-axis is AUCs computed using these labels (deep blue), labels with at least a two-star review status (light grey) and a more lenient labelling policy (sky blue), as defined in Supplementary Methods. **b.** AUC of EVE predictions (blue circle) and experimental predictions (blue cross) computed on ClinVar labels. Whilst most of the papers that provide these experimental results refer to the goal of predicting association to human disease, the assays vary in their relevance to disease phenotype. Results use high-quality labels whenever they are sufficient for robust validation (MSH2, P53, BRCA1) and lenient labels for all other cases, and 2017-release ClinVar data whenever experimental results were used in defining labels reported in 2021 (P53 and BRCA1). Reported averages of all displayed AUC values, and of AUCs computed exclusively on 2017 and 2021-release high-quality labels.

One of the defining characteristics of our model is the fact that it only uses one source of evidence – evolutionary data – to score variants according to their clinical significance. As such, it is straightforward to combine the EVE scores with other evidence in a manner similar to the strategy outlined by the ACMG-

AMP. We illustrate this by using EVE scores as Strong evidence, and obtain the reclassification depicted in Fig. 4.9. Further details about our approach are available in the supplementary material of our EVE paper [135].

B.2.4 Supervised models validation requirements

To estimate the number of genes on which a supervised method may be validated on a gene-by-gene basis, we repeatedly sample 10% of all ClinVar labels and count the number of genes for which there are sufficient labels for validation. Taking this to be 10 labels (of which at least one Benign and one Pathogenic), we find that the average number of genes is 52.

B.2.5 Code and data availability

All components of the end-to-end EVE approach are fully open source and available in our GitHub repository: `https://github.com/OATML-Markslab/EVE`. All training data and predictions for all genes we trained models for can be downloaded from our website: `https://evemodel.org/`.

B.3 Predicting viral immune escape

B.3.1 Training data

Multiple sequence alignments for fitness models For each viral protein, we construct multiple sequence alignments performing 5 iterations of the profile-HMM based homology search tool Jackhmmer [92] against the UniRef100 database [162]. As previously reported for EVE (Appendix A.1), we generally keep sequences that align to at least 50% of the target sequence and columns with at least 70% coverage, except in the case of SARS-CoV-2 Spike where we use lower column coverage as needed (30-70%) to maximally cover experimental positions and significant pandemic sites. For our pre-pandemic (pre-2020) alignment used as the primary model throughout § 4.4, we remove pandemic sequences using the "date of creation" variable from UniRef. We optimize search depth to maximize sequence coverage and the effective number of sequences (N_{eff}) included after re-weighting similar protein

sequences in the alignment within a Hamming distance cutoff (theta) of 0.01. To select sequence depth, we prioritized alignments with coverage >0.7L and N_{eff}/L>1, or if this was not attainable, relaxed the requirements for N_{eff}/L (Table S2).

Alignments with pandemic sequences We construct an "evolutionary alignment" with non-SARS-CoV-2 sequences as described above using Jackhmmer (with at least 50% sequence coverage, at least 30% column coverage, and theta of 0.01). We extract the full sequences pulled into the Jackhmmer alignment and re-align the sequences using super5 [90], then remove gapped positions relative to the Wuhan sequence. We also construct a "pandemic alignment" with all unique Spike sequences (with count >100) seen up until 11/27/21 (when BA.2 first appeared in GISAID). We then concatenate that "pandemic alignment" with the "evolutionary alignment" to create the final alignment.

Protein structures for accessibility calculation For each viral surface protein, we selected crystal structures representing known structural states available to B-cell and antibody interactions (extracellular conformations). All heteroatoms and protein chains not part of the multimeric viral surface protein were removed.

B.3.2 Evaluation data

Antibody footprints To identify known antibody footprints of viral surface proteins in the RCSB PDB [88], we queried the database with the protein name and the word "antibody" and required that the source organism contain both "Homo sapiens" and the given virus name. Then for each structure we identified antibody and viral protein polymer entities and computed the antibody footprint as any residue with any atom within 3.5 angstroms of the antibody. Finally, we mapped footprints to the target viral protein sequence by using SIFTS to renumber all hits according to a UniProt ID, then used a MUSCLE multiple sequence alignment of the different UniProt sequences to map those hits to the target viral protein sequence. We use this same method to identify antibody footprints for specific clinical antibodies. For experimental evidence of clinical

antibody escape susceptibility, we used the Stanford Coronavirus Antiviral and Resistance Database (CoV-RDB) susceptibility summary for monoclonal antibodies under emergency use authorization [211].

Deep mutational scans We benchmark our models on a series of viral protein deep mutational scans [175–190][190, 197–201, 387, 388]. For each viral mutational scan, we select the variable or variables of protein fitness or antibody escape treated as primary in the publications. For mutants where the result is provided as residue frequencies observed at a given site (such as results expressed as preferences and processed by dms_tools2), we normalize the data at each site by dividing by the value of the wild-type residue. For the HIV analysis, we exclude antibody VRC34.01 due to its large spread of escape mutation distal to the epitope [389]. For SARS-CoV-2 RBD, we use only antibodies/sera escape data from the Wuhan sequence for our primary results. We also utilize data provided about the antibodies tested for the SARS-CoV-2 escape DMS studies, including the class of each antibody as well as the SARS-CoV-2 neutralization potency and sarbecovirus binding breadth [177]. We use the RBD dimeric ACE2 binding and expression DMS data for analysis [190]. Additional details about how we preprocessed these different DMS assays for subsequent evaluation are provided in Thadani et al. [390].

Pandemic sequencing data We downloaded data on Spike variants and their deposit dates in the Global Initiative on Sharing All Influenza Data (GISAID) Epi-CoV project database (`www.gisaid.org`) [210] on 10/24/22. We further processed this data to get counts of combinations of mutations, the date of emergence, and PANGO lineage, as well as to get the month of emergence for each single mutation in Spike. We also downloaded consensus mutations for each PANGO lineage on 10/31/22 and mutation frequencies on 10/26/22 from Covid-19 CG [391].

B.3.3 Modeling approach

Overarching framework We express the probability of a single amino acid substitution to lead to immune escape as the product of three conditional probabilities (Fig. 4.10):

$$\begin{aligned} P(\text{Mutation escapes}) = \; & P(\text{Mutation maintains fitness}) \\ & * P(\text{Mutation accessible to Ab} \mid \text{fit}) \\ & * P(\text{Mutation disrupts Ab binding} \mid \text{fit, accessible}) \end{aligned} \tag{B.1}$$

where Ab is a shorthand for 'Antibody'.

The EVEscape index estimates the log-likelihood of escape as per the above equation. The fitness factor is obtained via a deep generative model for fitness prediction, while the accessibility and dissimilarity factors are features derived respectively from the known 3D structures for the viral protein and chemical characteristics of the amino acids involved in the mutation compared to the wild-type. Once selected, each factor is standardized and fed into a temperature scaled logistic function:

$$P(\text{Mutation escapes}) = \sum_{x \in \{f,a,d\}} \sigma\left(\frac{\text{std}(F_x)}{T_x}\right) \tag{B.2}$$

where the std(.) operator corresponds to standard scaling, $\sigma(.)$ to the logistic function, and f, a & d index fitness, accessibility & dissimilarity respectively. We then take the log-transform of the product of the 3 terms to obtain the final EVEscape scores. Factor-specific temperature scaling helps recalibrate probability estimates for each term (T_f, T_a and T_d for fitness, accessibility and dissimilarity respectively). Through hyperparameter search, we find that the fitness and accessibility components are already properly calibrated ($T_f = T_a = 1.0$), while the dissimilarity component benefits from being slightly rescaled ($T_d = 2.0$).

Fitness metric Unless specified otherwise, all experiments discussed in § 4.4 leverage the Bayesian VAE from EVE (§ 3.2.2) as the underlying fitness prediction model. In that case, the measure F_f in Eq. B.3.3 corresponds to the evolutionary index described in § 4.2.3. In § 4.4.3 and Fig. B.10, we also discuss the benefits from leveraging TranceptEVE (§ 3.4.4) as the fitness predictor, in which case the fitness term F_f is obtained via the scoring process described in § 4.2.3.

Accessibility metric Surface accessibility plays a key role in identifying where antibodies are most likely to contact a protein. While relative solvent accessibility (RSA) and weighted contact number (WCN) both reflect features of accessibility, we selected WCN as this metric also captures protrusion from the core structure that corresponds with where antibodies are known to bind proteins [202–205].

We computed weighted contact numbers [205] for each residue from structure as the sum of the square of the reciprocal distance between residue i and all other residues in the full protein (i.e., the full Spike trimer for SARS-CoV-2):

$$WCN_i = \sum_{j \neq i} \frac{1}{r_{ij}^2} \tag{B.3}$$

where r_{ij} is the distance between the geometric centers of the residue i and residue j side chains. By using squared distance, this value focuses on the degree of local interaction, and acts as a measure of exposure to the local environment that would permit antibody binding. It is both a simple and fast metric. We impute missing values in WCN due to gaps in the protein structure using the mean of WCN values of the residues preceding and following the gap. When computing antibody-binding likelihood metrics across different structural conformations (i.e., both open and closed structures for SARS-CoV-2 Spike) we used the maximum accessibility (or minimum weighted contact numbers).

Dissimilarity metric To predict the likelihood of a given mutation displacing an antibody interaction, we used a charge-hydrophobicity based measure of functional dissimilarity between the wild-type residue and the mutation residue. These are chosen as properties known to impact protein-protein interactions [206, 207]. To

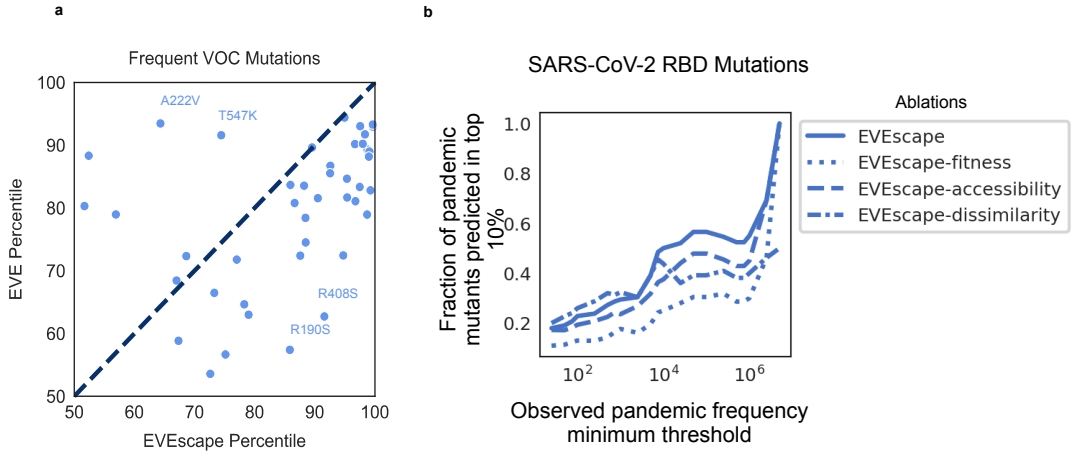

Figure B.6: The role of EVEscape components in capturing pandemic variant mutations. a. EVEscape is more predictive than EVE alone at capturing frequent VOC mutations in full Spike. VOC mutations with high EVE scores and lower EVEscape scores (i.e., A222V and T547K) are known to impact structure and to not escape sera neutralization. Mutations with the highest EVEscape but low EVE scores (i.e., R190S and R408S) are in hydrophobic pockets that may promote antibody binding [216]. **b.** EVEscape is more predictive of high-frequency pandemic mutations than ablations of any of its 3 components. Notably, the ablation of the dissimilarity term leads to similar performance at identifying low-frequency mutations, but inferior performance at identifying high-frequency mutations

compute a combined charge-hydrophobicity dissimilarity index, we standard-scaled the charge and hydrophobicity differences and then took the sum of the scaled differences. We use the Eisenberg-weiss hydrophobicity consensus scale [208] and amino acid charge (as 1/0/-1) at physiological pH.

B.3.4 Code and data availability

All components of the end-to-end EVE approach are fully open source and available in our GitHub repository: `https://github.com/OATML-Markslab/EVEscape`. Predictions for emerging variants and variants of concern are available on our website: `https://evemodel.org/`.

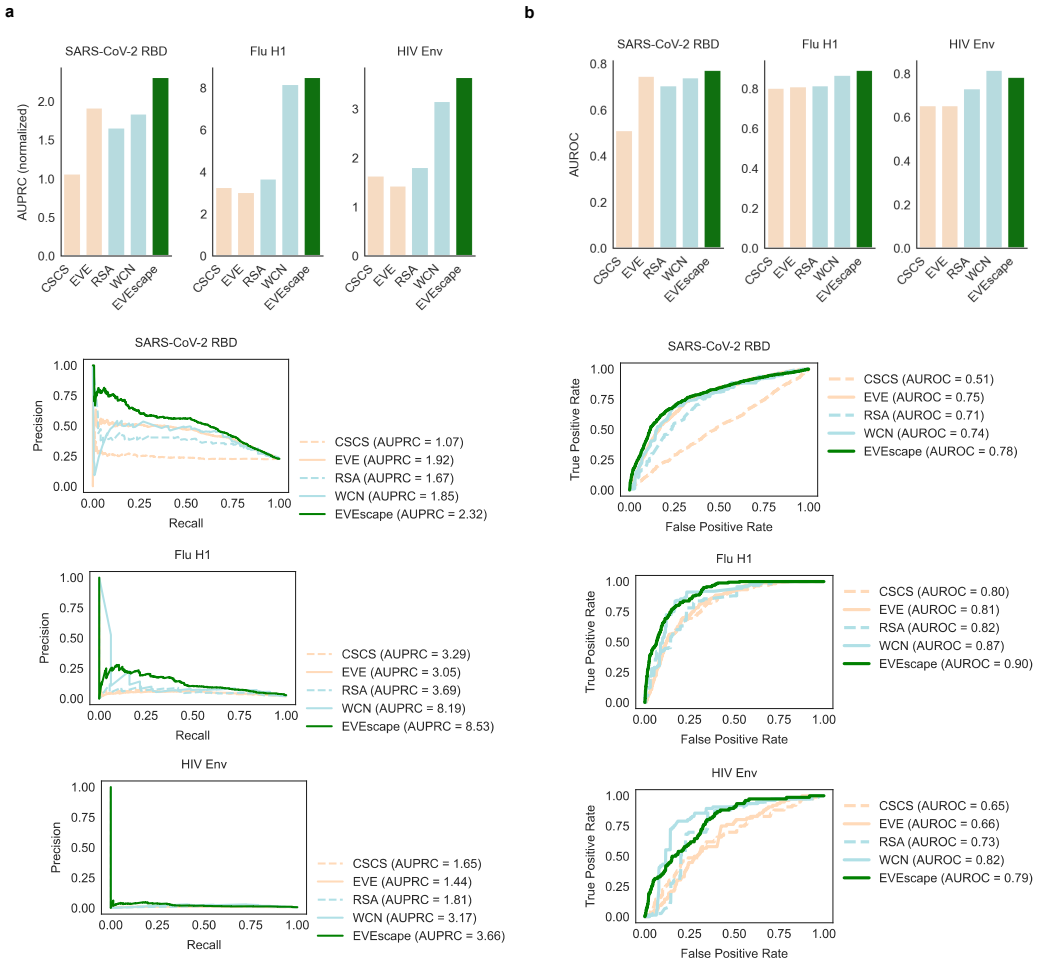

Figure B.7: EVEscape performance on escape DMS data is generalizable across viruses. Precision-Recall (with AUPRC normalized by "null" model) (**a.**) and AUROC (**b**) of predicting DMS escape mutations, for SARS-CoV-2 RBD, Flu H1, and HIV Env. The "null" model AUPRC is equivalent to the fraction of observed escapes, and therefore AUPRC values are not comparable between viral proteins with different fractions of escape mutations (i.e. RBD and HIV Env). The fraction of observed escapes in the DMS experiments are 0.19 for RBD, for 0.015 for Flu, and 0.006 for HIV – Flu and HIV data examined far fewer antibody and sera samples

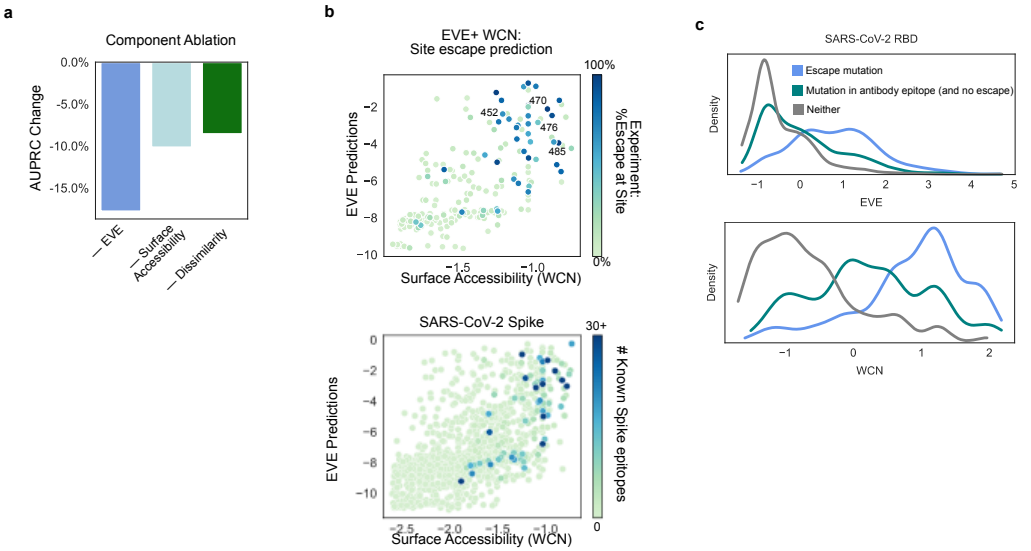

Figure B.8: Surface accessibility metrics, mutation effect models, and dissimilarity provide complementary information for predicting antibody epitopes and escape mutations a. All features of EVEscape contribute to performance in predicting RBD escape mutants. **b.** Sites with either high WCN accessibility or high EVE fitness predictions have a greater percent of escape mutants (upper). WCN and EVE predictions provide similar information about the location of Spike epitopes as represented in antibody-Spike crystal structures in RCSB PDB (lower). **c.** Density of standard-scaled EVEscape components differ for SARS-CoV-2 RBD escape (and antibody epitopes) and non-escape mutations.

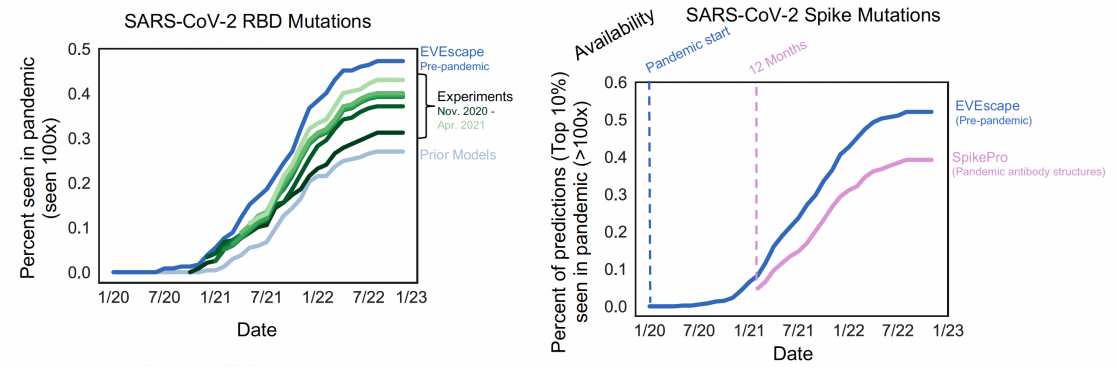

Figure B.9: EVEscape as accurate as experimental scans at anticipating pandemic variation: retrospective analysis Fraction of RBD predictions in top 15% of EVEscape, DMS experiments, and prior models [20] seen by each date over 100 times in GISAID (left). DMS experiments are separated into which studies were available by each starting date. EVEscape predictions for for full Spike and prior SpikePro model [192] (right)

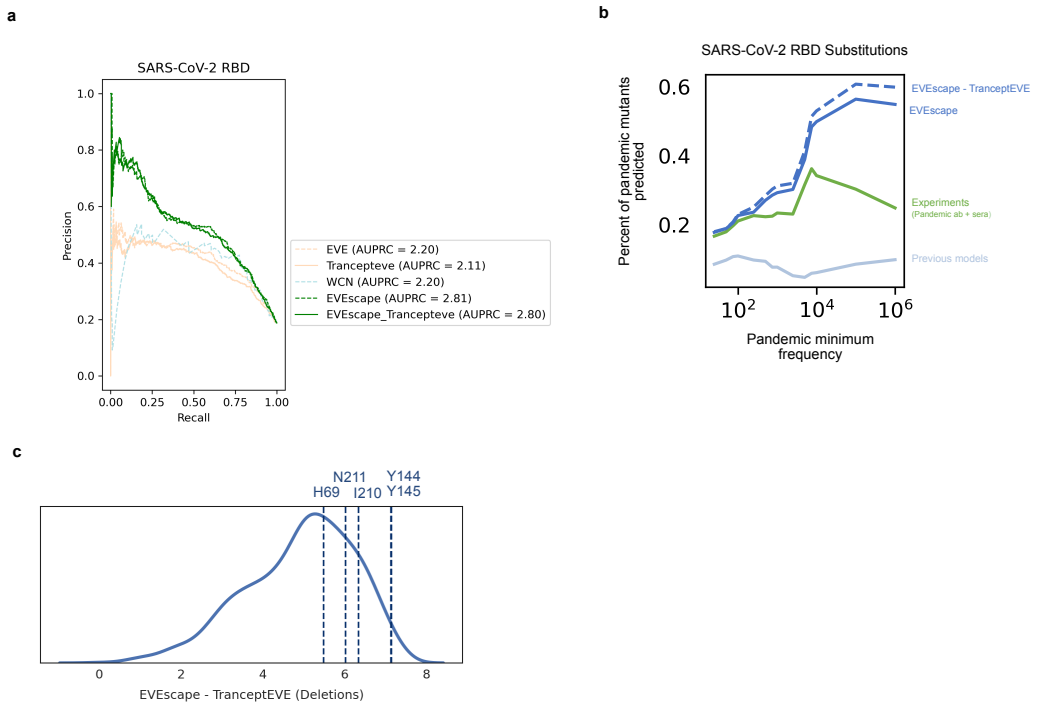

Figure B.10: Performance lift from using TranceptEVE as the fitness predictor in EVEscape. The EVEscape fitness component can be based on TranceptEVE (§ 3.4.4), which is capable of scoring substitutions as well as indels. **a.** EVEscape using TranceptEVE as the fitness model performs equivalently to EVEscape using EVE at predicting substitutions from deep mutational scans that escape antibody binding. **b.** Percent of predicted substitutions in top decile of prediction based on their observed frequency during the pandemic shows EVEscape with TranceptEVE performs slightly better than with EVE at predicting pandemic substitutions. **c.** Histogram of EVEscape scores with TranceptEVE for all single deletions to Spike. Single deletions seen in the pandemic more than 1000 times are predicted higher than most other single deletions, especially the very frequent pandemic deletion Y144- (seen more than a million times)

C

Appendix to Chapter 5 – Design

Contents

C.1 Gene target identification

C.1.1 GeneDisco benchmark details

Intervention descriptions

The treatment descriptors characterize a genetic intervention and generally should correspond to data sources that are informative about genetic functional similarity - i.e. defining which genes if acted upon, would potentially respond similarly to perturbation. Ideally, any dataset considered for use as a treatment descriptor must

be available for all potentially available interventions in the considered experimental setting. In GeneDisco, we provide three standardised gene descriptor sets for genetic interventions:

Achilles. The Achilles project generated dependency scores across cancer cell lines by assaying 808 cell lines covering a broad range of tissue types and cancer types Dempster et al. [392]. The genetic intervention effects are based on interventional CRISPR screens performed across the included cell lines. When using the Achilles treatment descriptors, each genetic intervention is summarized using a gene representation T with $q = 808$ corresponding to the dependency scores measured in each cell line. In Achilles, after processing and normalisation ([392]), the final dependency scores provided are such that the median negative control (non-essential) gene effect for each cell line is 0, and the median positive control (essential) gene effect for each cell line is -1. The rationale for using treatment descriptors based on the Achilles dataset is that genetic effects measured across the various tissues and cancer types in the 808 cell line assays included in Dempster et al. [392] could serve as a similarity vector in functional space that may extrapolate to other biological contexts due to its breadth.

Search Tool for the Retrieval of Interacting Genes/Proteins (STRING) Network Embeddings. The STRING [393] database collates known and predicted protein-protein interactions (PPIs) for both physical as well as for functional associations. In order to derive a vector representation suitable to serve as a genetic intervention descriptor, we utilised the network embeddings of the PPIs contained in STRING as provided by Cho et al. [394, 395] with dimensionality $q = 799$. PPI network embeddings could be an informative descriptor of functional gene similarity since proteins that functionally interact with the same network partners may serve similar biological functions [396, 397].

Cancer Cell Line Encyclopedia (CCLE). The CCLE [398] project collected quantitative proteomics data from thousands of proteins by mass spectrometry across 375 diverse cancer cell lines. The generated protein quantification profiles with dimensionality $q = 420$ could indicate similarity of genetic interventions

since similar expression profiles across a broad range of biological contexts may indicate functional similarity.

Assays

As ground-truth interventional outcome datasets, we leverage various genome-wide CRISPR screens, primarily from the domain of immunology, that evaluated the causal effect of intervening on a large number of genes in cellular model systems in order to identify the genetic perturbations that induce a desired phenotype of interest.

Interleukin-2 production in primary human T cells This dataset is based on a genome-wide CRISPR interference (CRISPRi) screen in primary human T cells to uncover the genes regulating the production of Interleukin-2 (IL-2). CRISPRi screens test for loss-of-function genes by reducing their expression levels. IL-2 is a cytokine produced by $CD4^+$ T cells and is a major driver of T cell expansion during adaptive immune responses. Assays were performed on primary T cells from 2 different donors. The detailed experimental protocol is described in Schmidt et al. [241]. The target outcome measurement is the log fold change (high/low sorting bins) in IL-2 normalized read counts (averaged across two biological replicates for robustness). Sorting was done via flow cytometry after intracellular cytokine staining. IL-2 is central to several immunotherapies against cancer and autoimmunity.

Interferon-γ production in primary human T cells This dataset is also based on Schmidt et al. [241], except that this experiment was performed to understand genes driving production of Interferon-γ (IFN-γ). IFN-γ is a cytokine produced by $CD4^+$ and $CD8^+$ T cells that induces additional T cells. The target outcome measurement is the log fold change (high/low sorting bins) in IFN-γ normalized read counts (averaged across two biological replicates for robustness). IFN-γ is critical to cancerous tumor killing and resistance to IFN-γ is one escape mechanism for malignant cells.

Vulnerability of Leukemia cells to NK cells This genome-wide CRISPR screen was performed in the K562 cell line to identify genes regulating the sensitivity of leukemia cells to cytotoxic activity of primary human NK cells. Detailed protocol is described in Zhuang et al. [242]. The measured outcome is the log fold counts of gRNAs in surviving K562 cells (after exposition to NK cells) compared to control (no exposition to NK cells). Gene scores are normalized fold changes for all gRNAs targeting this gene. Better understanding and control over the genes that drive the vulnerability of leukemia cells to NK cells will help improve anti-cancer treatment efficacy for leukemia patients, for example by preventing relapse during hematopoeitic stem cell transplantation.

Modulation of Tau proteins in neurons This assay is based on Sanchez et al. [399] in which authors have conducted genome-wide CRISPR screens in two neuroblastoma cell lines to identify genes that, when knocked out, either increased or decreased expression of endogenous tau proteins. After editing, cells are FACS sorted based on low tau expression (bottom quartile) and high tau expression (top quartile). Statistical significance of gRNA enrichment is determined via a redundant siRNA activity (RSA) analysis. RSA up scores were used to find genes enriched in the top quartile of cells with highest tau protein, while RSA down scores were used to find genes enriched in the bottom quartile of cells with the lowest tau protein. While the exact mechanism leading to the buildup of Tau proteins is unknown, their accumulation is correlated with several neurodegenerative pathologies (e.g., Alzheimer's disease, progressive supranuclear palsy, and frontotemporal dementia). The genes identified in this screen may therefore help gaining a better understanding of the underlying disease pathways as well as leading to potential treatments.

Regulation of endosomal entry in cells for SARS-CoV-2 This dataset is based on the genome-wide CRISPR screen described in Zhu et al. [240]. The assay was designed to identify endosomal entry-specific regulators of SARS-CoV-2 virions in A549-ACE2 cells. The top candidates from the CRISPR screen were determined according to their MAGeCK score (- log10). The endosomal pathway is one of the

two pathways (along with fusion at the plasma membrane) used by SARS-CoV-2 to infect cells. A better understanding of mechanisms underpinning this pathway may help identify targets for the development of new SARS-CoV-2 antiviral therapeutics.

C.1.2 DiscoBAX approach details

Sub-modularity of S

Observation 2. *The score function* $S : \mathcal{P}(\mathcal{G}) \to \mathbb{R}$ *defined by*

$$S(G) = \mathbb{E}_{f_{out}}[\max_{g \in G} \max(0, f_{out}(g))] \tag{C.1}$$

is submodular.

Proof.

$$\begin{aligned}
S(G \cup \{g\}) &= \mathbb{E}_{f_{\text{out}}}[\max_{g' \in G \cup \{g\}} \max(0, f_{\text{out}}(g'))] \\
&= \mathbb{E}_{\eta}[\max_{g' \in G \cup \{g\}} \max(0, f_{\text{out}}(g', \eta))] \\
&\leq \mathbb{E}_{\eta} \max_{G}[\max(0, f_{\text{out}}(g', \eta)) + \max(0, f_{\text{out}}(g, \eta))] \\
&= \mathbb{E}_{\eta} \max_{G}[\max(0, f_{\text{out}}(g', \eta))] + \mathbb{E}_{\eta}[\max(0, f_{\text{out}}(g, \eta))] \\
&= S(G) + S(\{g\})
\end{aligned}$$

□

Corollary 1. *The greedy algorithm which iteratively selects points maximizing* $S(G)$ *is a* $1 - 1/e$ *approximation of the optimal.*

C.1.3 DiscoBAX detailed experimental results

Clustering of optimal interventions

In the GeneDisco experiments, we define a diversity metric based on the recall of Top-K clusters. These clusters are obtained for each assay as follows. All experiments we carried out leverage the Achilles dataset [392] to represent the different interventions. We first select the optimal interventions as the ones in the top percentile of disease phenotype for a given assay. We then project the Achilles

representations of each intervention into a lower-dimensional subspace of dimension 20 with PCA. We then fit a Gaussian Mixture Model (GMM) with 20 mixtures to obtain the different clusters, selecting the best result out of 20 random initializations.

Detailed performance analysis

We provide below detailed results across the five CRISPR assays from the GeneDisco benchmark. For the active learning baselines already present in GeneDisco we used the same hyperparameters as in [237]. For the additional baselines discussed in § 5.2.4, we use standard/default hyperparameters everywhere except as specified in Appendix C.1.3. To prevent model overfitting during the various active learning cycles, we closely followed experimental protocol in Mehrjou et al. [237] and selected similar model architectures and hyperparameters.

We observe in Table C.1 that DiscoBAX outperforms all other 13 baselines we compare against on 3 out of the 5 datasets included in the GeneDisco benchmark, performs on par (significant overlap of confidence intervals) with the best methods on the 4^{th} one (Tau protein) and is only outperformed by "random selection" on the last one (SARS-coV2). As discussed in § 5.2.4, the fact that random outperforms all other 13 methods on that dataset seems to indicate an issue with the data (eg., the feature space does not correlate with the disease phenotype, high label noise) rather than an algorithmic issue. Critically, none of the other baselines performs consistently high on all 5 assays: for instance, 'random' performs relatively poorly on all other 4 assays and the other methods that are on par with DiscoBAX on the Tau protein assay (e.g., BADGE, Coreset) have inconsistent performance on other assays. Aggregated performance across assays is reported in Table 5.1 and demonstrates the overall higher performance of DiscoBAX over other baselines. The superior sample efficiency of the scheme is also apparent Fig. C.1 as DiscoBAX exhibits superior recall and diversity score throughout the different learning cycles.

GeneDisco experiments - hyperparameter selection

For the three BAX alogirthms (Top-K BAX, Levelset BAX and DiscoBAX), we optimize the main hyperparameters of each method (i.e., respectively the K

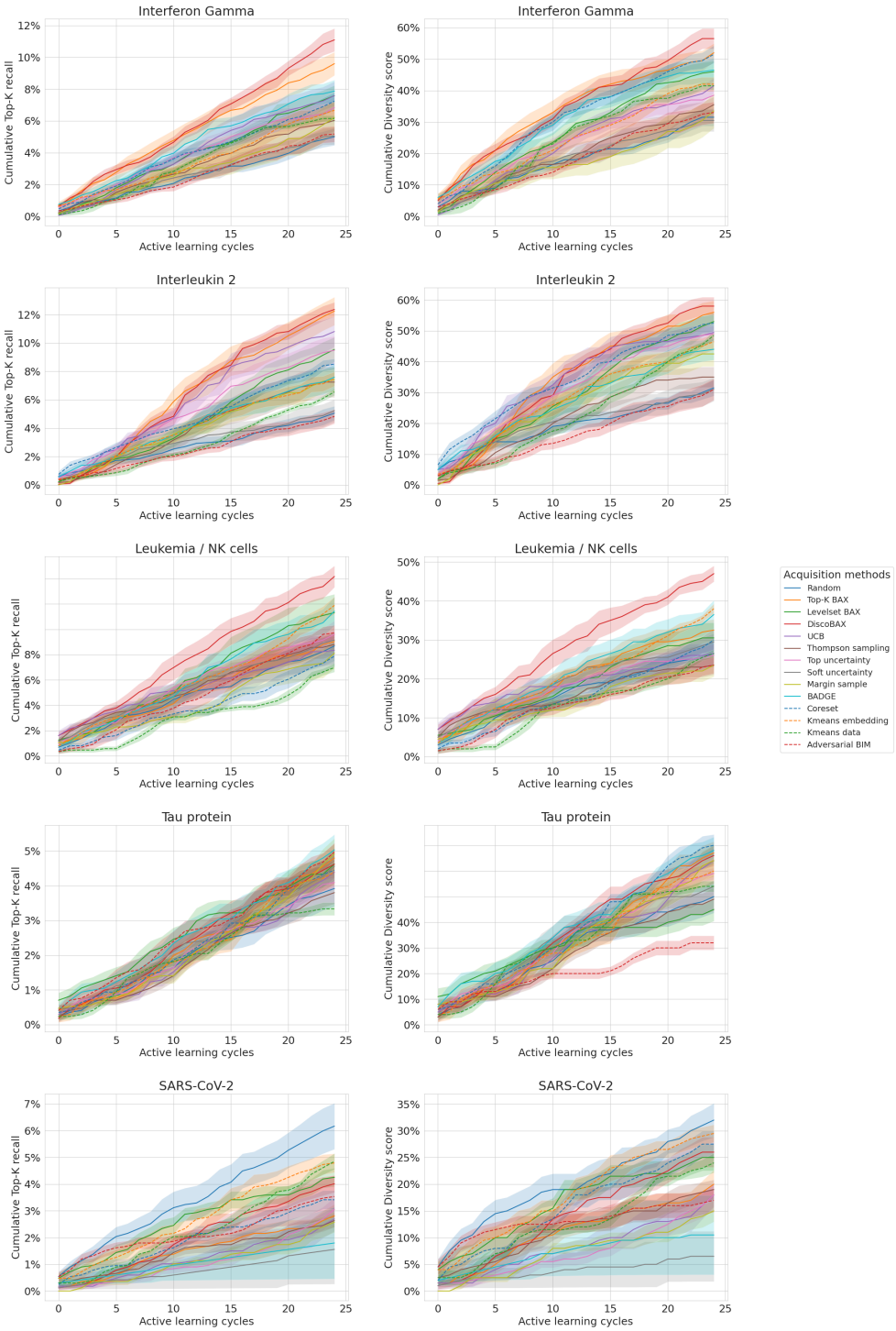

Figure C.1: Top-K recall and Diversity score Vs acquisition cycles for all GeneDisco CRISPR assays

parameter the level threshold and the number of Sets in SetSelect). To mitigate the risk of overfitting, we select our hyperparameters based on a single assay (the 'Tau protein' assay), and use the obtained optimal values in experiments for all assays. We perform a grid search for each hyperparameter, repeating each experiment over 5 seeds. We find that on that dataset, optimal values for the hyperparameters are respectively k=5, Levelset=1.0 and S=10.

C.2 De novo molecular design

C.2.1 Analysis of variance of uncertainty estimators

We demonstrate the lower variance of the Importance sampling-based estimator compared to the naive Monte Carlo estimator, focusing on the Character VAE for molecular generation setting described in § 5.3.3.

Setup. We sample 1,000 points at random in latent space from an isotropic Gaussian with fixed standard deviation σ (we repeat the experiment for different values of the standard deviation). We assess at these points the mutual information between outputs (generated molecule SMILES) and decoder parameters with the Importance sampling-based estimator (IS-MI) described in § 5.3.2, and the naive Monte Carlo (MC-MI) equivalent. The MC-MI estimator is obtained directly from equation 5.5 by sampling output sequences y_s uniformly at random from $\mathcal{S}$, the space of all possible SMILES strings. As per the setting described in Appendix C.2.2, we limit the length of molecular sequences to 120 characters from a vocabulary comprised of 34 elements (plus the padding character). Consequently, $\mathcal{S}$ is finite and we can uniformly sample from it by independently sampling characters at each position uniformly at random from the vocabulary. Both estimators are computed by sampling a fixed number of decoder parameters using Monte Carlo dropout [239] (we used 100 model samples in all experiments).

Results. We analyze the impact of the number of y_s samples for each estimator on the variance of the corresponding estimators, measured over 10 independent runs. More specifically, we compute the standard deviation over the 10 runs, normalized by the estimator mean across runs. We observe that the IS-MI estimator has a

normalized standard deviation 2-10x smaller than the MC-MI estimator across the different experiments (Fig. C.2).

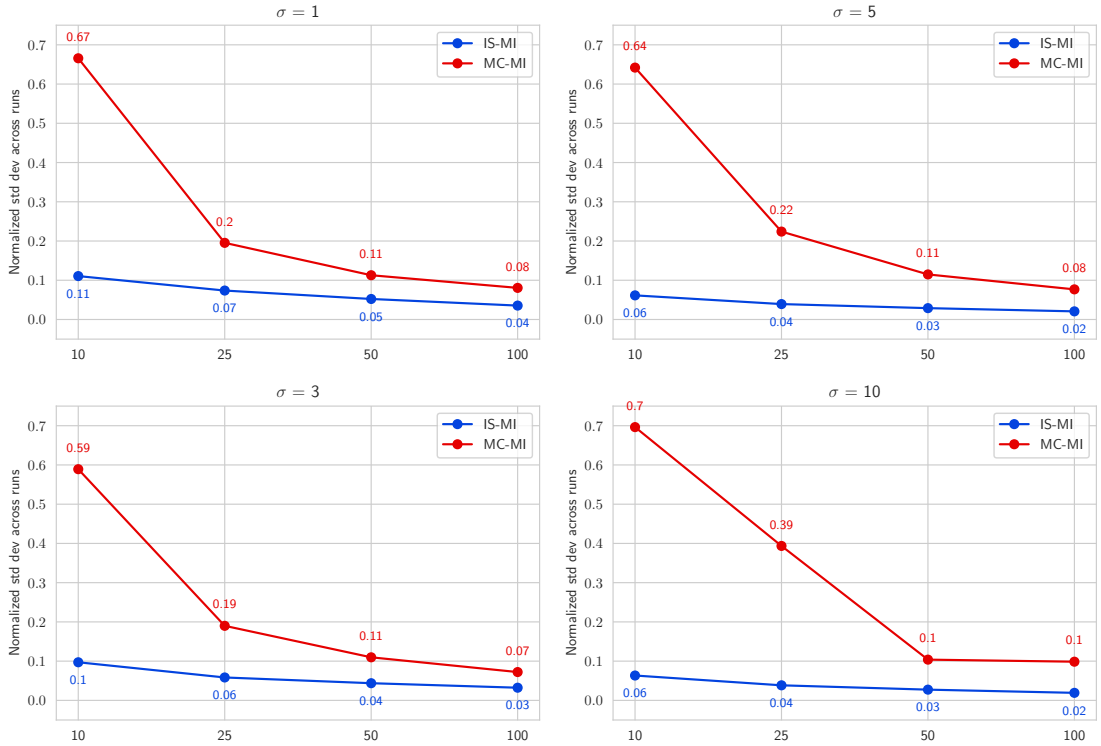

Figure C.2: Variance analysis for uncertainty estimators Comparison of the normalized standard deviations based on the number of sampled output sequences for the IS-MI Vs MC-MI estimators. We vary the standard deviation of the isotropic Gaussian used to sample points in latent space, from 1 (top left) to 10 (bottom right).

C.2.2 Molecule generation experiments with CVAE

Data

Data source. We use a dataset of 250k drug-like molecules from the ZINC database [255]. Each molecule is represented via its SMILES representation [81], ie. as a sequence of characters (from a vocabulary of 34 elements, plus the padding character). Following [17], molecule length is capped at 120, and shorter strings are space-padded to this length.

Target. The black-box objective in this set of experiments is the 'penalized logP', defined as the octanol-water partition coefficient penalized by the synthetic accessibility score and the number of long cycles. We follow prior work [42, 243,

247, 250] and compute this metric as follows:

$$\text{Penalized log P}(x) = \widehat{\log P(x)} - \widehat{SAS(x)} - \widehat{cycle(x)} \tag{C.2}$$

where $\log P(x)$ is the octanol-water partition coefficient, $SAS(x)$ is the synthetic accessibility score, $cycle(x)$ counts the number of rings that have more than six atoms, and the $\widehat{\ }$ operator represents the standard normalization based on the raw training subset from ZINC (ie. subtracting the mean of the training set, and dividing by the standard deviation).

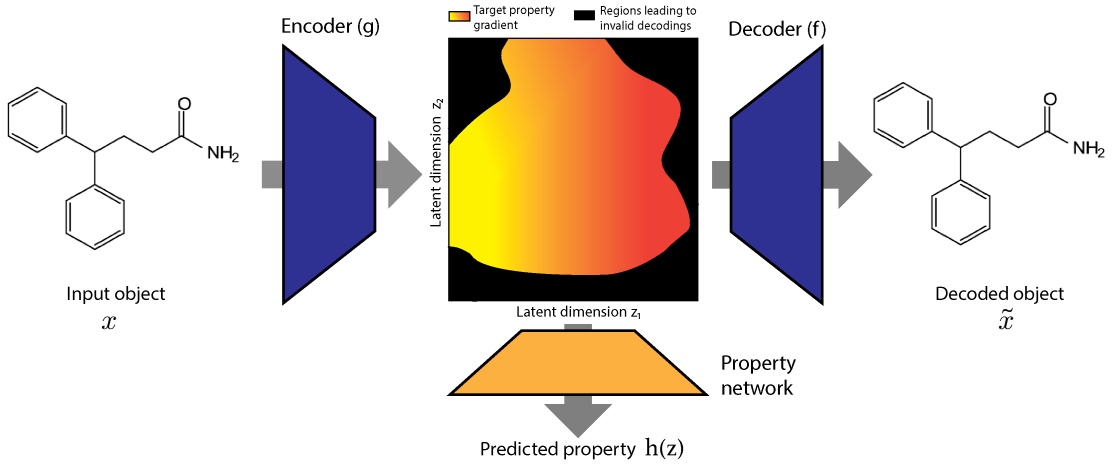

Figure C.3: Joint training architecture A Variational autoencoder (VAE) is jointly trained with an auxiliary network predicting the value of the black-box objective from the latent space encoding. Optimization is then carried out in latent space via gradient ascent or Bayesian optimization

Model details

Architecture. We adopt a model architecture similar to Gómez-Bombarelli et al. [17]: the encoder is comprised of 3 convolutional layers, the decoder is composed of a stack of 3 GRU layers [400] and the property network has a simple feed forward architecture with 3 hidden layers. A detailed description is provided in Table C.3.

Training. The total loss we minimize is the sum of the VAE ELBO and the MSE loss on the black-box property prediction task. We train the network with the Adam algorithm [401] with a learning rate of 5.10^{-4} (reduced by a factor 2 with a patience of 10 epochs) for 150 epochs total. We anneal the KL divergence with a

sigmoid schedule for the first 30 epochs to avoid potential posterior collapse. We also use teacher forcing on the character sampled at each time step in the decoder during training, and gradient clipping (upper bound set to 10) to avoid exploding gradients.

Uncertainty estimators

We evaluate the IS-MI estimator values in different regions of latent space. We consider 4 distinct sets of points:

- **'Train':** Embeddings into latent space of 1k molecules sampled randomly from the train set;
- **'Test':** Embeddings into latent space of 1k molecules sampled randomly from the test set;
- **'Prior':** 1k random samples from the VAE prior;
- **'Far from prior':** 1k random samples from an isotropic Gaussian with standard deviation equal to 20.

We compute the IS-MI estimator based on Algorithm 5.3.2, use Monte Carlo dropout sampling to obtain 100 samples from decoder parameters, and sample 100 molecule SMILES from the importance distribution. In this setting, the TI-MI estimator provides very poor uncertainty estimates as it assigns very low uncertainty values for a majority of 'out-of-distribution' points that were sampled far from the prior (Fig.C.4). This is consistent with what we see in Fig.5.5b. In this experiment, we analyze the proportion of valid decodings when keeping the x% points we are most certain about, based on the various estimators considered (ie. IS-MI, NLLP and TI-MI). If low uncertainty for a given estimator corresponds to high validity, then the % of valid decodings should increase as we keep a narrower set of most confident points. This is what we observe for the IS-MI estimator, unlike the TI-MI estimator which incorrectly selects points leading to invalid decodings as the lowest uncertainty points.

Detailed optimization results

Gradient ascent. We start from 200 point sampled at random from the training set and embedded in latent space. We perform 10 gradient update steps with

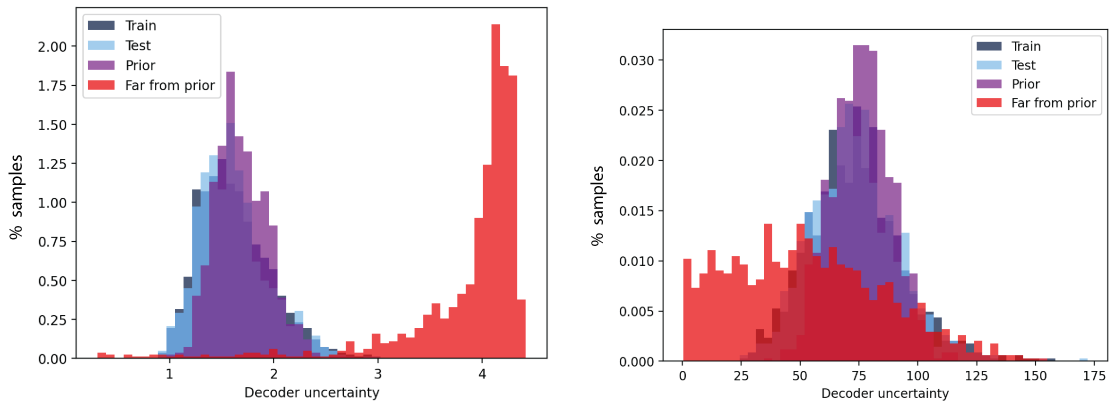

Figure C.4: Molecular generation with CVAE - Decoder uncertainty distribution. Analysis for the IS-MI estimator (left) and the TI-MI estimator (right)

a value of alpha equal to 20. In the experiments where we impose a maximum threshold on the decoder uncertainty, we set that threshold as the 99^{th} percentile of corresponding estimator values observed on the training data. Leveraging the IS-MI estimator during optimization helps reaching higher values of the black-box objective (Table C.4). While constraints imposed with the NLLP baselines do help maintaining a higher % of valid decodings, using this baseline is detrimental to optimization performance.

Bayesian Optimization. We train a single task Gaussian Process (GP) on 500 points sampled at random from the training set and embedded in latent. We use the Expected Improvement as our acquisition function, sequentially generate 100 new molecules (and re-train the GP after each acquisition). Similar to our gradient ascent experiments, we set the uncertainty threshold to the 99^{th} percentile of corresponding estimator values observed on the training data. We observe that when using the IS-MI estimator, we not only increase the % of valid decodings by 1.5-10x compared to baselines, but we also reach higher values of the 'penalized logP' objective across all settings (Table. C.5). These results are robust to the choice of distribution decile used to define the uncertainty threshold, as shown in Table.C.6.

C.2.3 Molecule generation experiments with JTVAE

Data

We refer the reader to Appendix C.2.2 as we used the same experimental setting as for the CVAE experiments, ie. we train our model on a subset of 250k drug-like molecules from the ZINC database, and seek to optimize the 'penalized logP' metric.

Model architecture

Architecture. We jointly train a Junction Tree VAE model (JT-VAE) with a property network predicting the 'penalized logP' property based on latent representation, leveraging the same architecture design as in Jin et al. [250] for the constrained optimization task. The only difference we introduce is the incorporation of dropout layers in the junction tree decoder and graph decoder to allow sampling from the decoder parameters via Monte Carlo dropout (see Table C.7).

Training. In line with the experiments discussed above, and following the same training procedure as per Jin et al. [250], we minimize the sum of the VAE loss and the MSE on the black-box prediction task.

Uncertainty estimators

We compute the IS-MI estimator based on Algorithm 5.3.2. Following notations from Algorithm 5.3.2, we sample a molecule $\tilde{y}_s$ by successively decoding from the junction tree decoder and then the graph decoder. We then decompose $\log p_{s,m}$ as the sum of the log probabilities corresponding to each prediction made by the junction tree decoder and graph decoder (for the graph outcome generated in the previous step), namely the topology and node predictions in the junction tree decoder, and the subgraph prediction in the graph decoder. Similar to the setting discussed in Appendix C.2.3, we inspect the distribution of the IS-MI values obtained on the same 4 datasets (using 1k samples for each dataset), and estimate the mutual information with 100 samples from decoder parameters and 100 molecules sampled from the importance sampling distribution. We observe similar results as before: overlap between IS-MI distributions on the 'Train', 'Test' and 'Prior' sets. This

overlap is stronger between the first two sets as the embedding of the training data in latent does not necessarily follow a standard normal distribution after model training. The distribution of IS-MI values on the 'Far from prior' set is disjoint from the first 3 sets, with the highest values obtained on this set.

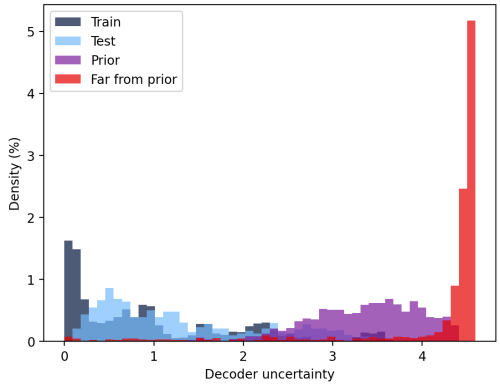

Figure C.5: Molecular generation with JT-VAE - Decoder uncertainty distribution.

Detailed optimization results

All molecules generated by the JT-VAE are valid by design. However, not all generated molecules will be of high *quality*, as measured for example by the quality filters from Brown et al. [256] discussed in § 5.3.3. Since in this molecular generation setting, our objective is to generate new molecules with high penalized logP values that could be used as potential drugs, we want to ensure that the candidate drugs we shortlist for further investigation also pass these quality filters. In each optimization experiment described below, we prioritize a small number of candidate molecules (eg., top 10 molecules with highest logP values) and use the quality filters from Brown et al. [256] as a proxy for the subsequent (costly) verification of these candidates by medicinal chemists. If a generated molecule does not pass these quality filters, its penalized logP value is assigned to a default value (eg., average penalized logP value on the training set). In all optimization experiments, we estimate mutual information with 100 samples from decoder parameters, and a single molecule sampled from the importance distribution.

Gradient ascent. We start from 100 molecules sampled at random from the training set, that we then embed in latent space. We perform 100 gradient update steps with a large value of α (as per notations in § 5.3.2), eg., 100 or 200. This leads state-of-the-art performance in terms of penalized logP values (see Table C.8 and Fig. C.6). However, as we move 'further away' in latent space, the quality of generated molecules tends to degrade. By setting an upper bound on the uncertainty of the decoder (eg., 95^{th} percentile of IS-MI values observed on the training data) during optimization, we are able to generate molecules with both high penalized logP values and high quality (see Table C.9). Selecting different uncertainty threshold values enable to reach different trade-offs between quality and black-box objective. A similar approach with upper bounds in terms of NLLP values (eg., threshold defined as 95^{th} or 99^{th} percentiles of NLLP values on the training data) does help promoting high quality molecules but leads to much lower penalized logP values.

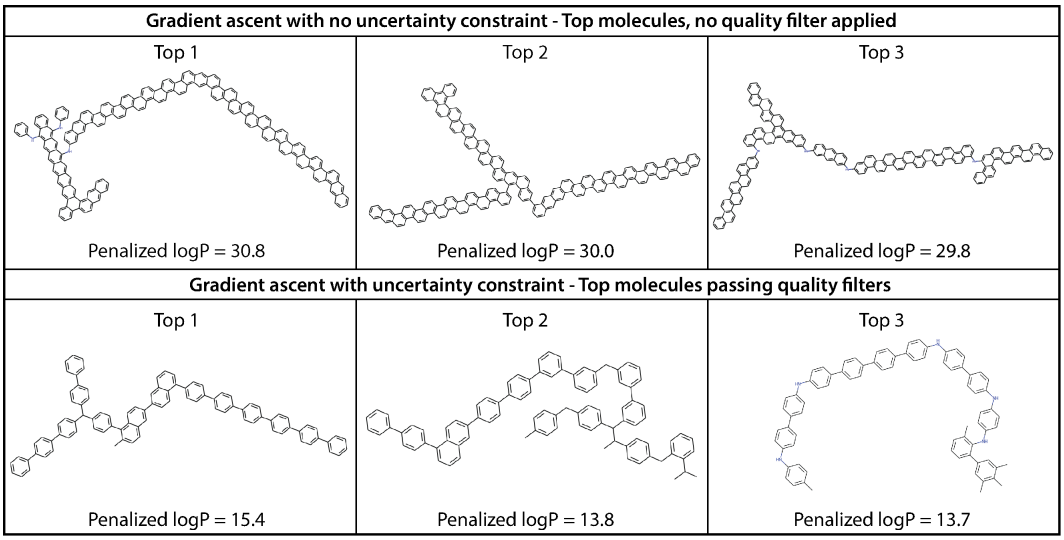

Figure C.6: Top molecules generated via gradient ascent with a JT-VAE ($\alpha = 200$).

Bayesian Optimization. We train a single task Gaussian Process (GP) on 500 points sampled at random from the training set and embedded in latent. We use the Expected Improvement as our acquisition function, sequentially generate 500 new molecules (and re-train the GP after each acquisition). In experiments in which we impose an upper bound on decoder uncertainty, we set that bound

as the 99^{th} percentile of decoder uncertainty values observed on the training data. Similar to what we observe in the gradient ascent experiments, leveraging the IS-MI estimator helps generating candidate molecules with both high 'penalized logP' values and high quality (Table C.10). The NLLP of proposal points at each step of the batch Bayesian Optimization tend to always be above the NLLP threshold (99^{th} percentile of values on the training data). In these situations, we select the point with lowest NLLP value from the proposal batch as described in § 5.3.3. This explains why the results obtained with NLLP are closer to what we obtain without any constraint during optimization, and why we lose the ability to increase the quality of generated molecules in this setting.

C.3 Protein Design

C.3.1 Uncertainty calibration

We quantify the uncertainty in the ProteinNPT and baselines models as the standard deviation of predicted target values across several forward passes via Monte Carlo dropout [239]. All experiments are conducted with a dropout rate equal to 0.1 and averaging across 10 forward passes.

We plot the test-set calibration curves across the three cross-validation schemes and aggregated over DMS assays (Fig. C.7). As we progressively remove points with highest uncertainty, we observe that the average Mean Squared Error (MSE) monotonically decreases, thereby confirming the proper calibration of our uncertainty estimates.

C.3.2 Iterative protein redesign - Detailed results

We follow the experimental protocol described in § 5.4.2 for iterative redesign experiments with the ProteinNPT and other baselines. The 24 representative assays selected for this analysis are as follows:

- BLAT ECOLX
- GRB2 HUMAN
- Q6WV13 9MAXI
- PABP YEAST

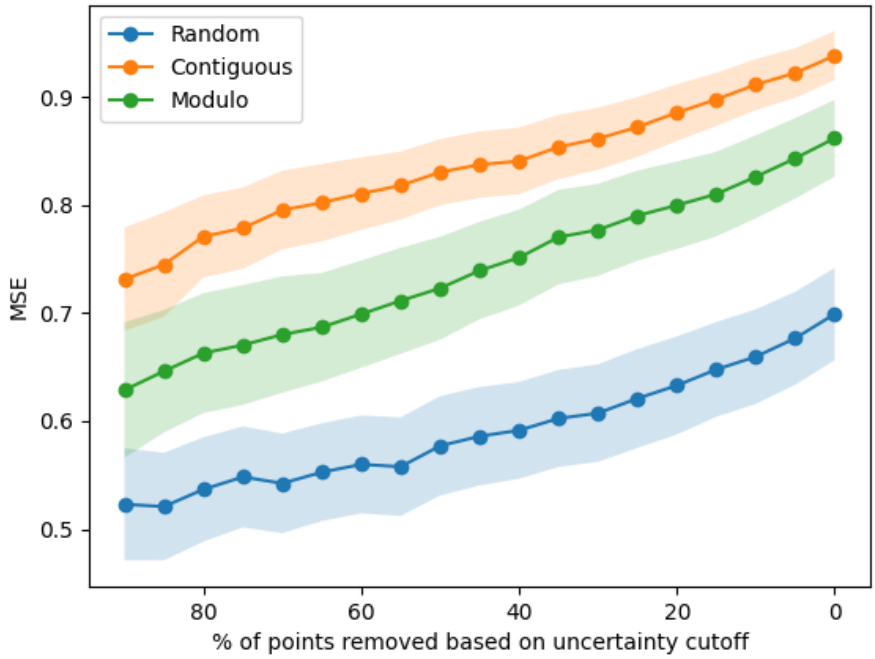

Figure C.7: ProteinNPT - Aggregated test-set calibration curves. For each cross validation scheme, the Mean Squared Error monotonically decreases as we progressively remove the test points with highest uncertainty.

- RL401 YEAST
- GAL4 YEAST
- Q8WTC7 9CNID
- BLAT ECOLX
- DLG4 HUMAN
- TAT HV1BR
- CALM1 HUMAN
- REV HV1H2
- ESTA BACSU
- DYR ECOLI
- CAPSD BPMS2
- SYUA HUMAN
- UBC9 HUMAN
- VKOR1 HUMAN
- NUD15 HUMAN
- RASH HUMAN
- ANCSZ
- P53 HUMAN
- HXK4 HUMAN
- A4D664 9INFA
- MSH2 HUMAN

We run each iterative design experiments three times, with a different random seed to select the initial labelled pool $\mathcal{D}_L$. We set the λ hyperparameter in the UCB acquisition function to 0.1 for all experiments, for all baselines, based on an

initial hyperparameter search on a subset of 3 DMS assays. Performance plots across acquisition cycle for each DMS assay are provided in Figure C.8.

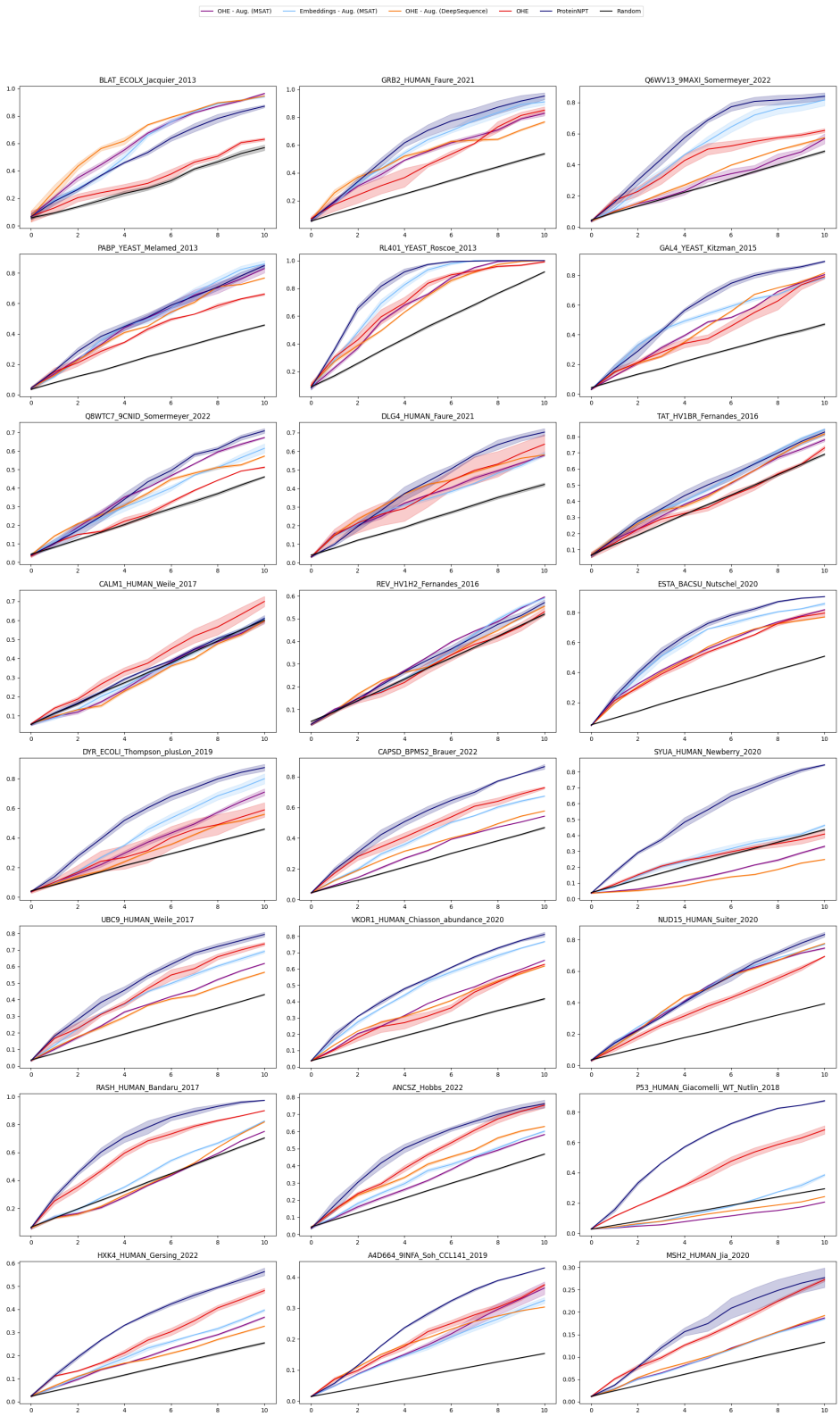

Figure C.8: In silico iterative protein redesign experiments for each DMS assay. We compute the proportion of acquired points at each cycle for which the fitness is in the top 3 deciles for the DMS assay. Results are provided for each of the 24 assays selected. The shaded areas represent standard errors across the 3 independent runs.

Table C.1: Performance comparison on GeneDisco CRISPR assays We report the performance of DiscoBAX and all baselines methods on all GeneDisco assays.

Dataset	Method	Top-K recall	Diversity score	Overall score
Interferon γ	Adversarial BIM	33% (3.6%)	5.2% (0.5%)	13.1% (1.4%)
	BADGE	46.5% (3.9%)	7.9% (0.7%)	19.1% (1.7%)
	Coreset	51.5% (3%)	7.2% (0.6%)	19.3% (1.3%)
	DiscoBAX (ours)	**56.5% (3.4%)**	**11.1%** (0.8%)	**25% (1.6%)**
	Kmeans Data	41.5% (1.3%)	6.1% (0.2%)	16% (0.5%)
	Kmeans Embedding	42% (2.4%)	6.7% (0.6%)	16.7% (1.2%)
	Levelset BAX	46% (3.6%)	7.6% (0.7%)	18.7% (1.5%)
	Margin sample	33.5% (5.8%)	6.2% (1.1%)	14.4% (2.6%)
	qEI	45% (3.6%)	7.9% (0.4%)	18.9% (1.1%)
	qPOI	44% (3.1%)	8.1% (0.4%)	18.9% (1.1%)
	qUCB	43.7% (3.7%)	7.8% (0.4%)	18.5% (1.2%)
	Random	31.5% (2.9%)	5% (0.6%)	12.5% (1.3%)
	Soft Uncertainty	30.5% (3.7%)	5.1% (0.6%)	12.4% (1.5%)
	Thompson Sampling	35.5% (2.6%)	6% (0.7%)	14.6% (1.4%)
	Top-K BAX	52% (3.1%)	9.6% (0.8%)	22.3% (1.5%)
	Top Uncertainty	38.5% (3%)	6.8% (0.7%)	16.2% (1.4%)
	UCB	41.5% (2.6%)	7.6% (0.9%)	17.7% (1.6%)
Interleukin 2	Adversarial BIM	31% (3.6%)	4.8% (0.5%)	12.2% (1.4%)
	BADGE	44% (3.6%)	7.6% (1%)	18.3% (1.9%)
	Coreset	52.5% (2.9%)	8.5% (0.4%)	21.1% (1.1%)
	DiscoBAX (ours)	**58% (3.1%)**	**12.4% (0.5%)**	**26.8% (1.3%)**
	Kmeans Data	48.5% (1.7%)	6.6% (0.3%)	17.8% (0.7%)
	Kmeans Embedding	46.5% (2.8%)	7.5% (0.5%)	18.6% (1.2%)
	Levelset BAX	53% (3%)	9.5% (0.9%)	22.5% (1.6%)
	Margin sample	42.5% (4.2%)	7.4% (0.9%)	17.8% (2%)
	qEI	52.5% (2.9%)	11.4% (0.9%)	24.5% (1.6%)
	qPOI	54% (2.8%)	11.9% (0.9%)	25.3% (1.6%)
	qUCB	52.5% (4.7%)	11.3% (1%)	24.4% (2.2%)
	Random	31.5% (2.7%)	5.1% (0.5%)	12.6% (1.2%)
	Soft Uncertainty	31% (4%)	5.2% (0.8%)	12.7% (1.8%)
	Thompson Sampling	35% (3.5%)	7.2% (1.1%)	15.9% (2%)
	Top-K BAX	56% (3.9%)	12.2% (1%)	26.2% (2%)
	Top Uncertainty	49% (2.8%)	9.5% (1.1%)	21.6% (1.7%)
	UCB	49.5% (2.8%)	10.8% (1.1%)	23.1% (1.8%)
SARS-CoV-2	Adversarial BIM	17% (2.4%)	3.5% (0.8%)	7.7% (1.4%)
	BADGE	10.5% (7.8%)	1.8% (1.4%)	4.3% (3.3%)
	Coreset	27.5% (2.6%)	3.4% (0.4%)	9.7% (1%)
	DiscoBAX (ours)	26% (3%)	4% (0.3%)	10.2% (1%)
	Kmeans Data	24% (1.9%)	4.9% (0.3%)	10.8% (0.8%)
	Kmeans Embedding	29.5% (1.7%)	4.8% (0.4%)	11.9% (0.8%)
	Levelset BAX	25% (2.6%)	4.3% (0.4%)	10.3% (1%)
	Margin sample	15% (2.1%)	2.6% (0.4%)	6.2% (0.9%)
	qEI	23.5% (3.3%)	4.1% (0.8%)	9.8% (1.6%)
	qPOI	21.7% (3%)	3.5% (0.4%)	8.7% (1%)
	qUCB	21.5% (2.9%)	4.4% (0.6%)	9.7% (1.3%)
	Random	**32% (3.3%**	**6.2% (0.9%)**	**14% (1.7%)**
	Soft Uncertainty	6.5% (4.9%)	1.6% (1.4%)	3.2% (2.6%)
	Thompson Sampling	19% (1.9%)	2.6% (0.3%)	7.1% (0.7%)
	Top-K BAX	20% (2.8%)	2.8% (0.4%)	7.5% (1.1%)
	Top Uncertainty	18% (2.3%)	3.1% (0.5%)	7.4% (1%)
	UCB	18% (2.4%)	2.7% (0.5%)	7% (1.1%)

Dataset	Method	Top-K recall	Diversity score	Overall score
Leukemia/NK	Adversarial BIM	23.5% (2.2%)	4.9% (0.3%)	10.7% (0.8%)
	BADGE	36.5% (3.9%)	5.7% (0.6%)	14.4% (1.5%)
	Coreset	30% (3.2%)	3.9% (0.4%)	10.9% (1.2%)
	DiscoBAX (ours)	**47% (2.1%)**	**7.1% (0.4%)**	**18.2% (1%)**
	Kmeans Data	26.5% (1.1%)	3.5% (0.2%)	9.6% (0.4%)
	Kmeans Embedding	38% (1.3%)	5.9% (0.4%)	15% (0.7%)
	Levelset BAX	30.5% (4.1%)	5.7% (0.8%)	13.1% (1.8%)
	Margin sample	23.5% (3.1%)	4.1% (0.6%)	9.8% (1.4%)
	qEI	26.5% (3.2%)	4.3% (0.6%)	10.7% (1.4%)
	qPOI	31% (1.5%)	4.8% (0.6%)	12.2% (0.9%)
	qUCB	33% (2.9%)	5.4% (0.7%)	13.4% (1.4%)
	Random	26.5% (3.5%)	4.3% (0.6%)	10.7% (1.5%)
	Soft Uncertainty	29.5% (2.3%)	4.6% (0.4%)	11.6% (0.9%)
	Thompson Sampling	23.5% (2.6%)	4.4% (0.4%)	10.2% (1.1%)
	Top-K BAX	32.5% (2.9%)	4.5% (0.4%)	12.1% (1.1%)
	Top Uncertainty	26% (3.1%)	4.8% (0.6%)	11.2% (1.3%)
	UCB	26.5% (3%)	4.2% (0.6%)	10.5% (1.3%)
Tau protein	Adversarial BIM	16% (1.5%)	**5% (0.3%)**	8.9% (0.6%)
	BADGE	34% (2.8%)	**5% (0.5%)**	**13.1% (1.1%)**
	Coreset	**35% (2.2%)**	4.4% (0.3%)	12.5% (0.9%)
	DiscoBAX (ours)	33% (2.1%)	4.6% (0.4%)	12.3% (0.9%)
	Kmeans Data	27% (1.1%)	3.3% (0.2%)	9.5% (0.5%)
	Kmeans Embedding	30% (2.6%)	4.4% (0.4%)	11.5% (1%)
	Levelset BAX	22.5% (2.4%)	4.6% (0.5%)	10.2% (1.1%)
	Margin sample	32% (3.3%)	4.9% (0.5%)	12.5% (1.2%)
	qEI	32.1% (2.8%)	4.3% (0.6%)	11.7% (1.3%)
	qPOI	31% (2.5%)	4.5% (0.5%)	11.8% (1.1%)
	qUCB	31.2% (2.6%)	4.4% (0.4%)	11.7% (1%)
	Random	25% (3.3%)	3.9% (0.5%)	9.9% (1.3%)
	Soft Uncertainty	27% (2.4%)	4.6% (0.4%)	11.1% (0.9%)
	Thompson Sampling	24.5% (2.4%)	3.8% (0.3%)	9.7% (0.9%)
	Top-K BAX	33.5% (2.4%)	4.8% (0.4%)	12.7% (1%)
	Top Uncertainty	29.5% (1.2%)	4.1% (0.2%)	11% (0.5%)
	UCB	32% (2.7%)	4.4% (0.3%)	11.8% (1%)

Table C.2: GeneDisco experiment - Hyperparameter selection

Method	Hyperparameter value	Top-K recall	Diversity score	Overall score
Top-K BAX	2	32% (3.6%)	4% (0.5%)	11.3% (1.3%)
	3	32% (3.3%)	4.3% (0.5%)	11.8% (1.1%)
	5	33% (2.4%)	4.4% (0.4%)	**12.1% (0.9%)**
	10	30% (3.2%)	4.2% (0.4%)	11.2% (1.3%)
Levelset BAX	0.8	19% (2.1%)	3.4% (0.3%)	8% (0.8%)
	1	30% (4.3%)	5.4% (0.7%)	**12.7% (1.8%)**
	1.2	21% (1.3%)	4.1% (0.6%)	9.3% (0.9%)
	1.5	29% (0.7%)	5.4% (0.5%)	12.5% (0.6%)
DiscoBAX	2	36% (5.3%)	4.8% (0.8%)	13.1% (2%)
	3	32% (2.6%)	4.1% (0.5%)	11.4% (1.1%)
	5	37.5% (2.7%)	5.4% (0.4%)	14.2% (1%)
	10	38% (1.8%)	5.5% (0.3%)	**14.5% (0.8%)**

Table C.3: Molecular generation with CVAE - Model architecture details

Component	Description
Encoder	• 3 consecutive 1D convolutional layers with 9,9 and 10 filters respectively, kernel sizes 9,9 and 11 respectively, with batch norm and RELU activations after each convolutional layer • Continuous latent space of dimension 56
Decoder	• A stack of 3 Gated recurrent unit (GRU) layers [400] with hidden dimension 500, with dropout 0.2 (between layers) and RELU activations except for the last layer which has a softmax activation over the SMILES vocabulary. • At each step, the character generated at the previous step is concatenated with the latent embedding and fed as input
Property network	• 3-layer feedforward network with 1,000 units each • RELU activations (after each layer except the final one) and dropout 0.2

Table C.4: Molecular generation with CVAE - Gradient ascent results.

Decoder uncertainty	Penalized logP Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑	Validity (%) ↑
None	5.1 ± 0.2	4.6 ± 0.2	4.1 ± 0.1	1.6 ± 0.5	$4.1\% \pm 0.4\%$
NLLP	4.8 ± 0.1	4.6 ± 0	4.4 ± 0	4.3 ± 0	$\mathbf{56.6\% \pm 0.9\%}$
TI-MI	3.3 ± 2	4.6 ± 0.1	4.4 ± 0.1	0.8 ± 1.8	$4.1\% \pm 0.5\%$
IS-MI	$\mathbf{5.8 \pm 0.2}$	$\mathbf{5.4 \pm 0.1}$	$\mathbf{5.1 \pm 0.1}$	$\mathbf{4.9 \pm 0.1}$	$21.5\% \pm 1\%$

Table C.5: Molecular generation with CVAE - Bayesian Optimization results. NA values in the table corresponds to no valid molecule decoded across the 10 independent runs. 'Validity' measures the proportion of generated molecules that correspond to valid SMILES expressions. 'Unicity' corresponds to the ratio of the number of distinct generated molecules to the total number of generated molecules. 'Novelty' is defined as the proportion of generated molecules that were not present in the training data.

Search bounds	Decoder uncertainty	Penalized logP Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑	Validity (%) ↑	Unicity (%) ↑	Novelty (%) ↑
5	None	4.0 ± 0.2	3.5 ± 0.2	3.2 ± 0.2	2.5 ± 0.2	$21.9\% \pm 1.4\%$	$100\% \pm 0\%$	$100\% \pm 0\%$
	NLLP	4.2 ± 0.2	3.6 ± 0.1	3.2 ± 0.1	2.7 ± 0.1	$29.6\% \pm 1.3\%$	$100\% \pm 0\%$	$100\% \pm 0\%$
	TI-MI	4.1 ± 0.2	3.5 ± 0.2	3.1 ± 0.1	2.3 ± 0.1	$21.0\% \pm 0.8\%$	$100\% \pm 0\%$	$100\% \pm 0\%$
	IS-MI	$\mathbf{4.5 \pm 0.2}$	$\mathbf{3.7 \pm 0.2}$	$\mathbf{3.5 \pm 0.2}$	$\mathbf{3.0 \pm 0.1}$	$\mathbf{33.2\% \pm 1.8\%}$	$100\% \pm 0\%$	$100\% \pm 0\%$
10	None	3.9 ± 1.1	-1.9 ± 6.9	-14.4 ± 4	-2.3 ± 2.8	$1.1\% \pm 0.4\%$	$80\% \pm 8\%$	$80\% \pm 8\%$
	NLLP	2.9 ± 0.7	0.2 ± 1.4	2.3 ± 0.8	0.5 ± 0.8	$2.8\% \pm 0.7\%$	$60\% \pm 16\%$	$60\% \pm 16\%$
	TI-MI	5.9 ± 3.3	-1.9 ± 1.7	0.2 ± 0.7	1.1 ± 1.5	$1.6\% \pm 0.4\%$	$80\% \pm 13\%$	$80\% \pm 13\%$
	IS-MI	$\mathbf{6.6 \pm 0.5}$	$\mathbf{4.6 \pm 0.6}$	$\mathbf{3.6 \pm 0.3}$	$\mathbf{1.6 \pm 0.8}$	$\mathbf{10.6\% \pm 0.8\%}$	$\mathbf{99\% \pm 1\%}$	$\mathbf{100\% \pm 0\%}$
15	None	10.3 ± 3.9	-3.0 ± 2.7	NA	5.0 ± 2.6	$1.0\% \pm 0.3\%$	$80\% \pm 11\%$	$80\% \pm 11\%$
	NLLP	3.9 ± 2.3	-4.6 ± 4.9	NA	0.8 ± 1.2	$1.0\% \pm 0.3\%$	$64\% \pm 16\%$	$64\% \pm 16\%$
	TI-MI	6.7 ± 3.6	0.0 ± 1.7	NA	6.4 ± 3.9	$1.1\% \pm 0.3\%$	$73\% \pm 15\%$	$73\% \pm 15\%$
	IS-MI	$\mathbf{27.6 \pm 2.1}$	$\mathbf{15.4 \pm 3.9}$	$\mathbf{7.8 \pm 3.2}$	$\mathbf{9.9 \pm 1.3}$	$\mathbf{5.5\% \pm 0.7\%}$	$\mathbf{96\% \pm 2\%}$	$\mathbf{100\% \pm 0\%}$

Table C.6: Molecular generation with CVAE - Impact of the choice of decoder uncertainty threshold on Bayesian Optimization results. This analysis focuses on the IS-MI estimator only. NA values in the table corresponds to no valid molecule decoded. The second column represents the percentile of the IS-MI estimator values on the training data used to define the threshold for censored Bayesian Optimization.

Search bounds	Uncertainty threshold	Penalized logP				Validity
		Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑	(%) ↑
5	None	4.0 ± 0.2	3.5 ± 0.2	3.2 ± 0.2	2.5 ± 0.2	$21.9\% \pm 1.4\%$
	Median	4.2 ± 0.1	$\mathbf{3.9 \pm 0.1}$	$\mathbf{3.7 \pm 0.1}$	$\mathbf{3.4 \pm 0.1}$	$\mathbf{54.8\% \pm 1.2\%}$
	P90	4.3 ± 0.1	3.8 ± 0.1	3.6 ± 0.1	3.3 ± 0.1	$48.2\% \pm 2.3\%$
	P95	4.3 ± 0.1	3.8 ± 0.2	3.6 ± 0.1	3.2 ± 0.1	$44.0\% \pm 1.7\%$
	P99	$\mathbf{4.5 \pm 0.2}$	3.7 ± 0.2	3.5 ± 0.2	3 ± 0.1	$33.2\% \pm 1.8\%$
	Max	4.3 ± 0.2	3.6 ± 0.2	3.1 ± 0.2	2.6 ± 0.2	$26.9\% \pm 1.8\%$
10	None	3.9 ± 1.2	-12.7 ± 6.9	-14.4 ± 4	-2.3 ± 2.8	$1.1\% \pm 0.4\%$
	Median	$\mathbf{7.6 \pm 0.7}$	4.7 ± 0.5	3.8 ± 0.3	2.5 ± 0.4	$\mathbf{12.4\% \pm 0.7\%}$
	P90	$\mathbf{7.6 \pm 0.7}$	$\mathbf{4.8 \pm 0.5}$	3.6 ± 0.3	2.5 ± 0.4	$11.1\% \pm 0.7\%$
	P95	7.0 ± 0.9	4.5 ± 0.6	$\mathbf{3.9 \pm 0.4}$	$\mathbf{2.6 \pm 0.4}$	$12.0\% \pm 0.7\%$
	P99	6.6 ± 0.6	4.6 ± 0.6	3.6 ± 0.3	1.6 ± 0.8	$10.6\% \pm 0.8\%$
	Max	5.7 ± 0.8	3.8 ± 0.3	2.9 ± 0.3	0.9 ± 0.7	$9.1\% \pm 0.8\%$
15	None	10.3 ± 4.3	-3.0 ± 2.7	NA	5.0 ± 2.6	$1.0\% \pm 0.3\%$
	Median	21.6 ± 3.9	10.3 ± 4.2	8.3 ± 3.8	6.1 ± 1.9	$5.8\% \pm 0.8\%$
	P90	24.0 ± 3.3	14.1 ± 4.2	7.8 ± 3.4	7.6 ± 1.7	$5.8\% \pm 0.7\%$
	P95	26.5 ± 2.5	$\mathbf{17.2 \pm 4.1}$	$\mathbf{8.9 \pm 4.1}$	8.4 ± 1.6	$\mathbf{6.2\% \pm 0.7\%}$
	P99	$\mathbf{27.6 \pm 2.2}$	15.4 ± 3.9	7.8 ± 3.2	$\mathbf{9.9 \pm 1.3}$	$5.5\% \pm 0.7\%$
	Max	23.7 ± 3.3	13.6 ± 3.7	7.1 ± 3.8	8.2 ± 2.2	$5.5\% \pm 0.7\%$

Table C.7: Molecular generation with JT-VAE - Model architecture details. All components are identical to the JT-VAE architecture used for constrained optimization from Jin et al. [250], except for the dropout layers detailed below

Component	Description
Encoder	• Junction tree encoder and molecular graph encoder • Continuous latent space of dimension 56
Decoder	• Junction tree decoder with dropout layer (0.2 drop rate) applied to the input and the output of the GRU used for message passing, and dropout layer (0.2 rate) applied right before the final layer for both the topology prediction and node prediction networks • Molecular graph decoder with dropout layer (0.2 rate) applied right before the final layer of the subgraph prediction network
Property network	• 2-layer feedforward network with 450 units and 1 unit resp., tanh activation for the first layer, no activation for the second, and dropout 0.2 before both layer

Table C.8: Molecular generation - Top optimization performance. We achieve state-of-the-art performance on the molecular generation task using a JT-VAE model jointly trained with an auxiliary network predicting penalized logP from latent embeddings (as per §3.3 of [250]) and performing gradient ascent as described in § 5.3.2. Results were obtained by embedding in latent space 100 points selected at random from the test set and then performing 100 gradient updates with $\alpha = 200$. We report mean performance over 10 runs, as well as the best generated molecules across these 10 runs.

Model	Optimization method	Penalized logP		
		Top 1 ↑	Top 2 ↑	Top 3 ↑
JT-VAE [250]	Bayesian Optimization	5.30	4.93	4.49
MolDQN [251]	Reinforcement learning	11.84	11.84	11.82
GraphAF [252]	Reinforcement learning	12.23	11.29	11.05
CCGF [247]	Chance-constrained optimization	12.32	11.79	11.61
ChemBO [402]	Bayesian Optimization	18.39	-	-
JT-VAE [253]	Bayesian Optim. & retraining (median of 5 runs)	21.20	15.34	15.34
JT-VAE [253]	Bayesian Optim. & retraining (best over 5 runs)	27.84	27.59	27.21
JT-VAE (ours)	Gradient ascent (mean of 10 runs)	23.65	21.17	19.45
JT-VAE (ours)	Gradient ascent (best over 10 runs)	**30.81**	**30.00**	**29.82**

Table C.9: Molecular generation with JT-VAE - Gradient ascent results. The 'Uncertainty threshold' column represents the percentile of training values used to define the uncertainty threshold.

$\alpha = 100$

Decoder uncertainty	Uncertainty threshold	Penalized logP - Before filters				Quality	Penalized logP - Passing filters			
		Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑	top 10 (%) ↑	Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑
None	None	$\mathbf{22.4 \pm 0.9}$	$\mathbf{19.9 \pm 0.6}$	$\mathbf{18.7 \pm 0.4}$	$\mathbf{16.6 \pm 0.3}$	$3\% \pm 2\%$	4.3 ± 2.2	0.0 ± 0.0	0.0 ± 0.0	0.4 ± 0.2
NLLP	P95	3.4 ± 0.1	2.9 ± 0.1	2.6 ± 0.1	2.4 ± 0.1	$71\% \pm 4\%$	3.3 ± 0.2	2.7 ± 0.1	2.5 ± 0.1	1.8 ± 0.1
	P99	3.8 ± 0.1	3.4 ± 0.1	3.2 ± 0.1	3.0 ± 0.1	$\mathbf{89\% \pm 3\%}$	3.8 ± 0.1	3.3 ± 0.1	3.2 ± 0.1	2.7 ± 0.1
	Max	4.5 ± 0.1	4.2 ± 0.1	4.0 ± 0.1	3.8 ± 0.1	$82\% \pm 4\%$	4.4 ± 0.1	4.0 ± 0.1	3.8 ± 0.1	3.1 ± 0.1
IS-MI	P95	11.4 ± 1.4	8.1 ± 0.3	7.7 ± 0.2	7.6 ± 0.2	$81\% \pm 5\%$	8.3 ± 0.3	7.7 ± 0.2	$\mathbf{7.4 \pm 0.2}$	$\mathbf{5.8 \pm 0.3}$
	P99	17.5 ± 1.4	13.3 ± 0.9	11.4 ± 0.7	10.3 ± 0.5	$48\% \pm 6\%$	$\mathbf{9.9 \pm 0.7}$	$\mathbf{8.4 \pm 0.2}$	6.9 ± 0.8	3.9 ± 0.4
	Max	19.4 ± 1.1	16.7 ± 1.0	15.1 ± 0.9	13.0 ± 0.6	$19\% \pm 4\%$	$\mathbf{9.9 \pm 1.1}$	4.9 ± 1.5	2.6 ± 1.2	1.8 ± 0.3

$\alpha = 200$

Decoder uncertainty	Uncertainty threshold	Penalized logP - Before filters				Quality	Penalized logP - Passing filters			
		Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑	top 10 (%) ↑	Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑
None	None	$\mathbf{23.7 \pm 1.3}$	$\mathbf{21.2 \pm 0.8}$	$\mathbf{19.5 \pm 0.8}$	$\mathbf{17.0 \pm 0.6}$	$1\% \pm 1\%$	1.2 ± 1.2	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.1
NLLP	P95	3.0 ± 0.1	2.8 ± 0.1	2.7 ± 0.1	2.5 ± 0.05	$82\% \pm 6\%$	3.0 ± 0.1	2.7 ± 0.1	2.6 ± 0.1	2.0 ± 0.2
	P99	3.3 ± 0.1	3.0 ± 0.1	2.8 ± 0.1	2.6 ± 0.1	$80\% \pm 4\%$	3.2 ± 0.1	2.9 ± 0.1	2.7 ± 0.1	2.1 ± 0.1
	Max	3.6 ± 0.1	3.2 ± 0.1	3.0 ± 0.1	2.7 ± 0.1	$81\% \pm 4\%$	3.6 ± 0.1	3.1 ± 0.1	3.0 ± 0.1	2.2 ± 0.1
IS-MI	P95	8.4 ± 0.8	6.8 ± 0.4	6.4 ± 0.4	6.0 ± 0.3	$\mathbf{89\% \pm 3\%}$	7.7 ± 0.7	6.5 ± 0.3	6.1 ± 0.3	$\mathbf{5.3 \pm 0.3}$
	P99	15.3 ± 1.6	9.5 ± 1.0	7.2 ± 0.3	7.1 ± 0.4	$69\% \pm 4\%$	7.6 ± 0.3	$\mathbf{6.8 \pm 0.3}$	$\mathbf{6.3 \pm 0.3}$	4.0 ± 0.1
	Max	20.5 ± 1.3	17.3 ± 1.4	13.8 ± 1.2	11.4 ± 0.8	$31\% \pm 7\%$	$\mathbf{8.6 \pm 1.3}$	6.1 ± 1.1	3.4 ± 1.1	2.3 ± 0.5

Table C.10: Molecular generation with JT-VAE - Bayesian Optimization results.

Decoder uncertainty	Penalized logP - Before filters				Quality	Penalized logP - Passing filters			
	Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑	top 10 (%) ↑	Top 1 ↑	Top 2 ↑	Top 3 ↑	Avg. top 10 ↑
None	$\mathbf{18.6 \pm 1.5}$	$\mathbf{14.1 \pm 1.8}$	$\mathbf{11.5 \pm 1.2}$	$\mathbf{9.6 \pm 0.7}$	$18\% \pm 6\%$	7.1 ± 2.2	3.0 ± 1.1	2.2 ± 1.1	1.4 ± 0.4
NLLP	15.7 ± 1.5	12.2 ± 0.8	9.7 ± 0.5	8.7 ± 0.4	$23\% \pm 5\%$	7.8 ± 0.6	4.6 ± 1.0	1.1 ± 0.7	1.5 ± 0.3
IS-MI	14.3 ± 1.7	9.2 ± 1.0	7.6 ± 0.9	6.3 ± 0.4	$\mathbf{47\% \pm 3\%}$	$\mathbf{9.7 \pm 2.1}$	$\mathbf{4.8 \pm 0.4}$	$\mathbf{4.2 \pm 0.3}$	$\mathbf{2.1 \pm 0.3}$

References

[1] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[2] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[3] Aditya Ramesh et al. “Zero-Shot Text-to-Image Generation”. In: *ArXiv* abs/2102.12092 (2021).

[4] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *International Conference on Machine Learning*. 2021.

[5] Aäron van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: *ArXiv* abs/1609.03499 (2016).

[6] Alec Radford et al. *Robust Speech Recognition via Large-Scale Weak Supervision*. arXiv:2212.04356 [cs, eess]. Dec. 2022. URL: http://arxiv.org/abs/2212.04356 (visited on 06/17/2023).

[7] Pengcheng Yin and Graham Neubig. *A Syntactic Neural Model for General-Purpose Code Generation*. arXiv:1704.01696 [cs]. Apr. 2017. URL: http://arxiv.org/abs/1704.01696 (visited on 06/17/2023).

[8] Yujia Li et al. “Competition-level code generation with AlphaCode”. In: *Science* 378 (2022), pp. 1092–1097.

[9] Guillaume Lample and François Charton. *Deep Learning for Symbolic Mathematics*. arXiv:1912.01412 [cs]. Dec. 2019. URL: http://arxiv.org/abs/1912.01412 (visited on 06/17/2023).

[10] Aitor Lewkowycz et al. *Solving Quantitative Reasoning Problems with Language Models*. arXiv:2206.14858 [cs]. June 2022. URL: http://arxiv.org/abs/2206.14858 (visited on 05/26/2023).

[11] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. “Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multi-Layer Calorimeters”. In: *Physical Review Letters* 120.4 (Jan. 2018). arXiv:1705.02355 [hep-ex, physics:hep-ph, stat], p. 042003. URL: http://arxiv.org/abs/1705.02355 (visited on 06/17/2023).

[12] ATLAS Collaboration. *AtlFast3: the next generation of fast simulation in ATLAS*. arXiv:2109.02551 [hep-ex]. Feb. 2022. URL: http://arxiv.org/abs/2109.02551 (visited on 05/26/2023).

[13] Xingjian Shi et al. *Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model*. arXiv:1706.03458 [cs]. Oct. 2017. URL: http://arxiv.org/abs/1706.03458 (visited on 06/18/2023).

[14] Suman V. Ravuri et al. "Skilful precipitation nowcasting using deep generative models of radar". In: *Nature* 597 (2021), pp. 672–677.

[15] Nathan Killoran et al. *Generating and designing DNA with deep generative models*. arXiv:1712.06148 [cs, q-bio, stat]. Dec. 2017. URL: http://arxiv.org/abs/1712.06148 (visited on 06/18/2023).

[16] John M. Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (2021), pp. 583–589.

[17] Rafael Gómez-Bombarelli et al. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". In: *ACS Central Science* 4 (2016), pp. 268–276.

[18] Fergus Imrie et al. "Deep Generative Models for 3D Linker Design". eng. In: *Journal of Chemical Information and Modeling* 60.4 (Apr. 2020), pp. 1983–1995.

[19] Joseph L. Watson et al. *Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models*. en. Pages: 2022.12.09.519842 Section: New Results. Dec. 2022. URL: https://www.biorxiv.org/content/10.1101/2022.12.09.519842v1 (visited on 06/18/2023).

[20] Brian L. Hie et al. "Learning the language of viral evolution and escape". In: *Science* 371 (2020), pp. 284–288.

[21] M. Cyrus Maher et al. "Predicting the mutational drivers of future SARS-CoV-2 variants of concern". In: *Science Translational Medicine* 14.633 (Jan. 2022). Publisher: American Association for the Advancement of Science, eabk3445. URL: https://www.science.org/doi/10.1126/scitranslmed.abk3445 (visited on 06/18/2023).

[22] David Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, 2019.

[23] Jeff Heaton. "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning". In: *Genetic Programming and Evolvable Machines* 19 (2017), pp. 305–307.

[24] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[25] Ian J. Goodfellow et al. *Generative Adversarial Networks*. arXiv:1406.2661 [cs, stat]. June 2014. URL: http://arxiv.org/abs/1406.2661.

[26] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". en. In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. URL: https://doi.org/10.1007/BF02478259 (visited on 06/12/2023).

[27] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (visited on 06/12/2023).

[28] Alec Radford et al. “Improving Language Understanding by Generative Pre-Training”. In: (2018).

[29] H. P. J. Buermans and J. T. den Dunnen. “Next generation sequencing technology: Advances and applications”. en. In: *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease*. From genome to function 1842.10 (Oct. 2014), pp. 1932–1941. URL: https://www.sciencedirect.com/science/article/pii/S092544391400180X (visited on 07/10/2023).

[30] Erika Check Hayden. “Technology: The $1,000 genome”. en. In: *Nature* 507.7492 (Mar. 2014). Number: 7492 Publisher: Nature Publishing Group, pp. 294–295. URL: https://www.nature.com/articles/507294a (visited on 07/10/2023).

[31] Bruce Alberts et al. *Molecular Biology of the Cell*. 4th. Garland Science, 2002.

[32] Frances H. Arnold. “Directed Evolution: Bringing New Chemistry to Life”. In: *Angewandte Chemie International Edition* 57.16 (2018). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.201708408, pp. 4143–4148. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201708408 (visited on 01/06/2023).

[33] Po-Ssu Huang, Scott E. Boyken, and David Baker. “The coming of age of de novo protein design”. en. In: *Nature* 537.7620 (Sept. 2016). Number: 7620 Publisher: Nature Publishing Group, pp. 320–327. URL: https://www.nature.com/articles/nature19946 (visited on 02/19/2023).

[34] Melissa J. Landrum and Brandi L. Kattman. “ClinVar at five years: Delivering on the promise”. eng. In: *Human Mutation* 39.11 (Nov. 2018), pp. 1623–1630.

[35] Fritz Obermeyer et al. “Analysis of 6.4 million SARS-CoV-2 genomes identifies mutations associated with fitness”. eng. In: *Science (New York, N.Y.)* 376.6599 (June 2022), pp. 1327–1332.

[36] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. 2014. arXiv: 1401.4082 [stat.ML].

[37] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[38] Thomas A. Hopf et al. “Mutation effects predicted from sequence co-variation”. eng. In: *Nature Biotechnology* 35.2 (Feb. 2017), pp. 128–135.

[39] Ethan C Alley et al. “Unified rational protein engineering with sequence-only deep representation learning”. In: *bioRxiv* (2019), p. 589333.

[40] 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 (Apr. 2021). Publisher: Proceedings of the National Academy of Sciences, e2016239118. URL: https://www.pnas.org/doi/10.1073/pnas.2016239118 (visited on 03/02/2023).

[41] Joshua Meier et al. “Language models enable zero-shot prediction of the effects of mutations on protein function”. In: *bioRxiv* (2021). eprint: https://www.biorxiv.org/content/early/2021/07/10/2021.07.09.450648.full.pdf. URL: https://www.biorxiv.org/content/early/2021/07/10/2021.07.09.450648.

[42] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. “Grammar Variational Autoencoder”. In: *ICML*. 2017.

[43] Michael I. Jordan et al. “An Introduction to Variational Methods for Graphical Models”. en. In: *Machine Learning* 37.2 (Nov. 1999), pp. 183–233. URL: https://doi.org/10.1023/A:1007665907178 (visited on 07/04/2023).

[44] Martin J. Wainwright and Michael I. Jordan. “Graphical Models, Exponential Families, and Variational Inference”. English. In: *Foundations and Trends® in Machine Learning* 1.1–2 (Nov. 2008). Publisher: Now Publishers, Inc., pp. 1–305. URL: https://www.nowpublishers.com/article/Details/MAL-001 (visited on 07/04/2023).

[45] Zhilin Yang et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. arXiv:1906.08237 [cs]. Jan. 2020. URL: http://arxiv.org/abs/1906.08237 (visited on 07/05/2023).

[46] Mohammad Bavarian et al. *Efficient Training of Language Models to Fill in the Middle*. arXiv:2207.14255 [cs]. July 2022. URL: http://arxiv.org/abs/2207.14255 (visited on 07/05/2023).

[47] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. arXiv:1909.11942 [cs]. Feb. 2020. URL: http://arxiv.org/abs/1909.11942 (visited on 07/05/2023).

[48] Kevin Clark et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. arXiv:2003.10555 [cs]. Mar. 2020. URL: http://arxiv.org/abs/2003.10555 (visited on 07/05/2023).

[49] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[50] OpenAI. *GPT-4 Technical Report*. arXiv:2303.08774 [cs]. Mar. 2023. URL: http://arxiv.org/abs/2303.08774 (visited on 06/12/2023).

[51] Ruibin Xiong et al. *On Layer Normalization in the Transformer Architecture*. arXiv:2002.04745 [cs, stat]. June 2020. URL: http://arxiv.org/abs/2002.04745 (visited on 07/05/2023).

[52] Jianlin Su et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. arXiv:2104.09864 [cs]. Aug. 2022. URL: http://arxiv.org/abs/2104.09864 (visited on 07/05/2023).

[53] Ofir Press, Noah A. Smith, and Mike Lewis. *Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation*. 2021. arXiv: 2108.12409 [cs.CL].

[54] Jonathan Ho et al. *Axial Attention in Multidimensional Transformers*. arXiv:1912.12180 [cs]. Dec. 2019. URL: http://arxiv.org/abs/1912.12180 (visited on 02/20/2023).

[55] Alex Kendall and Yarin Gal. *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* 2017. arXiv: 1703.04977 [cs.CV].

[56] Nicholas Carlini and David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2016. arXiv: 1608.04644 [cs.CR].

[57] Reuben Feinman et al. *Detecting Adversarial Samples from Artifacts*. 2017. arXiv: 1703.00410 [stat.ML].

[58] Lewis Smith and Yarin Gal. *Understanding Measures of Uncertainty for Adversarial Example Detection*. 2018. arXiv: 1803.08533 [stat.ML].

[59] Andrey Malinin and Mark Gales. *Uncertainty in Structured Prediction*. 2020. arXiv: 2002.07650 [stat.ML].

[60] Jonas Mockus. "Bayesian Approach to Global Optimization: Theory and Applications". In: 1989.

[61] Bobak Shahriari et al. "Taking the human out of the loop: A review of Bayesian optimization". English (US). In: *Proceedings of the IEEE* 104.1 (Jan. 2016). Publisher: Institute of Electrical and Electronics Engineers Inc., pp. 148–175. URL: https://collaborate.princeton.edu/en/publications/taking-the-human-out-of-the-loop-a-review-of-bayesian-optimizatio (visited on 07/06/2023).

[62] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. arXiv:1206.2944 [cs, stat]. Aug. 2012. URL: http://arxiv.org/abs/1206.2944 (visited on 07/06/2023).

[63] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. en. Nov. 2005. URL: https://direct.mit.edu/books/book/2320/Gaussian-Processes-for-Machine-Learning (visited on 02/19/2023).

[64] Radford M. Neal. "Bayesian Learning for Neural Networks". In: 1995.

[65] David John Cameron MacKay. "Bayesian Interpolation". In: *Neural Computation* 4 (1992), pp. 415–447.

[66] David John Cameron MacKay. "Information Theory, Inference, and Learning Algorithms". In: *IEEE Transactions on Information Theory* 50 (2004), pp. 2544–2545.

[67] Jonas Močkus. "On Bayesian methods for seeking the extremum". In: *Optimization techniques IFIP technical conference*. Springer. 1975, pp. 400–404.

[68] Niranjan Srinivas et al. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In: *IEEE Transactions on Information Theory* 58.5 (May 2012). arXiv:0912.3995 [cs], pp. 3250–3265. URL: http://arxiv.org/abs/0912.3995 (visited on 07/06/2023).

[69] H. J. Kushner. "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise". In: *Journal of Basic Engineering* 86.1 (Mar. 1964), pp. 97–106. URL: https://doi.org/10.1115%2F1.3653121.

[70] Zi Wang et al. "Batched High-dimensional Bayesian Optimization via Structural Kernel Learning". en. In: *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2017, pp. 3656–3664. URL: https://proceedings.mlr.press/v70/wang17h.html (visited on 07/06/2023).

[71] Burr Settles. *Active Learning Literature Survey*. en. Technical Report. Accepted: 2012-03-15T17:23:56Z. University of Wisconsin-Madison Department of Computer Sciences, 2009. URL: https://minds.wisconsin.edu/handle/1793/60660 (visited on 07/06/2023).

[72] David D. Lewis and Jason Catlett. "Heterogeneous Uncertainty Sampling for Supervised Learning". en. In: *Machine Learning Proceedings 1994*. Ed. by William W. Cohen and Haym Hirsh. San Francisco (CA): Morgan Kaufmann, Jan. 1994, pp. 148–156. URL: https://www.sciencedirect.com/science/article/pii/B978155860335650026X (visited on 07/07/2023).

[73] H. S. Seung, M. Opper, and H. Sompolinsky. "Query by committee". In: *Proceedings of the fifth annual workshop on Computational learning theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, July 1992, pp. 287–294. URL: https://dl.acm.org/doi/10.1145/130385.130417 (visited on 07/07/2023).

[74] David J. C. MacKay. "Information-based objective functions for active data selection". en. In: *Neural Computation* 4.4 (July 1992). Number: 4 Publisher: MIT Press, pp. 590–604. URL: https://resolver.caltech.edu/CaltechAUTHORS:MACnc92c (visited on 07/07/2023).

[75] Neil Houlsby et al. *Bayesian Active Learning for Classification and Preference Learning*. arXiv:1112.5745 [cs, stat]. Dec. 2011. URL: http://arxiv.org/abs/1112.5745 (visited on 07/07/2023).

[76] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. "Deep Bayesian Active Learning with Image Data". en. In: *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2017, pp. 1183–1192. URL: https://proceedings.mlr.press/v70/gal17a.html (visited on 07/07/2023).

[77] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. *BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning*. arXiv:1906.08158 [cs, stat]. Oct. 2019. URL: http://arxiv.org/abs/1906.08158 (visited on 07/07/2023).

[78] Robert Pinsler et al. *Bayesian Batch Active Learning as Sparse Subset Approximation*. arXiv:1908.02144 [cs, stat]. Feb. 2021. URL: http://arxiv.org/abs/1908.02144 (visited on 07/07/2023).

[79] Jeremy M. Berg, John L Tymoczko, and Lubert Stryer. "Biochemistry 5th ed". In: 2002.

[80] J. Drews. "Drug discovery: a historical perspective". eng. In: *Science (New York, N.Y.)* 287.5460 (Mar. 2000), pp. 1960–1964.

[81] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of Chemical Information and Computer Sciences* 28.1 (Feb. 1988). Publisher: American Chemical Society, pp. 31–36. URL: https://doi.org/10.1021/ci00057a005 (visited on 07/07/2023).

[82] Christopher M. Dobson. "Chemical space and biology". en. In: *Nature* 432.7019 (Dec. 2004). Number: 7019 Publisher: Nature Publishing Group, pp. 824–828. URL: https://www.nature.com/articles/nature03192 (visited on 07/07/2023).

[83] Leonardo G. Ferreira et al. "Molecular docking and structure-based drug design strategies". eng. In: *Molecules (Basel, Switzerland)* 20.7 (July 2015), pp. 13384–13421.

[84] Francis Crick. "Central Dogma of Molecular Biology". en. In: *Nature* 227.5258 (Aug. 1970). Number: 5258 Publisher: Nature Publishing Group, pp. 561–563. URL: https://www.nature.com/articles/227561a0 (visited on 07/07/2023).

[85] Thomas R. Cech and Joan A. Steitz. "The noncoding RNA revolution-trashing old rules to forge new ones". eng. In: *Cell* 157.1 (Mar. 2014), pp. 77–94.

[86] John L. Rinn and Howard Y. Chang. "Genome regulation by long noncoding RNAs". eng. In: *Annual Review of Biochemistry* 81 (2012), pp. 145–166.

[87] The UniProt Consortium. "UniProt: a worldwide hub of protein knowledge". en. In: *Nucleic Acids Research* 47.D1 (Jan. 2019), pp. D506–D515. URL: https://academic.oup.com/nar/article/47/D1/D506/5160987 (visited on 10/09/2020).

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

[89] J Drew Thompson, Desmond G. Higgins, and Toby J. Gibson. "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." In: *Nucleic acids research* 22 22 (1994), pp. 4673–80.

[90] Robert C. Edgar. "Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny". en. In: *Nature Communications* 13.1 (Nov. 2022). Number: 1 Publisher: Nature Publishing Group, p. 6968. URL: https://www.nature.com/articles/s41467-022-34630-w (visited on 05/11/2023).

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

[92] Sean R. Eddy. "Accelerated Profile HMM Searches". en. In: *PLoS Computational Biology* 7.10 (Oct. 2011). Ed. by William R. Pearson, e1002195. URL: https://dx.plos.org/10.1371/journal.pcbi.1002195 (visited on 10/09/2020).

[93] Cédric Notredame and Desmond G. Higgins. "SAGA: Sequence Alignment by Genetic Algorithm". In: *Nucleic Acids Research* 24.8 (Apr. 1996), pp. 1515–1524. URL: https://doi.org/10.1093/nar/24.8.1515 (visited on 07/07/2023).

[94] Lusheng Wang and Tao Jiang. "On the Complexity of Multiple Sequence Alignment". In: *Journal of Computational Biology* 1.4 (Jan. 1994). Publisher: Mary Ann Liebert, Inc., publishers, pp. 337–348. URL: https://www.liebertpub.com/doi/10.1089/cmb.1994.1.337 (visited on 07/07/2023).

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

[96] Jeffrey I. Boucher et al. "Viewing protein fitness landscapes through a next-gen lens". eng. In: *Genetics* 198.2 (Oct. 2014), pp. 461–471.

[97] Carlos L. Araya and Douglas M. Fowler. "Deep mutational scanning: assessing protein function on a massive scale". eng. In: *Trends in Biotechnology* 29.9 (Sept. 2011), pp. 435–442.

[98] Martin Jinek et al. "A programmable dual-RNA-guided DNA endonuclease in adaptive bacterial immunity". eng. In: *Science (New York, N.Y.)* 337.6096 (Aug. 2012), pp. 816–821.

[99] Jennifer A. Doudna and Emmanuelle Charpentier. "Genome editing. The new frontier of genome engineering with CRISPR-Cas9". eng. In: *Science (New York, N.Y.)* 346.6213 (Nov. 2014), p. 1258096.

[100] David Benjamin Turitz Cox, Randall Jeffrey Platt, and Feng Zhang. "Therapeutic genome editing: prospects and challenges". en. In: *Nature Medicine* 21.2 (Feb. 2015). Number: 2 Publisher: Nature Publishing Group, pp. 121–131. URL: https://www.nature.com/articles/nm.3793 (visited on 07/07/2023).

[101] wwPDB consortium. "Protein Data Bank: the single global archive for 3D macromolecular structure data". In: *Nucleic Acids Research* 47.D1 (Jan. 2019), pp. D520–D528. URL: https://doi.org/10.1093/nar/gky949 (visited on 04/28/2023).

[102] Lorna Richardson et al. "MGnify: the microbiome sequence data analysis resource in 2023". In: *Nucleic Acids Research* 51.D1 (Jan. 2023), pp. D753–D759. URL: https://doi.org/10.1093/nar/gkac1080.

[103] Chloe Hsu et al. "Learning inverse folding from millions of predicted structures". en. In: *Proceedings of the 39th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, June 2022, pp. 8946–8970. URL: https://proceedings.mlr.press/v162/hsu22a.html (visited on 02/03/2023).

[104] Zaixiang Zheng et al. *Structure-informed Language Models Are Protein Designers*. arXiv:2302.01649 [cs]. Feb. 2023. URL: http://arxiv.org/abs/2302.01649 (visited on 06/18/2023).

[105] Pauline C Ng and Steven Henikoff. "Predicting deleterious amino acid substitutions". In: *Genome research* 11.5 (2001), pp. 863–874.

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

[107] Adam J Riesselman, John B Ingraham, and Debora S Marks. "Deep generative models of genetic variation capture the effects of mutations". In: *Nature methods* 15.10 (2018), pp. 816–822.

[108] Samuel R. Bowman et al. *Generating Sentences from a Continuous Space*. 2016. arXiv: 1511.06349 [cs.LG].

[109] David J.C. MacKay. *A Practical Bayesian Framework for Backprop Networks*. 1992.

[110] Magnus Ekeberg et al. "Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models". en. In: *Physical Review E* 87.1 (Jan. 2013). URL: https://link.aps.org/doi/10.1103/PhysRevE.87.012707 (visited on 02/23/2019).

[111] Thomas A Hopf et al. "Sequence co-evolution gives 3D contacts and structures of protein complexes". In: *eLife* 3 (), e03430. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4360534/.

[112] Jung-Eun Shin et al. "Protein design and variant prediction using autoregressive generative models". In: *Nature communications* 12.1 (2021), pp. 1–11.

[113] Predrag Radivojac et al. "Protein flexibility and intrinsic disorder". In: *Protein Science* 13.1 (2004), pp. 71–80.

[114] Agnes Toth-Petroczy et al. "Structured states of disordered proteins from genomic sequences". In: *Cell* 167.1 (2016), pp. 158–170.

[115] Eli N Weinstein and Debora S Marks. "A structured observation distribution for generative biological sequence prediction and forecasting". In: *bioRxiv* (2021), pp. 2020–07.

[116] Michael Heinzinger et al. "Modeling the language of life–deep learning protein sequences". In: *bioRxiv* (2019), p. 614313.

[117] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9 (1997), pp. 1735–1780.

[118] Ali Madani et al. *ProGen: Language Modeling for Protein Generation*. 2020. arXiv: 2004.03497 [q-bio.BM].

[119] 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*. 2020, pp. 1–8.

[120] Ahmed Elnaggar et al. "ProtTrans: towards cracking the language of Life's code through self-supervised deep learning and high performance computing". In: *arXiv preprint arXiv:2007.06225* (2020).

[121] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[122] Swati C Manekar and Shailesh R Sathe. "A benchmark study of k-mer counting methods for high-throughput sequencing". In: *GigaScience* 7.12 (2018), giy125.

[123] David R. So et al. *Primer: Searching for Efficient Transformers for Language Modeling*. 2021. arXiv: 2109.08668 [cs.LG].

[124] Baris E. Suzek et al. "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches". In: *Bioinformatics* 31.6 (Nov. 2014), pp. 926–932. eprint: https://academic.oup.com/bioinformatics/article-pdf/31/6/926/569379/btu739.pdf. URL: https://doi.org/10.1093/bioinformatics/btu739.

[125] Amit Kessel and Nir Ben-Tal. *Introduction to Proteins: Structure, Function, and Motion, SECOND EDITION (Chapman & Hall/CRC Mathematical and Computational Biology)*. Mar. 2018.

[126] Erik Nijkamp et al. “ProGen2: Exploring the Boundaries of Protein Language Models”. In: *ArXiv* abs/2206.13517 (2022).

[127] Daniel Hesslow et al. “RITA: a Study on Scaling Up Generative Protein Sequence Models”. In: *ArXiv* abs/2205.05789 (2022).

[128] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. “ProtGPT2 is a deep unsupervised language model for protein design”. In: *Nature Communications* 13 (2022).

[129] Brian Lester, Rami Al-Rfou, and Noah Constant. *The Power of Scale for Parameter-Efficient Prompt Tuning*. arXiv:2104.08691 [cs]. Sept. 2021. URL: http://arxiv.org/abs/2104.08691 (visited on 01/30/2023).

[130] Roshan M. Rao et al. “MSA Transformer”. en. In: *Proceedings of the 38th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2021, pp. 8844–8856. URL: https://proceedings.mlr.press/v139/rao21a.html (visited on 01/08/2023).

[131] Dan Jurafsky and James H. Martin. “Speech and Language Processing, 2nd Edition”. In: 2008.

[132] Eran Kotler et al. “A Systematic p53 Mutation Library Links Differential Functional Impact to Cancer Mutation Pattern and Evolutionary Conservation”. en. In: *Molecular Cell* 71.1 (July 2018), 178–190.e8. URL: https://linkinghub.elsevier.com/retrieve/pii/S1097276518304544 (visited on 01/23/2022).

[133] Christian Dallago et al. “Learned Embeddings from Deep Learning to Visualize and Predict Protein Sets”. en. In: *Current Protocols* 1.5 (2021). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpz1.113, e113. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpz1.113 (visited on 01/04/2023).

[134] Jeffrey I Boucher, Daniel NA Bolon, and Dan S Tawfik. “Quantifying and understanding the fitness effects of protein mutations: Laboratory versus nature”. In: *Protein Science* 25.7 (2016), pp. 1219–1226.

[135] Jonathan Frazer et al. “Disease variant prediction with deep generative models of evolutionary data”. en. In: *Nature* 599.7883 (Nov. 2021). Number: 7883 Publisher: Nature Publishing Group, pp. 91–95. URL: https://www.nature.com/articles/s41586-021-04043-8 (visited on 02/19/2023).

[136] Nadav Brandes et al. “Genome-wide prediction of disease variants with a deep protein language model”. In: *bioRxiv* (2022).

[137] Pascal Notin et al. “Tranception: Protein Fitness Prediction with Autoregressive Transformers and Inference-time Retrieval”. en. In: *Proceedings of the 39th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, June 2022, pp. 16990–17017. URL: https://proceedings.mlr.press/v162/notin22a.html.

[138] Pascal Notin et al. “TranceptEVE: Combining Family-specific and Family-agnostic Models of Protein Sequences for Improved Fitness Prediction”. en. In: Dec. 2022. URL: https://openreview.net/forum?id=l7Oo9DcLmR1 (visited on 02/19/2023).

[139] Jannik Kossen et al. *Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning*. arXiv:2106.02584 [cs, stat] version: 2. Feb. 2022. URL: http://arxiv.org/abs/2106.02584.

[140] Chloe Hsu et al. "Learning protein fitness models from evolutionary and assay-labeled data". en. In: *Nature Biotechnology* 40.7 (July 2022), pp. 1114–1122. URL: https://www.nature.com/articles/s41587-021-01146-5.

[141] Roshan Rao et al. *Evaluating Protein Transfer Learning with TAPE*. arXiv:1906.08230 [cs, q-bio, stat]. June 2019. URL: http://arxiv.org/abs/1906.08230 (visited on 06/04/2023).

[142] Christian Dallago et al. "FLIP: Benchmark tasks in fitness landscape inference for proteins". In: (2021).

[143] Ahmed Elnaggar et al. *Ankh: Optimized Protein Language Model Unlocks General-Purpose Modelling*. arXiv:2301.06568 [cs, q-bio]. Jan. 2023. URL: http://arxiv.org/abs/2301.06568 (visited on 02/01/2023).

[144] Hannes Stärk et al. "Light attention predicts protein location from the language of life". In: *Bioinformatics Advances* 1.1 (Jan. 2021), vbab035. URL: https://doi.org/10.1093/bioadv/vbab035 (visited on 02/20/2023).

[145] Cristopher V. Van Hout et al. "Exome sequencing and characterization of 49,960 individuals in the UK Biobank". en. In: *Nature* 586.7831 (Oct. 2020). Number: 7831 Publisher: Nature Publishing Group, pp. 749–756. URL: https://www.nature.com/articles/s41586-020-2853-0 (visited on 05/09/2023).

[146] Yanan Cao et al. "The ChinaMAP analytics of deep whole genome sequences in 10,588 individuals". en. In: *Cell Research* 30.9 (Sept. 2020). Number: 9 Publisher: Nature Publishing Group, pp. 717–731. URL: https://www.nature.com/articles/s41422-020-0322-9 (visited on 05/09/2023).

[147] Daniel F. Gudbjartsson et al. "Large-scale whole-genome sequencing of the Icelandic population". en. In: *Nature Genetics* 47.5 (May 2015). Number: 5 Publisher: Nature Publishing Group, pp. 435–444. URL: https://www.nature.com/articles/ng.3247 (visited on 05/09/2023).

[148] Konrad J. Karczewski et al. "The mutational constraint spectrum quantified from variation in 141,456 humans". en. In: *Nature* 581.7809 (May 2020). Number: 7809 Publisher: Nature Publishing Group, pp. 434–443. URL: https://www.nature.com/articles/s41586-020-2308-7 (visited on 05/09/2023).

[149] Heidi L. Rehm et al. "ClinGen–the Clinical Genome Resource". eng. In: *The New England Journal of Medicine* 372.23 (June 2015), pp. 2235–2242.

[150] Daniele Raimondi et al. "DEOGEN2: prediction and interactive visualization of single amino acid variant deleteriousness in human proteins". eng. In: *Nucleic Acids Research* 45.W1 (July 2017), W201–W206.

[151] Bing-Jian Feng. "PERCH: A Unified Framework for Disease Gene Prioritization". eng. In: *Human Mutation* 38.3 (Mar. 2017), pp. 243–251.

[152] Iuliana Ionita-Laza et al. “A spectral approach integrating functional genomic annotations for coding and noncoding variants”. eng. In: *Nature Genetics* 48.2 (Feb. 2016), pp. 214–220.

[153] Karthik A. Jagadeesh et al. “M-CAP eliminates a majority of variants of uncertain significance in clinical exomes at high sensitivity”. eng. In: *Nature Genetics* 48.12 (Dec. 2016), pp. 1581–1586.

[154] Philipp Rentzsch et al. “CADD: predicting the deleteriousness of variants throughout the human genome”. eng. In: *Nucleic Acids Research* 47.D1 (Jan. 2019), pp. D886–D894.

[155] Ivan A. Adzhubei et al. “A method and server for predicting damaging missense mutations”. eng. In: *Nature Methods* 7.4 (Apr. 2010), pp. 248–249.

[156] Gregory M. Findlay et al. “Accurate classification of BRCA1 variants with saturation genome editing”. en. In: *Nature* 562.7726 (Oct. 2018), pp. 217–222. URL: http://www.nature.com/articles/s41586-018-0461-z (visited on 01/23/2022).

[157] Dominik G. Grimm et al. “The evaluation of tools used to predict the impact of missense variants is hindered by two types of circularity”. eng. In: *Human Mutation* 36.5 (May 2015), pp. 513–523.

[158] Sue Richards et al. “Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology”. eng. In: *Genetics in Medicine: Official Journal of the American College of Medical Genetics* 17.5 (May 2015), pp. 405–424.

[159] Debora S. Marks et al. “Protein 3D structure computed from evolutionary sequence variation”. eng. In: *PloS One* 6.12 (2011), e28766.

[160] Thomas A Hopf et al. “Sequence co-evolution gives 3D contacts and structures of protein complexes”. In: *eLife* 3 (Sept. 2014). Ed. by John Kuriyan. Publisher: eLife Sciences Publications, Ltd, e03430. URL: https://doi.org/10.7554/eLife.03430 (visited on 05/09/2023).

[161] Alan Lapedes, Bertrand Giraud, and Christopher Jarzynski. *Using Sequence Alignments to Predict Protein Structure and Stability With High Accuracy.* arXiv:1207.2484 [q-bio]. July 2012. URL: http://arxiv.org/abs/1207.2484 (visited on 05/09/2023).

[162] B. E. Suzek et al. “UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches”. en. In: *Bioinformatics* 31.6 (Mar. 2015), pp. 926–932. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu739 (visited on 10/09/2020).

[163] G. J. McLachlan and K. Basford. “Mixture models : inference and applications to clustering”. In: 1988.

[164] Andrew M. Glazer et al. “Deep Mutational Scan of an *SCN5A* Voltage Sensor”. en. In: *Circulation: Genomic and Precision Medicine* 13.1 (Feb. 2020). URL: https://www.ahajournals.org/doi/10.1161/CIRCGEN.119.002786 (visited on 01/23/2022).

[165] Andrew O. Giacomelli et al. "Mutational processes shape the landscape of TP53 mutations in human cancer". en. In: *Nature Genetics* 50.10 (Oct. 2018), pp. 1381–1387. URL: http://www.nature.com/articles/s41588-018-0204-y (visited on 01/23/2022).

[166] Taylor L. Mighell, Sara Evans-Dutson, and Brian J. O'Roak. "A Saturation Mutagenesis Approach to Understanding PTEN Lipid Phosphatase Activity and Genotype-Phenotype Relationships". en. In: *The American Journal of Human Genetics* 102.5 (May 2018), pp. 943–955. URL: https://linkinghub.elsevier.com/retrieve/pii/S0002929718301071 (visited on 01/23/2022).

[167] Xiaoyan Jia et al. "Massively parallel functional testing of MSH2 missense variants conferring Lynch syndrome risk". en. In: *The American Journal of Human Genetics* 108.1 (Jan. 2021), pp. 163–175. URL: https://linkinghub.elsevier.com/retrieve/pii/S0002929720304390 (visited on 01/23/2022).

[168] Gianfranco Frigo et al. "Homozygous SCN5A mutation in Brugada syndrome with monomorphic ventricular tachycardia and structural heart abnormalities". In: *EP Europace* 9.6 (June 2007), pp. 391–397. URL: https://doi.org/10.1093/europace/eum053 (visited on 05/10/2023).

[169] Hideki Itoh et al. "Asymmetry of parental origin in long QT syndrome: preferential maternal transmission of KCNQ1 variants linked to channel dysfunction". en. In: *European Journal of Human Genetics* 24.8 (Aug. 2016). Number: 8 Publisher: Nature Publishing Group, pp. 1160–1166. URL: https://www.nature.com/articles/ejhg2015257 (visited on 05/10/2023).

[170] Xiaojing Pan et al. "Structure of the human voltage-gated sodium channel Nav1.4 in complex with β1". eng. In: *Science (New York, N.Y.)* 362.6412 (Oct. 2018), eaau2486.

[171] Joshua J. Warren et al. "Structure of the human MutSalpha DNA lesion recognition complex". eng. In: *Molecular Cell* 26.4 (May 2007), pp. 579–592.

[172] Sarah E. Brnich et al. "Recommendations for application of the functional evidence PS3/BS3 criterion using the ACMG/AMP sequence variant interpretation framework". In: *Genome Medicine* 12.1 (Dec. 2019), p. 3. URL: https://doi.org/10.1186/s13073-019-0690-2 (visited on 05/10/2023).

[173] David C. Montefiori. "Measuring HIV neutralization in a luciferase reporter gene assay". eng. In: *Methods in Molecular Biology (Clifton, N.J.)* 485 (2009), pp. 395–405.

[174] Fabian Schmidt et al. "Measuring SARS-CoV-2 neutralizing antibody activity using pseudotyped and chimeric viruses". eng. In: *The Journal of Experimental Medicine* 217.11 (Nov. 2020), e20201181.

[175] Jinhui Dong et al. "Genetic and structural basis for SARS-CoV-2 variant neutralization by a two-antibody cocktail." en. In: *Nat Microbiol* (2021), pp. 1233–1244. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8543371 (visited on 05/11/2023).

[176] Tyler N. Starr et al. “Prospective mapping of viral mutations that escape antibodies used to treat COVID-19”. eng. In: *Science (New York, N.Y.)* 371.6531 (Feb. 2021), pp. 850–854.

[177] Tyler N. Starr et al. “SARS-CoV-2 RBD antibodies that maximize breadth and resistance to escape”. en. In: *Nature* 597.7874 (Sept. 2021). Number: 7874 Publisher: Nature Publishing Group, pp. 97–102. URL: https://www.nature.com/articles/s41586-021-03807-6 (visited on 05/11/2023).

[178] M. Alejandra Tortorici et al. “Broad sarbecovirus neutralization by a human monoclonal antibody”. en. In: *Nature* 597.7874 (Sept. 2021). Number: 7874 Publisher: Nature Publishing Group, pp. 103–108. URL: https://www.nature.com/articles/s41586-021-03817-4 (visited on 05/11/2023).

[179] Yunlong Cao et al. “BA.2.12.1, BA.4 and BA.5 escape antibodies elicited by Omicron infection”. en. In: *Nature* 608.7923 (Aug. 2022). Number: 7923 Publisher: Nature Publishing Group, pp. 593–602. URL: https://www.nature.com/articles/s41586-022-04980-y (visited on 05/11/2023).

[180] Yunlong Cao et al. “Omicron escapes the majority of existing SARS-CoV-2 neutralizing antibodies”. en. In: *Nature* 602.7898 (Feb. 2022). Number: 7898 Publisher: Nature Publishing Group, pp. 657–663. URL: https://www.nature.com/articles/s41586-021-04385-3 (visited on 05/11/2023).

[181] Michael B. Doud, Juhye M. Lee, and Jesse D. Bloom. “How single mutations affect viral escape from broad and narrow antibodies to H1 influenza hemagglutinin”. en. In: *Nature Communications* 9.1 (Apr. 2018). Number: 1 Publisher: Nature Publishing Group, p. 1386. URL: https://www.nature.com/articles/s41467-018-03665-3 (visited on 05/11/2023).

[182] Adam S. Dingens et al. “An Antigenic Atlas of HIV-1 Escape from Broadly Neutralizing Antibodies Distinguishes Functional and Structural Epitopes”. eng. In: *Immunity* 50.2 (Feb. 2019), 520–532.e3.

[183] Allison J. Greaney et al. “Antibodies elicited by mRNA-1273 vaccination bind more broadly to the receptor binding domain than do those from SARS-CoV-2 infection”. eng. In: *Science Translational Medicine* 13.600 (June 2021), eabi9915.

[184] Allison J. Greaney et al. “Complete Mapping of Mutations to the SARS-CoV-2 Spike Receptor-Binding Domain that Escape Antibody Recognition”. eng. In: *Cell Host & Microbe* 29.1 (Jan. 2021), 44–57.e9.

[185] Allison J. Greaney et al. “Comprehensive mapping of mutations in the SARS-CoV-2 receptor-binding domain that affect recognition by polyclonal human plasma antibodies”. eng. In: *Cell Host & Microbe* 29.3 (Mar. 2021), 463–476.e6.

[186] Allison J. Greaney et al. "Mapping mutations to the SARS-CoV-2 RBD that escape binding by different classes of antibodies". en. In: *Nature Communications* 12.1 (July 2021). Number: 1 Publisher: Nature Publishing Group, p. 4196. URL: https://www.nature.com/articles/s41467-021-24435-8 (visited on 05/11/2023).

[187] Allison J. Greaney et al. "A SARS-CoV-2 variant elicits an antibody response with a shifted immunodominance hierarchy". eng. In: *bioRxiv: The Preprint Server for Biology* (Oct. 2021), p. 2021.10.12.464114.

[188] Allison J. Greaney et al. "The SARS-CoV-2 Delta variant induces an antibody response largely focused on class 1 and 2 antibody epitopes". en. In: *PLOS Pathogens* 18.6 (June 2022). Publisher: Public Library of Science, e1010592. URL: https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1010592 (visited on 05/11/2023).

[189] Tyler N. Starr et al. "Complete map of SARS-CoV-2 RBD mutations that escape the monoclonal antibody LY-CoV555 and its cocktail with LY-CoV016". eng. In: *bioRxiv: The Preprint Server for Biology* (Feb. 2021), p. 2021.02.17.431683.

[190] Tyler N. Starr et al. "Deep Mutational Scanning of SARS-CoV-2 Receptor Binding Domain Reveals Constraints on Folding and ACE2 Binding". en. In: *Cell* 182.5 (Sept. 2020), 1295–1310.e20. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867420310035 (visited on 01/23/2022).

[191] Fabrizio Pucci and Marianne Rooman. "Prediction and Evolution of the Molecular Fitness of SARS-CoV-2 Variants: Introducing SpikePro". eng. In: *Viruses* 13.5 (May 2021), p. 935.

[192] Nash D. Rochman et al. "Epistasis at the SARS-CoV-2 Receptor-Binding Domain Interface and the Propitiously Boring Implications for Vaccine Escape". eng. In: *mBio* 13.2 (Apr. 2022), e0013522.

[193] Karim Beguir et al. "Early computational detection of potential high-risk SARS-CoV-2 variants". eng. In: *Computers in Biology and Medicine* 155 (Mar. 2023), p. 106618.

[194] Joseph M. Taft et al. "Deep mutational learning predicts ACE2 binding and antibody escape to combinatorial mutations in the SARS-CoV-2 receptor-binding domain". en. In: *Cell* 185.21 (Oct. 2022), 4008–4022.e14. URL: https://www.sciencedirect.com/science/article/pii/S0092867422011199 (visited on 05/11/2023).

[195] Lizhi Ian Gong, Marc A Suchard, and Jesse D Bloom. "Stability-mediated epistasis constrains the evolution of an influenza protein". In: *eLife* 2 (May 2013). Ed. by Mercedes Pascual. Publisher: eLife Sciences Publications, Ltd, e00631. URL: https://doi.org/10.7554/eLife.00631 (visited on 05/11/2023).

[196] Tyler N. Starr et al. "Shifting mutational constraints in the SARS-CoV-2 receptor-binding domain during viral evolution". eng. In: *Science (New York, N.Y.)* 377.6604 (July 2022), pp. 420–424.

[197] Hugh K Haddox et al. "Mapping mutational effects along the evolutionary landscape of HIV envelope". en. In: *eLife* 7 (Mar. 2018), e34420. URL: https://elifesciences.org/articles/34420 (visited on 01/23/2022).

[198] Michael Doud and Jesse Bloom. "Accurate Measurement of the Effects of All Amino-Acid Mutations on Influenza Hemagglutinin". en. In: *Viruses* 8.6 (June 2016), p. 155. URL: http://www.mdpi.com/1999-4915/8/6/155 (visited on 01/23/2022).

[199] Nicholas C. Wu et al. "Different genetic barriers for resistance to HA stem antibodies in influenza H3 and H1 viruses". eng. In: *Science (New York, N.Y.)* 368.6497 (June 2020), pp. 1335–1340.

[200] Jeremy I. Roop et al. "Identification of HIV-1 Envelope Mutations that Enhance Entry Using Macaque CD4 and CCR5". en. In: *Viruses* 12.2 (Feb. 2020). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 241. URL: https://www.mdpi.com/1999-4915/12/2/241 (visited on 05/11/2023).

[201] Maria Duenas-Decamp et al. "Saturation mutagenesis of the HIV-1 envelope CD4 binding loop reveals residues controlling distinct trimer conformations". In: *PLoS pathogens* 12.11 (2016), e1005988.

[202] J. M. Thornton et al. "Location of 'continuous' antigenic determinants in the protruding regions of proteins". eng. In: *The EMBO journal* 5.2 (Feb. 1986), pp. 409–413.

[203] J Novotný et al. "Antigenic determinants in proteins coincide with surface regions accessible to large probes (antibody domains)." In: *Proceedings of the National Academy of Sciences of the United States of America* 83.2 (Jan. 1986), pp. 226–230. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC322830/ (visited on 05/11/2023).

[204] Pernille Haste Andersen, Morten Nielsen, and Ole Lund. "Prediction of residues in discontinuous B-cell epitopes using protein 3D structures". eng. In: *Protein Science: A Publication of the Protein Society* 15.11 (Nov. 2006), pp. 2558–2567.

[205] Chih-Peng Lin et al. "Deriving protein dynamical properties from weighted protein contact number". eng. In: *Proteins* 72.3 (Aug. 2008), pp. 929–935.

[206] Cyrus Chothia and Joël Janin. "Principles of protein–protein recognition". en. In: *Nature* 256.5520 (Aug. 1975). Number: 5520 Publisher: Nature Publishing Group, pp. 705–708. URL: https://www.nature.com/articles/256705a0 (visited on 05/11/2023).

[207] Jens Vindahl Kringelum et al. "Structural analysis of B-cell epitopes in antibody:protein complexes". eng. In: *Molecular Immunology* 53.1-2 (Jan. 2013), pp. 24–34.

[208] D. Eisenberg, R. M. Weiss, and T. C. Terwilliger. "The hydrophobic moment detects periodicity in protein hydrophobicity". eng. In: *Proceedings of the National Academy of Sciences of the United States of America* 81.1 (Jan. 1984), pp. 140–144.

[209] S Henikoff and J G Henikoff. "Amino acid substitution matrices from protein blocks." In: *Proceedings of the National Academy of Sciences of the United States of America* 89.22 (Nov. 1992), pp. 10915–10919. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC50453/ (visited on 05/11/2023).

[210] Shruti Khare et al. “GISAID’s Role in Pandemic Response”. In: *China CDC Weekly* 3.49 (Dec. 2021), pp. 1049–1051. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8668406/ (visited on 05/11/2023).

[211] Philip L. Tzou et al. “Coronavirus Resistance Database (CoV-RDB): SARS-CoV-2 susceptibility to monoclonal antibodies, convalescent plasma, and plasma from vaccinated persons”. eng. In: *PloS One* 17.3 (2022), e0261045.

[212] Gabriele Cerutti et al. “Potent SARS-CoV-2 neutralizing antibodies directed against spike N-terminal domain target a single supersite”. eng. In: *Cell Host & Microbe* 29.5 (May 2021), 819–833.e7.

[213] Luca Piccoli et al. “Mapping Neutralizing and Immunodominant Sites on the SARS-CoV-2 Spike Receptor-Binding Domain by Structure-Guided High-Resolution Serology”. eng. In: *Cell* 183.4 (Nov. 2020), 1024–1042.e21.

[214] Fatima Amanat et al. “SARS-CoV-2 mRNA vaccination induces functionally diverse antibodies to NTD, RBD, and S2”. eng. In: *Cell* 184.15 (July 2021), 3936–3948.e10.

[215] Ning Wang et al. “Subunit Vaccines Against Emerging Pathogenic Human Coronaviruses”. In: *Frontiers in Microbiology* 11 (2020). URL: https://www.frontiersin.org/articles/10.3389/fmicb.2020.00298 (visited on 05/11/2023).

[216] Sandhya Bangaru et al. “Structural analysis of full-length SARS-CoV-2 spike protein from an advanced vaccine candidate”. eng. In: *Science (New York, N.Y.)* 370.6520 (Nov. 2020), pp. 1089–1094.

[217] Tiziana Ginex et al. “The structural role of SARS-CoV-2 genetic background in the emergence and success of spike mutations: The case of the spike A222V mutation”. en. In: *PLOS Pathogens* 18.7 (July 2022). Publisher: Public Library of Science, e1010631. URL: https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1010631 (visited on 05/11/2023).

[218] Lue Ping Zhao et al. “Rapidly Identifying New Coronavirus Mutations of Potential Concern in the Omicron Variant Using an Unsupervised Learning Strategy”. eng. In: *Research Square* (Feb. 2022), rs.3.rs–1280819.

[219] Kathryn E. Kistler, John Huddleston, and Trevor Bedford. “Rapid and parallel adaptive mutations in spike S1 drive clade success in SARS-CoV-2”. In: *Cell Host & Microbe* 30.4 (Apr. 2022), 545–555.e4. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8938189/ (visited on 05/11/2023).

[220] Takuya Tada et al. “Increased resistance of SARS-CoV-2 Omicron variant to neutralization by vaccine-elicited and therapeutic antibodies”. eng. In: *EBioMedicine* 78 (Apr. 2022), p. 103944.

[221] Brian Hie et al. “Learning the language of viral evolution and escape”. eng. In: *Science (New York, N.Y.)* 371.6526 (Jan. 2021), pp. 284–288.

[222] R Shyama Prasad Rao et al. “Evolutionary Dynamics of Indels in SARS-CoV-2 Spike Glycoprotein”. In: *Evolutionary Bioinformatics Online* 17 (Dec. 2021), p. 11769343211064616. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8655444/ (visited on 05/11/2023).

[223] Willie Neiswanger, Ke Alexander Wang, and Stefano Ermon. “Bayesian algorithm execution: Estimating computable properties of black-box functions using mutual information”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8005–8015.

[224] Ronen Ben Jehuda, Yuval Shemer, and Ofer Binah. “Genome editing in induced pluripotent stem cells using CRISPR/Cas9”. In: *Stem Cell Reviews and Reports* 14.3 (2018), pp. 323–336.

[225] Michael Dickson and Jean Paul Gagnon. “Key factors in the rising cost of new drug discovery and development”. In: *Nature Reviews Drug Discovery* 3.5 (2004), pp. 417–429.

[226] Michael Dickson and Jean Paul Gagnon. “The cost of new drug discovery and development”. In: *Discovery Medicine* 4.22 (2009), pp. 172–179.

[227] Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. “Innovation in the pharmaceutical industry: new estimates of R&D costs”. In: *Journal of Health Economics* 47 (2016), pp. 20–33.

[228] Nurken Berdigaliyev and Mohamad Aljofan. “An overview of drug discovery and development”. In: *Future Medicinal Chemistry* 12.10 (2020), pp. 939–947.

[229] Cole Trapnell. “Defining cell types and states with single-cell genomics”. In: *Genome Research* 25.10 (2015), pp. 1491–1498.

[230] Yehudit Hasin, Marcus Seldin, and Aldons Lusis. “Multi-omics approaches to disease”. In: *Genome Biology* 18.1 (2017), pp. 1–15.

[231] Thomas Worzfeld et al. “The unique molecular and cellular microenvironment of ovarian cancer”. In: *Frontiers in Oncology* 7 (2017), p. 24.

[232] Lia Chappell, Andrew JC Russell, and Thierry Voet. “Single-cell (multi) omics technologies”. In: *Annual Review of Genomics and Human Genetics* 19 (2018), pp. 15–41.

[233] Adam L. MacLean, Tian Hong, and Qing Nie. “Exploring intermediate cell states through the lens of single cells”. In: *Current Opinion in Systems Biology* 9 (2018). Mathematic Modelling, pp. 32–41. URL: https://www.sciencedirect.com/science/article/pii/S2452310017302238.

[234] Andrei Cristian Nica et al. “Evaluating Generalization in GFlowNets for Molecule Design”. In: *ICLR2022 Machine Learning for Drug Discovery*. 2022.

[235] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical bayesian optimization of machine learning algorithms”. In: *Advances in neural information processing systems* 25 (2012).

[236] Bobak Shahriari et al. “Taking the human out of the loop: A review of Bayesian optimization”. In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[237] Arash Mehrjou et al. “GeneDisco: A Benchmark for Experimental Design in Drug Discovery”. In: *arXiv preprint arXiv:2110.11875* (2021).

[238] Ashish Goel, Sudipto Guha, and Kamesh Munagala. "How to Probe for an Extreme Value". In: *ACM Trans. Algorithms* 7.1 (Dec. 2010). URL: https://doi.org/10.1145/1868237.1868250.

[239] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[240] Yunkai Zhu et al. "A genome-wide CRISPR screen identifies host factors that regulate SARS-CoV-2 entry". In: *Nature communications* 12.1 (2021), pp. 1–11.

[241] Ralf Schmidt et al. "CRISPR activation and interference screens in primary human T cells decode cytokine regulation". In: *bioRxiv* (2021). eprint: https://www.biorxiv.org/content/early/2021/05/12/2021.05.11.443701.full.pdf. URL: https://www.biorxiv.org/content/early/2021/05/12/2021.05.11.443701.

[242] Xiaoxuan Zhuang, Daniel P. Veltri, and Eric O. Long. "Genome-Wide CRISPR Screen Reveals Cancer Cell Resistance to NK Cells Induced by NK-Derived IFN-γ". In: *Frontiers in Immunology* 10 (2019), p. 2879. URL: https://www.frontiersin.org/article/10.3389/fimmu.2019.02879.

[243] Hanjun Dai et al. "Syntax-Directed Variational Autoencoder for Structured Data". In: *ICLR*. 2018.

[244] John Bradshaw et al. "A Model to Search for Synthesizable Molecules". In: *NeurIPS*. 2019.

[245] David Janz, Jos van der Westhuizen, and José Miguel Hernández-Lobato. *Actively Learning what makes a Discrete Sequence Valid*. 2017. arXiv: 1708.04465 [stat.ML].

[246] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. "Constrained Bayesian optimization for automatic chemical design using variational autoencoders". In: *Chemical Science* (2020).

[247] Xianggen Liu et al. "A Chance-Constrained Generative Framework for Sequence Optimization". In: *ICML*. 2020.

[248] David Duvenaud et al. "Convolutional Networks on Graphs for Learning Molecular Fingerprints". In: *Advances in Neural Information Processing Systems*. 2015.

[249] Yujia Li et al. "Learning Deep Generative Models of Graphs". In: *ICLR*. 2018.

[250] Wengong Jin, Regina Barzilay, and T. Jaakkola. "Junction Tree Variational Autoencoder for Molecular Graph Generation". In: *ICML*. 2018.

[251] Zhenpeng Zhou et al. "Optimization of Molecules via Deep Reinforcement Learning". In: *Scientific Reports* 9.1 (July 2019). URL: http://dx.doi.org/10.1038/s41598-019-47148-x.

[252] Chence Shi et al. "GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation". In: *ICLR*. 2020.

[253] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. "Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining". In: *Advances in Neural Information Processing Systems*. 2020.

[254] Andrew Mchutchon and Carl Rasmussen. "Gaussian Process Training with Input Noise". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/a8e864d04c95572d1aece099af852d0a-Paper.pdf.

[255] J. J. Irwin et al. "ZINC: A Free Tool to Discover Chemistry for Biology". In: *Journal of Chemical Information and Modeling* 52 (2012), pp. 1757–1768.

[256] Nathan Brown et al. "GuacaMol: Benchmarking Models for de Novo Molecular Design". In: *Journal of Chemical Information and Modeling* 59.3 (Mar. 2019), pp. 1096–1108. URL: http://dx.doi.org/10.1021/acs.jcim.8b00839.

[257] Ivan V. Korendovych. "Rational and Semirational Protein Design". In: *Methods in molecular biology (Clifton, N.J.)* 1685 (2018), pp. 15–23. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5912912/ (visited on 07/07/2023).

[258] Michael H. Hecht et al. "De novo proteins from designed combinatorial libraries". In: *Protein Science : A Publication of the Protein Society* 13.7 (July 2004), pp. 1711–1723. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2279937/ (visited on 07/07/2023).

[259] Julia Koehler Leman et al. "Macromolecular modeling and design in Rosetta: recent methods and frameworks". en. In: *Nature Methods* 17.7 (July 2020). Number: 7 Publisher: Nature Publishing Group, pp. 665–680. URL: https://www.nature.com/articles/s41592-020-0848-2 (visited on 07/07/2023).

[260] Hongyuan Lu et al. "Machine learning-aided engineering of hydrolases for PET depolymerization". en. In: *Nature* 604.7907 (Apr. 2022). Number: 7907 Publisher: Nature Publishing Group, pp. 662–667. URL: https://www.nature.com/articles/s41586-022-04599-z (visited on 02/28/2023).

[261] Benjamin Schubert et al. "Population-specific design of de-immunized protein biotherapeutics". en. In: *PLOS Computational Biology* 14.3 (Mar. 2018). Publisher: Public Library of Science, e1005983. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005983 (visited on 04/04/2023).

[262] Andrew Giessel et al. "Therapeutic enzyme engineering using a generative neural network". en. In: *Scientific Reports* 12.1 (Jan. 2022). Number: 1 Publisher: Nature Publishing Group, p. 1536. URL: https://www.nature.com/articles/s41598-022-05195-x (visited on 04/04/2023).

[263] Benjamin Fram et al. "Simultaneous enhancement of multiple functional properties using evolution-informed protein design". eng. In: *bioRxiv: The Preprint Server for Biology* (May 2023), p. 2023.05.09.539914.

[264] William P. Russ et al. "An evolution-based model for designing chorismate mutase enzymes". In: *Science* 369.6502 (July 2020). Publisher: American Association for the Advancement of Science, pp. 440–445. URL: https://www.science.org/doi/10.1126/science.aba3304 (visited on 07/08/2023).

[265] Ge Liu et al. "Antibody complementarity determining region design using high-capacity machine learning". In: *Bioinformatics* 36.7 (Apr. 2020), pp. 2126–2133. URL: https://doi.org/10.1093/bioinformatics/btz895 (visited on 01/06/2023).

[266] Yougen Li et al. "A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments". en. In: *Nature Biotechnology* 25.9 (Sept. 2007). Number: 9 Publisher: Nature Publishing Group, pp. 1051–1056. URL: https://www.nature.com/articles/nbt1333 (visited on 07/08/2023).

[267] Regina S. Salvat et al. "Computationally optimized deimmunization libraries yield highly mutated enzymes with low immunogenicity and enhanced activity". In: *Proceedings of the National Academy of Sciences* 114.26 (June 2017). Publisher: Proceedings of the National Academy of Sciences, E5085–E5093. URL: https://www.pnas.org/doi/full/10.1073/pnas.1621233114 (visited on 04/04/2023).

[268] Bruce J. Wittmann, Yisong Yue, and Frances H. Arnold. *Machine Learning-Assisted Directed Evolution Navigates a Combinatorial Epistatic Fitness Landscape with Minimal Screening Burden*. en. Pages: 2020.12.04.408955 Section: New Results. Dec. 2020. URL: https://www.biorxiv.org/content/10.1101/2020.12.04.408955v1 (visited on 01/06/2023).

[269] Richard J. Fox et al. "Improving catalytic function by ProSAR-driven enzyme evolution". en. In: *Nature Biotechnology* 25.3 (Mar. 2007). Number: 3 Publisher: Nature Publishing Group, pp. 338–344. URL: https://www.nature.com/articles/nbt1286 (visited on 07/08/2023).

[270] Ivan Anishchenko et al. "De novo protein design by deep network hallucination". en. In: *Nature* 600.7889 (Dec. 2021). Number: 7889 Publisher: Nature Publishing Group, pp. 547–552. URL: https://www.nature.com/articles/s41586-021-04184-w (visited on 01/05/2023).

[271] J. Dauparas et al. "Robust deep learning–based protein sequence design using ProteinMPNN". In: *Science* 378.6615 (Oct. 2022). Publisher: American Association for the Advancement of Science, pp. 49–56. URL: https://www.science.org/doi/10.1126/science.add2187 (visited on 02/20/2023).

[272] Jue Wang et al. "Scaffolding protein functional sites using deep learning". In: *Science* 377.6604 (July 2022). Publisher: American Association for the Advancement of Science, pp. 387–394. URL: https://www.science.org/doi/full/10.1126/science.abn2100 (visited on 02/21/2023).

[273] Jiayi Dou et al. "De novo design of a fluorescence-activating β-barrel". en. In: *Nature* 561.7724 (Sept. 2018). Number: 7724 Publisher: Nature Publishing Group, pp. 485–491. URL: https://www.nature.com/articles/s41586-018-0509-0 (visited on 07/08/2023).

[274] Andy Hsien-Wei Yeh et al. "De novo design of luciferases using deep learning". en. In: *Nature* 614.7949 (Feb. 2023). Number: 7949 Publisher: Nature Publishing Group, pp. 774–780. URL: https://www.nature.com/articles/s41586-023-05696-3 (visited on 07/08/2023).

[275] Alex Hawkins-Hooker et al. "Generating functional protein variants with variational autoencoders". en. In: *PLOS Computational Biology* 17.2 (Feb. 2021). Publisher: Public Library of Science, e1008736. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008736 (visited on 02/20/2023).

[276] Lukas Theo Schmitt et al. "Prediction of designer-recombinases for DNA editing with generative deep learning". en. In: *Nature Communications* 13.1 (Dec. 2022). Number: 1 Publisher: Nature Publishing Group, p. 7966. URL: https://www.nature.com/articles/s41467-022-35614-6 (visited on 04/04/2023).

[277] Surojit Biswas et al. "Low-N Protein Engineering with Data-Efficient Deep Learning". In: *Nature Methods* 18.4 (Apr. 2021), pp. 389–396.

[278] Michael Heinzinger et al. "Modeling aspects of the language of life through transfer-learning protein sequences". In: *BMC Bioinformatics* 20.1 (Dec. 2019), p. 723. URL: https://doi.org/10.1186/s12859-019-3220-8 (visited on 02/20/2023).

[279] Philip A. Romero, Andreas Krause, and Frances H. Arnold. "Navigating the protein fitness landscape with Gaussian processes". In: *Proceedings of the National Academy of Sciences* 110.3 (Jan. 2013). Publisher: Proceedings of the National Academy of Sciences, E193–E201. URL: https://www.pnas.org/doi/10.1073/pnas.1215251110 (visited on 01/04/2023).

[280] John Ingraham et al. "Generative Models for Graph-Based Protein Design". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://papers.nips.cc/paper_files/paper/2019/hash/f3a4ff4839c56a5f460c88cce3666a2b-Abstract.html (visited on 07/08/2023).

[281] John Ingraham et al. *Illuminating protein space with a programmable generative model*. en. Pages: 2022.12.01.518682 Section: New Results. Dec. 2022. URL: https://www.biorxiv.org/content/10.1101/2022.12.01.518682v1 (visited on 02/20/2023).

[282] Raphael R. Eguchi, Christian A. Choe, and Po-Ssu Huang. "Ig-VAE: Generative modeling of protein structure by direct 3D coordinate generation". en. In: *PLOS Computational Biology* 18.6 (June 2022). Publisher: Public Library of Science, e1010271. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010271 (visited on 07/08/2023).

[283] Geraldene Munsamy et al. "ZymCTRL: a conditional language model for the controllable generation of artificial enzymes". en. In: (2022).

[284] Anh Nguyen, Nikos Karampatziakis, and Weizhu Chen. *Meet in the Middle: A New Pre-training Paradigm*. arXiv:2303.07295 [cs]. Mar. 2023. URL: http://arxiv.org/abs/2303.07295 (visited on 07/10/2023).

[285] Elodie Laine, Yasaman Karami, and Alessandra Carbone. "GEMME: A Simple and Fast Global Epistatic Model Predicting Mutational Effects". In: *Molecular Biology and Evolution* 36.11 (Nov. 2019), pp. 2604–2619. URL: https://doi.org/10.1093/molbev/msz179.

[286] Zeming Lin et al. *Language models of protein sequences at the scale of evolution enable accurate structure prediction*. en. Pages: 2022.07.20.500902 Section: New Results. July 2022. URL: https://www.biorxiv.org/content/10.1101/2022.07.20.500902v1 (visited on 03/02/2023).

[287] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. arXiv:2203.15556 [cs]. Mar. 2022. URL: http://arxiv.org/abs/2203.15556 (visited on 06/18/2023).

[288] Ningyu Zhang et al. *OntoProtein: Protein Pretraining With Gene Ontology Embedding*. arXiv:2201.11147 [cs, q-bio]. June 2022. URL: http://arxiv.org/abs/2201.11147 (visited on 06/23/2023).

[289] Minghao Xu et al. *ProtST: Multi-Modality Learning of Protein Sequences and Biomedical Texts*. arXiv:2301.12040 [cs, q-bio]. July 2023. URL: http://arxiv.org/abs/2301.12040 (visited on 07/11/2023).

[290] Tianhao Yu et al. "In vitro continuous protein evolution empowered by machine learning and automation". eng. In: *Cell Systems* (May 2023), S2405–4712(23)00115–1.

[291] Jacob T. Rapp, Bennett J. Bremer, and Philip A. Romero. *Self-driving laboratories to autonomously navigate the protein fitness landscape*. en. Pages: 2023.05.20.541582 Section: New Results. May 2023. URL: https://www.biorxiv.org/content/10.1101/2023.05.20.541582v1 (visited on 07/11/2023).

[292] Andres M. Bran et al. *ChemCrow: Augmenting large-language models with chemistry tools*. arXiv:2304.05376 [physics, stat]. June 2023. URL: http://arxiv.org/abs/2304.05376 (visited on 07/11/2023).

[293] H. Jacquier et al. "Capturing the mutational landscape of the beta-lactamase TEM-1". en. In: *Proceedings of the National Academy of Sciences* 110.32 (Aug. 2013), pp. 13067–13072. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1215206110 (visited on 01/23/2022).

[294] Jochen Weile et al. "A framework for exhaustively mapping functional missense variants". en. In: *Molecular Systems Biology* 13.12 (Dec. 2017), p. 957. URL: https://onlinelibrary.wiley.com/doi/10.15252/msb.20177908 (visited on 01/23/2022).

[295] Arti Tripathi et al. “Molecular Determinants of Mutant Phenotypes, Inferred from Saturation Mutagenesis Data”. en. In: *Molecular Biology and Evolution* 33.11 (Nov. 2016), pp. 2960–2975. URL: https://academic.oup.com/mbe/article-lookup/doi/10.1093/molbev/msw182 (visited on 01/23/2022).

[296] Richard N. McLaughlin Jr et al. “The spatial architecture of protein function and adaptation”. en. In: *Nature* 491.7422 (Nov. 2012), pp. 138–142. URL: http://www.nature.com/articles/nature11500 (visited on 01/23/2022).

[297] Nicholas C. Wu et al. “Functional Constraint Profiling of a Viral Protein Reveals Discordance of Evolutionary Conservation and Functionality”. en. In: *PLOS Genetics* 11.7 (July 2015). Ed. by Michael Worobey, e1005310. URL: https://dx.plos.org/10.1371/journal.pgen.1005310 (visited on 01/23/2022).

[298] Benjamin P. Roscoe et al. “Analyses of the Effects of All Ubiquitin Point Mutants on Yeast Growth Rate”. en. In: *Journal of Molecular Biology* 425.8 (Apr. 2013), pp. 1363–1377. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022283613000636 (visited on 01/23/2022).

[299] C. Anders Olson, Nicholas C. Wu, and Ren Sun. “A Comprehensive Biophysical Description of Pairwise Epistasis throughout an Entire Protein Domain”. en. In: *Current Biology* 24.22 (Nov. 2014), pp. 2643–2651. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960982214012688 (visited on 01/23/2022).

[300] Jessica L. Bridgford et al. “Novel drivers and modifiers of MPL-dependent oncogenic transformation identified by deep mutational scanning”. en. In: *Blood* 135.4 (Jan. 2020), pp. 287–292. URL: https://ashpublications.org/blood/article/135/4/287/381157/Novel-drivers-and-modifiers-of-MPLdependent (visited on 01/23/2022).

[301] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *ICLR*. 2019.

[302] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *ArXiv* abs/2001.08361 (2020).

[303] Pascal Notin et al. “Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval”. In: *ICML*. 2022.

[304] Bharat V. Adkar et al. “Protein Model Discrimination Using Mutational Sensitivity Derived from Deep Sequencing”. en. In: *Structure* 20.2 (Feb. 2012), pp. 371–381. URL: https://linkinghub.elsevier.com/retrieve/pii/S0969212612000068 (visited on 01/23/2022).

[305] Robert W. Newberry et al. “Robust Sequence Determinants of α-Synuclein Toxicity in Yeast Implicate Membrane Binding”. en. In: *ACS Chemical Biology* 15.8 (Aug. 2020), pp. 2137–2153. URL: https://pubs.acs.org/doi/10.1021/acschembio.0c00339 (visited on 01/23/2022).

[306] Jason D. Fernandes et al. "Functional Segregation of Overlapping Genes in HIV". en. In: *Cell* 167.7 (Dec. 2016), 1762–1773.e12. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867416316038 (visited on 01/23/2022).

[307] Benjamin P. Roscoe and Daniel N.A. Bolon. "Systematic Exploration of Ubiquitin Sequence, E1 Activation Efficiency, and Experimental Fitness in Yeast". en. In: *Journal of Molecular Biology* 426.15 (July 2014), pp. 2854–2870. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022283614002587 (visited on 01/23/2022).

[308] Liat Rockah-Shmuel, Agnes Tóth-Petróczy, and Dan S. Tawfik. "Systematic Mapping of Protein Mutational Space by Prolonged Drift Reveals the Deleterious Effects of Seemingly Neutral Mutations". en. In: *PLOS Computational Biology* 11.8 (Aug. 2015). Ed. by Christine A. Orengo, e1004421. URL: https://dx.plos.org/10.1371/journal.pcbi.1004421 (visited on 01/23/2022).

[309] Daniel Melamed et al. "Deep mutational scanning of an RRM domain of the *Saccharomyces cerevisiae* poly(A)-binding protein". en. In: *RNA* 19.11 (Nov. 2013), pp. 1537–1551. URL: http://rnajournal.cshlp.org/lookup/doi/10.1261/rna.040709.113 (visited on 01/23/2022).

[310] Alexandre Melnikov et al. "Comprehensive mutational scanning of a kinase *in vivo* reveals substrate-dependent fitness landscapes". en. In: *Nucleic Acids Research* 42.14 (Aug. 2014), e112–e112. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gku511 (visited on 01/23/2022).

[311] Lisa Brenan et al. "Phenotypic Characterization of a Comprehensive Set of MAPK1 /ERK2 Missense Mutants". en. In: *Cell Reports* 17.4 (Oct. 2016), pp. 1171–1183. URL: https://linkinghub.elsevier.com/retrieve/pii/S2211124716313171 (visited on 01/23/2022).

[312] Eric M Jones et al. "Structural and functional characterization of G protein–coupled receptors with deep mutational scanning". en. In: *eLife* 9 (Oct. 2020), e54895. URL: https://elifesciences.org/articles/54895 (visited on 01/23/2022).

[313] Chase C. Suiter et al. "Massively parallel variant characterization identifies *NUDT15* alleles associated with thiopurine toxicity". en. In: *Proceedings of the National Academy of Sciences* 117.10 (Mar. 2020), pp. 5394–5401. URL: http://www.pnas.org/lookup/doi/10.1073/pnas.1915680117 (visited on 01/23/2022).

[314] John Z Chen, Douglas M Fowler, and Nobuhiko Tokuriki. "Comprehensive exploration of the translocation, stability and substrate recognition requirements in VIM-2 lactamase". en. In: *eLife* 9 (June 2020), e56707. URL: https://elifesciences.org/articles/56707 (visited on 01/23/2022).

[315] Victoria O. Pokusaeva et al. "An experimental assay of the interactions of amino acids from orthologous sequences shaping a complex fitness landscape". en. In: *PLOS Genetics* 15.4 (Apr. 2019). Ed. by Dmitri A. Petrov, e1008079. URL: https://dx.plos.org/10.1371/journal.pgen.1008079 (visited on 01/23/2022).

[316] Christopher D. Aakre et al. "Evolving New Protein-Protein Interaction Specificity through Promiscuous Intermediates". en. In: *Cell* 163.3 (Oct. 2015), pp. 594–606. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867415012726 (visited on 01/23/2022).

[317] Hugh K. Haddox, Adam S. Dingens, and Jesse D. Bloom. "Experimental Estimation of the Effects of All Amino-Acid Mutations to HIV's Envelope Protein on Viral Replication in Cell Culture". en. In: *PLOS Pathogens* 12.12 (Dec. 2016). Ed. by Ronald Swanstrom, e1006114. URL: https://dx.plos.org/10.1371/journal.ppat.1006114 (visited on 01/23/2022).

[318] YQ Shirleen Soh et al. "Comprehensive mapping of adaptation of the avian influenza polymerase protein PB2 to humans". en. In: *eLife* 8 (Apr. 2019), e45079. URL: https://elifesciences.org/articles/45079 (visited on 01/23/2022).

[319] Zhifeng Deng et al. "Deep Sequencing of Systematic Combinatorial Libraries Reveals β-Lactamase Sequence Constraints at High Resolution". en. In: *Journal of Molecular Biology* 424.3-4 (Dec. 2012), pp. 150–167. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022283612007711 (visited on 01/23/2022).

[320] Justin R. Klesmith et al. "Comprehensive Sequence-Flux Mapping of a Levoglucosan Utilization Pathway in *E. coli*". en. In: *ACS Synthetic Biology* 4.11 (Nov. 2015), pp. 1235–1243. URL: https://pubs.acs.org/doi/10.1021/acssynbio.5b00131 (visited on 01/23/2022).

[321] Krystian A. Kozek et al. "High-throughput discovery of trafficking-deficient variants in the cardiac potassium channel KV11.1". en. In: *Heart Rhythm* 17.12 (Dec. 2020), pp. 2180–2189. URL: https://linkinghub.elsevier.com/retrieve/pii/S1547527120305427 (visited on 01/23/2022).

[322] Nicholas C. Wu et al. "High-throughput profiling of influenza A virus hemagglutinin gene at single-nucleotide resolution". en. In: *Scientific Reports* 4.1 (May 2014), p. 4942. URL: http://www.nature.com/articles/srep04942 (visited on 01/23/2022).

[323] Michael A. Stiffler, Doeke R. Hekstra, and Rama Ranganathan. "Evolvability as a Function of Purifying Selection in TEM-1 β-Lactamase". en. In: *Cell* 160.5 (Feb. 2015), pp. 882–892. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867415000781 (visited on 01/23/2022).

[324] Andre J Faure et al. "Mapping the energetic and allosteric landscapes of protein binding domains". In: *Nature* 604.7904 (2022), pp. 175–183.

[325] Paul Kennouche et al. "Deep mutational scanning of the *Neisseria meningitidis* major pilin reveals the importance of pilus tip-mediated adhesion". en. In: *The EMBO Journal* 38.22 (Nov. 2019). URL: https://onlinelibrary.wiley.com/doi/10.15252/embj.2019102145 (visited on 01/23/2022).

[326] Marion Sourisseau et al. "Deep Mutational Scanning Comprehensively Maps How Zika Envelope Protein Mutations Affect Viral Growth and Antibody Escape". en. In: *Journal of Virology* 93.23 (Dec. 2019). Ed. by Julie K. Pfeiffer. URL: https://journals.asm.org/doi/10.1128/JVI.01291-19 (visited on 01/23/2022).

[327] Emily E. Wrenbeck, Laura R. Azouz, and Timothy A. Whitehead. "Single-mutation fitness landscapes for an enzyme on multiple substrates reveal specificity is globally encoded". en. In: *Nature Communications* 8.1 (Aug. 2017), p. 15695. URL: http://www.nature.com/articles/ncomms15695 (visited on 01/23/2022).

[328] Rohan Dandage et al. "Differential strengths of molecular determinants guide environment specific mutational fates". en. In: *PLOS Genetics* 14.5 (May 2018). Ed. by Ivan Matic, e1007419. URL: https://dx.plos.org/10.1371/journal.pgen.1007419 (visited on 01/23/2022).

[329] Elad Firnberg et al. "A Comprehensive, High-Resolution Map of a Gene's Fitness Landscape". en. In: *Molecular Biology and Evolution* 31.6 (June 2014), pp. 1581–1592. URL: https://academic.oup.com/mbe/article-lookup/doi/10.1093/molbev/msu081 (visited on 01/23/2022).

[330] Mireia Seuma et al. "The genetic landscape for amyloid beta fibril nucleation accurately discriminates familial Alzheimer's disease mutations". en. In: *eLife* 10 (Feb. 2021), e63364. URL: https://elifesciences.org/articles/63364 (visited on 01/23/2022).

[331] Michael B. Doud, Orr Ashenberg, and Jesse D. Bloom. "Site-Specific Amino Acid Preferences Are Mostly Conserved in Two Closely Related Protein Homologs". en. In: *Molecular Biology and Evolution* 32.11 (Nov. 2015), pp. 2944–2960. URL: https://academic.oup.com/mbe/article-lookup/doi/10.1093/molbev/msv167 (visited on 01/23/2022).

[332] Parul Mishra et al. "Systematic Mutant Analyses Elucidate General and Client-Specific Aspects of Hsp90 Function". en. In: *Cell Reports* 15.3 (Apr. 2016), pp. 588–598. URL: https://linkinghub.elsevier.com/retrieve/pii/S2211124716303175 (visited on 01/23/2022).

[333] Julia M Flynn et al. "Comprehensive fitness maps of Hsp90 show widespread environmental dependence". en. In: *eLife* 9 (Mar. 2020), e53810. URL: https://elifesciences.org/articles/53810 (visited on 01/23/2022).

[334] Clara J. Amorosi et al. "Massively parallel characterization of CYP2C9 variant enzyme activity and abundance". en. In: *The American Journal of Human Genetics* 108.9 (Sept. 2021), pp. 1735–1751. URL: https://linkinghub.elsevier.com/retrieve/pii/S000292972100269X (visited on 01/23/2022).

[335] Christina Nutschel et al. "Systematically Scrutinizing the Impact of Substitution Sites on Thermostability and Detergent Tolerance for *Bacillus subtilis* Lipase A". en. In: *Journal of Chemical Information and Modeling* 60.3 (Mar. 2020), pp. 1568–1584. URL: https://pubs.acs.org/doi/10.1021/acs.jcim.9b00954 (visited on 01/23/2022).

[336] Jacob O Kitzman et al. "Massively parallel single-amino-acid mutagenesis". en. In: *Nature Methods* 12.3 (Mar. 2015), pp. 203–206. URL: http://www.nature.com/articles/nmeth.3223 (visited on 01/23/2022).

[337] Eric D. Kelsic et al. "RNA Structural Determinants of Optimal Codons Revealed by MAGE-Seq". en. In: *Cell Systems* 3.6 (Dec. 2016), 563–571.e6. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405471216303684 (visited on 01/23/2022).

[338] Juhye M. Lee et al. "Deep mutational scanning of hemagglutinin helps predict evolutionary fates of human H3N2 influenza variants". en. In: *Proceedings of the National Academy of Sciences* 115.35 (Aug. 2018), E8276–E8285. URL: http://www.pnas.org/lookup/doi/10.1073/pnas.1806133115 (visited on 01/23/2022).

[339] Florian Mattenberger et al. "Globally defining the effects of mutations in a picornavirus capsid". en. In: *eLife* 10 (Jan. 2021), e64256. URL: https://elifesciences.org/articles/64256 (visited on 01/23/2022).

[340] Kenneth A. Matreyek et al. "Multiplex assessment of protein variant abundance by massively parallel sequencing". en. In: *Nature Genetics* 50.6 (June 2018), pp. 874–882. URL: http://www.nature.com/articles/s41588-018-0122-z (visited on 01/23/2022).

[341] Samuel Thompson et al. "Altered expression of a quality control protease in E. coli reshapes the in vivo mutational landscape of a model enzyme". en. In: *eLife* 9 (July 2020), e53476. URL: https://elifesciences.org/articles/53476 (visited on 01/23/2022).

[342] Philip A. Romero, Tuan M. Tran, and Adam R. Abate. "Dissecting enzyme function with microfluidic-based deep mutational scanning". en. In: *Proceedings of the National Academy of Sciences* 112.23 (June 2015), pp. 7159–7164. URL: http://www.pnas.org/lookup/doi/10.1073/pnas.1422285112 (visited on 01/23/2022).

[343] Hangfei Qi et al. "A Quantitative High-Resolution Genetic Profile Rapidly Identifies Sequence Determinants of Hepatitis C Viral Fitness and Drug Sensitivity". en. In: *PLoS Pathogens* 10.4 (Apr. 2014). Ed. by Claus O. Wilke, e1004064. URL: https://dx.plos.org/10.1371/journal.ppat.1004064 (visited on 01/23/2022).

[344] Pradeep Bandaru et al. "Deconstruction of the Ras switching cycle through saturation mutagenesis". en. In: *eLife* 6 (July 2017), e27810. URL: https://elifesciences.org/articles/27810 (visited on 01/23/2022).

[345] Heather J. Young et al. *Deep Mutagenesis of a Transporter for Uptake of a Non-Native Substrate Identifies Conformationally Dynamic Regions*. en. preprint. Biochemistry, Apr. 2021. URL: http://biorxiv.org/lookup/doi/10.1101/2021.04.19.440442 (visited on 01/23/2022).

[346] Yvonne H. Chan et al. "Correlation of fitness landscapes from three orthologous TIM barrels originates from sequence and structure constraints". en. In: *Nature Communications* 8.1 (Apr. 2017), p. 14614. URL: http://www.nature.com/articles/ncomms14614 (visited on 01/23/2022).

[347] Melissa A Chiasson et al. "Multiplexed measurement of variant abundance and activity reveals VKOR topology, active site and human variant impact". en. In: *eLife* 9 (Sept. 2020), e58026. URL: https://elifesciences.org/articles/58026 (visited on 01/23/2022).

[348] L. M. Starita et al. "Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis". en. In: *Proceedings of the National Academy of Sciences* 110.14 (Apr. 2013), E1263–E1272. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1303309110 (visited on 01/23/2022).

[349] C. L. Araya et al. "A fundamental protein property, thermodynamic stability, revealed solely from large-scale measurements of protein function". en. In: *Proceedings of the National Academy of Sciences* 109.42 (Oct. 2012), pp. 16858–16863. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1209751109 (visited on 01/23/2022).

[350] Dan Davidi et al. "Highly active rubiscos discovered by systematic interrogation of natural sequence diversity". In: *The EMBO journal* 39.18 (2020), e104081.

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

[352] Courtney E Gonzalez, Paul Roberts, and Marc Ostermeier. "Fitness effects of single amino acid insertions and deletions in TEM-1 β-lactamase". In: *Journal of molecular biology* 431.12 (2019), pp. 2320–2330.

[353] Sam Sinai et al. "Generative AAV capsid diversification by latent interpolation". In: *bioRxiv* (2021).

[354] Benedetta Bolognesi et al. "The mutational landscape of a prion-like domain". In: *Nature communications* 10.1 (2019), pp. 1–12.

[355] Li Jiang et al. "A balance between inhibitor binding and substrate processing confers influenza drug resistance". In: *Journal of molecular biology* 428.3 (2016), pp. 538–553.

[356] Kenneth A Matreyek et al. "Integrating thousands of PTEN variant activity and abundance measurements reveals variant subgroups and new dominant negatives in cancers". In: *Genome medicine* 13.1 (2021), pp. 1–17.

[357] Karen S Sarkisyan et al. "Local fitness landscape of the green fluorescent protein". In: *Nature* 533.7603 (2016), pp. 397–401.

[358] Max V Staller et al. “A high-throughput mutational scan of an intrinsically disordered acidic transcriptional activation domain”. In: *Cell systems* 6.4 (2018), pp. 444–455.

[359] David Mavor et al. “Determination of ubiquitin fitness landscapes under different chemical stresses in a classroom setting”. en. In: *eLife* 5 (Apr. 2016), e15802. URL: https://elifesciences.org/articles/15802 (visited on 01/23/2022).

[360] Ethan Ahler et al. “A Combined Approach Reveals a Regulatory Mechanism Coupling Src’s Kinase Activity, Localization, and Phosphotransferase-Independent Functions”. en. In: *Molecular Cell* 74.2 (Apr. 2019), 393–408.e20. URL: https://linkinghub.elsevier.com/retrieve/pii/S1097276519300930 (visited on 01/23/2022).

[361] Martin Steinegger et al. “HH-suite3 for fast remote homology detection and deep protein annotation”. In: *BMC Bioinformatics* 20 (2019).

[362] Zihang Jiang et al. *ConvBERT: Improving BERT with Span-based Dynamic Convolution*. arXiv:2008.02496 [cs]. Feb. 2021. URL: http://arxiv.org/abs/2008.02496.

[363] Jannis Born and Matteo Manica. “Regression Transformer enables concurrent sequence regression and generation for molecular language modelling”. en. In: *Nature Machine Intelligence* 5.4 (Apr. 2023). Number: 4 Publisher: Nature Publishing Group, pp. 432–444. URL: https://www.nature.com/articles/s42256-023-00639-z.

[364] Louisa Gonzalez Somermeyer et al. “Heterogeneity of the GFP Fitness Landscape and Data-Driven Protein Design”. In: *eLife* 11 (May 2022), e75842.

[365] Nicholas C Wu et al. “Adaptation in Protein Fitness Landscapes Is Facilitated by Indirect Paths”. In: *eLife* 5 (July 2016). Ed. by Richard A Neher, e16965.

[366] Daniel D. Brauer et al. *Comprehensive Fitness Landscape of a Multi-Geometry Protein Capsid Informs Machine Learning Models of Assembly*. Dec. 2021.

[367] David Ding et al. “Co-Evolution of Interacting Proteins through Non-Contacting and Non-Specific Mutations”. In: *Nature Ecology & Evolution* 6.5 (May 2022), pp. 590–603.

[368] Jeffrey M. Spencer and Xiaoliu Zhang. “Deep Mutational Scanning of S. Pyogenes Cas9 Reveals Important Functional Domains”. In: *Scientific Reports* 7.1 (Dec. 2017), p. 16836.

[369] Philipp Koch et al. “Optimization of the Antimicrobial Peptide Bac7 by Deep Mutational Scanning”. In: *BMC Biology* 20.1 (May 2022), p. 114.

[370] Tobias Stadelmann et al. *A Deep Mutational Scanning Platform to Characterize the Fitness Landscape of Anti-CRISPR Proteins*. Aug. 2021.

[371] Sarah Gersing et al. “A Comprehensive Map of Human Glucokinase Variant Activity”. In: *Genome Biology* 24.1 (Apr. 2023), p. 97.

[372] Helen T. Hobbs et al. “Saturation Mutagenesis of a Predicted Ancestral Syk-family Kinase”. In: *Protein Science: A Publication of the Protein Society* 31.10 (Oct. 2022), e4411.

[373] Benjamin J Livesey and Joseph A Marsh. "Using deep mutational scanning to benchmark variant effect predictors and identify disease mutations". en. In: *Molecular Systems Biology* 16.7 (July 2020). URL: https://onlinelibrary.wiley.com/doi/abs/10.15252/msb.20199380 (visited on 09/22/2020).

[374] Ngak-Leng Sim et al. "SIFT web server: predicting effects of amino acid substitutions on proteins". en. In: *Nucleic Acids Research* 40.W1 (July 2012), W452–W457. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gks539 (visited on 09/08/2020).

[375] Nawar Malhis et al. "LIST-S2: taxonomy based sorting of deleterious missense mutations across species". In: *Nucleic acids research* 48.W1 (2020), W154–W161.

[376] Hashem A Shihab et al. "Predicting the functional, molecular, and phenotypic consequences of amino acid substitutions using hidden Markov models". In: *Human mutation* 34.1 (2013), pp. 57–65.

[377] Sung Chun and Justin C Fay. "Identification of deleterious mutations within three human genomes". In: *Genome research* 19.9 (2009), pp. 1553–1561.

[378] Daniel Quang, Yifei Chen, and Xiaohui Xie. "DANN: a deep learning approach for annotating the pathogenicity of genetic variants". In: *Bioinformatics* 31.5 (2015), pp. 761–763.

[379] Jana Marie Schwarz et al. "MutationTaster evaluates disease-causing potential of sequence alterations". In: *Nature methods* 7.8 (2010), pp. 575–576.

[380] Boris Reva, Yevgeniy Antipin, and Chris Sander. "Predicting the functional impact of protein mutations: application to cancer genomics". eng. In: *Nucleic Acids Research* 39.17 (Sept. 2011), e118.

[381] Yongwook Choi et al. "Predicting the functional effect of amino acid substitutions and indels". In: (2012).

[382] Biao Li et al. "Automated inference of molecular mechanisms of disease from amino acid substitutions". In: *Bioinformatics* 25.21 (2009), pp. 2744–2750.

[383] Kaitlin E Samocha et al. "Regional missense constraint improves variant deleteriousness prediction". In: *BioRxiv* (2017), p. 148353.

[384] Laksshman Sundaram et al. "Predicting the clinical impact of human mutation with deep neural networks". In: *Nature genetics* 50.8 (2018), pp. 1161–1170.

[385] Xiaoming Liu et al. "dbNSFP v4: a comprehensive database of transcript-specific functional predictions and annotations for human nonsynonymous and splice-site SNVs". In: *Genome medicine* 12.1 (2020), pp. 1–8.

[386] Wesley D Penn et al. "Probing biophysical sequence constraints within the transmembrane domains of rhodopsin by deep mutational scanning". In: *Science advances* 6.10 (2020), eaay7505.

[387] Kui K. Chan et al. "An engineered decoy receptor for SARS-CoV-2 broadly binds protein S sequence variants". In: *Science Advances* 7.8 (Feb. 2021). Publisher: American Association for the Advancement of Science, eabf1738. URL: https://www.science.org/doi/10.1126/sciadv.abf1738 (visited on 05/11/2023).

[388] Julia M Flynn et al. "Comprehensive fitness landscape of SARS-CoV-2 Mpro reveals insights into viral resistance mechanisms". In: *eLife* 11 (June 2022). Ed. by C Brandon Ogbunugafor et al. Publisher: eLife Sciences Publications, Ltd, e77433. URL: https://doi.org/10.7554/eLife.77433 (visited on 05/11/2023).

[389] Adam S. Dingens et al. "Complete functional mapping of infection- and vaccine-elicited antibodies against the fusion peptide of HIV". en. In: *PLOS Pathogens* 14.7 (July 2018). Publisher: Public Library of Science, e1007159. URL: https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1007159 (visited on 05/11/2023).

[390] Nicole N. Thadani et al. *Learning from pre-pandemic data to forecast viral escape*. en. Pages: 2022.07.21.501023 Section: New Results. Apr. 2023. URL: https://www.biorxiv.org/content/10.1101/2022.07.21.501023v2 (visited on 05/11/2023).

[391] Albert Tian Chen et al. "COVID-19 CG enables SARS-CoV-2 mutation and lineage tracking by locations and dates of interest". In: *eLife* 10 (Feb. 2021). Ed. by Jesse D Bloom and Miles P Davenport. Publisher: eLife Sciences Publications, Ltd, e63409. URL: https://doi.org/10.7554/eLife.63409 (visited on 05/11/2023).

[392] Joshua M. Dempster et al. "Extracting Biological Insights from the Project Achilles Genome-Scale CRISPR Screens in Cancer Cell Lines". In: *bioRxiv* (2019). eprint: https://www.biorxiv.org/content/early/2019/07/31/720243.full.pdf. URL: https://www.biorxiv.org/content/early/2019/07/31/720243.

[393] Damian Szklarczyk et al. "The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets". In: *Nucleic Acids Research* 49.D1 (2021), pp. D605–D612.

[394] Hyunghoon Cho, Bonnie Berger, and Jian Peng. "Compact integration of multi-network topology for functional analysis of genes". In: *Cell Systems* 3.6 (2016), pp. 540–548.

[395] Hyunghoon Cho, Bonnie Berger, and Jian Peng. "Diffusion component analysis: unraveling functional topology in biological networks". In: *International Conference on Research in Computational Molecular Biology*. Springer. 2015, pp. 62–64.

[396] Alexei Vazquez et al. "Global protein function prediction from protein-protein interaction networks". In: *Nature biotechnology* 21.6 (2003), pp. 697–700.

[397] Sovan Saha et al. "FunPred-1: Protein function prediction from a protein interaction network using neighborhood analysis". In: *Cellular and Molecular Biology Letters* 19.4 (2014), pp. 675–691.

[398] David P Nusinow et al. "Quantitative proteomics of the cancer cell line encyclopedia". In: *Cell* 180.2 (2020), pp. 387–402.

[399] Carlos G Sanchez et al. "Genome-wide CRISPR screen identifies protein pathways modulating tau protein levels in neurons". In: *Communications biology* 4.1 (2021), pp. 1–14.

[400] Kyunghyun Cho et al. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *SSST@EMNLP*. 2014.

[401] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2015).

[402] Ksenia Korovina et al. “ChemBO: Bayesian Optimization of Small Organic Molecules with Synthesizable Recommendations”. In: *AISTATS*. 2019.