Title

Understanding and Improving Stability for Biomedical Machine Learning Algorithms

ISBN

9798293893645

Author

Ronen, Omer

Publication Date

2025-08-01

Peer reviewed|Thesis/dissertation

Understanding and Improving Stability for Biomedical Machine Learning Algorithms

by

Omer Ronen

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Bin Yu, Chair
Professor Haiyan Huang
Professor Yun S. Song

Summer 2025

Understanding and Improving Stability for Biomedical Machine Learning Algorithms

Abstract

Understanding and Improving Stability for Biomedical Machine Learning Algorithms

by

Omer Ronen

Doctor of Philosophy in Statistics

University of California, Berkeley

Professor Bin Yu, Chair

This thesis studies machine learning algorithms whose primary applications are in biomedical domains, including clinical risk prediction, computational drug discovery, and genomic association studies. The first chapter investigates Bayesian Additive Regression Trees (BART), a widely used Bayesian algorithm for decision trees and random forests, and has been applied for clinical risk prediction. Despite its popularity, BART's posterior inference, which relies on Markov chain Monte Carlo sampling, has consistently shown slow mixing properties for reasons that have remained unclear. Through theoretical analysis of a simplified BART model, we derive a lower bound for its mixing time under mild data distribution assumptions. A key insight from this bound is that the number of training data points acts as a complexity parameter that degrades the mixing time. Extensive empirical studies demonstrate that this insight largely holds true for the full BART algorithm, albeit to a lesser degree. The second chapter focuses on the over-exploration problem in Latent Space Optimization (LSO). LSO is a technique that applies Bayesian Optimization to discrete, high-dimensional, and structured spaces, like those containing small molecules via their high-dimentional SMILES or SELFIES representations. In drug discovery, LSO plays a crucial role navigating the combinatorial chemical space to identify promising molecules for further experimentation. The search is conducted within the latent space of a Variational AutoEncoder (VAE), and it is known to over-explore, leading to unrealistic solutions. We tackle this problem by developing a constraint applicable to any LSO problem. This constraint is derived from the pushforward density of a random variable, transformed by the VAE decoder. We leverage a connection between deep neural networks and continuous piecewise affine spline operators to significantly accelerate its computation, and show it improves optimization performance for various LSO benchmarks.

In the third and fourth chapters, we study the role of representation-level diversity when employing Protein Language Models (PLMs) for transfer learning, by following the Predictability, Computability and Stability (PCS) framework. Chapter three shows how considering

multiple reasonable representations of a protein sequence, obtained from various PLMs, can improve protein fitness prediction and uncertainty quantification in the supervised prediction of deep mutational scanning experiments. Chapter four demonstrates how using multiple representations of DNA sequences—derived from protein language models (PLMs) via the central dogma of molecular biology—can enhance the ability of iterative Random Forests to identify genetic epistasis interactions involving rare variants in large-scale genomic studies. This improvement is validated by the successful recovery of interactions previously reported in the literature.

Contents

Acknowledgments

First, I would like to thank my advisor, Bin Yu, for her mentorship, feedback, and support over the last five years. Bin has tremendously influenced me throughout my Ph.D., particularly through her vision for developing rigorous and relevant statistical tools that solve real problems—a vision I deeply hope and intend to follow. Bin consistently emphasized during meetings and discussions the importance of grounding our research in real problems and developing theory and methods that truly align with the goals of the problem.

I am grateful to Bin for giving me the freedom to pursue topics that I found interesting, even when she had not worked on these topics before. Even in such projects, Bin would deeply engage in discussion and offer critical evaluation of my ideas—contributions that have been invaluable to my work. I am also deeply appreciative of Bin's generosity with her time and the genuine interest she has taken in me as both a researcher and a person. Bin has closely engaged in many of the projects I worked on at every stage—from problem formulation to critical evaluation of ideas to refining the writing, adding tremendous value through her intellectual breadth and experience.

As a researcher, I admire her uncompromising dedication to rigor and clarity, and her remarkable appetite for solving hard and important problems—an appetite that only grows as the challenges increase, as was the case for some of our genomics projects. Bin will always be someone I look up to for her persistence in tackling difficult problems.

As a person, Bin is truly one of the kindest and best individuals I have ever met. I am especially grateful for her support during difficult times in my Ph.D. I consider myself extremely lucky to have spent the past five years working with her.

I would like to thank the members of my thesis committee: Professor Haiyan Huang and Professor Yun Song. I am grateful to Haiyan for her insightful questions and comments during my qualifying exam. I am also grateful to Yun for serving as the chair of my qualifying exam and for his valuable advice in our joint meetings with Bin. Additionally, I'd like to thank Professor Adam Yala for serving on my qualifying exam committee.

I have been very lucky to collaborate with many great researchers during my time at Berkeley. I'd like to thank Yan Shuo Tan, who mentored me during the early years of my Ph.D. I'm also grateful to Chandan Singh, who I looked up to at the beginning of my time at Berkeley, and has since become a great friend. Chengzhong Ye's thoughtful approach to science taught me a great deal, for which I am thankful. I'm grateful to have collaborated with Ahmed Imtiaz Humayun, Randall Balestriero, and Professor Richard Baraniuk. Specifically, I appreciate Imtiaz's willingness to pursue my interest in drug design, Randall's positive attitude and close engagement, and Rich's support.

I'd also like to thank Tiffany Tang and Ana Kenney for their collaboration on various genomics projects, in which I learned a lot on how to do good applied science. I would like to thank Theo Saarinen, James Duncan and Abhineet Agarawal with whom I worked on different tree-related projects. My thanks also go to Aliyah Hsu for many insightful conversations. I would also like to thank Alex Zhao, Anqi Wang and Ron Boger.

Finally, I'd like to thank other members of the Yu Group whose time overlapped with mine: Wooseok Ha, Nikhil Ghosh, Spencer Frei, Jakob Heiss, Sufian Hayou, David Deriso, Nathan Benjamin McNaughton, Robin Netzorg, Anthony Ozerov, Zachary Rewolinski, Jingfeng Wu, Zeyu Yun, Austin Zane, Corine Elliot, and Luiz Chamon.

I am extremely thankful for the various people in the Statistics Department with whom I had the pleasure of interacting. I'd like to thank Tanisha, La Shana, and David for all their help and support throughout my time at Berkeley. My gratitude also goes to Chris, Ryan, and Dan for their assistance with various computing aspects, which made my life a lot easier. Finally, I'd like to thank Erez, Jeremy, Tianyu, and Licong for being wonderful friends.

My PhD would not have been possible without the support of Professor Or Zuk, Professor Yuval Benjamini and Professor Pavel Chigansky, who supported my application. I would like to thank Or for introducing me to the world of computational biology, I would like to thank Yuval for his mentorship and help throughout my application process and I would like to thank Pavel for getting me first excited about statistics.

Finally, I would like to sincerely thank my family. My parents Osnat and Rami, and my brothers Tomer and Guy, for their unwavering support throughout the last five years. Lastly, I am incredibly grateful to my partner and best friend, Hadar, who suported me through the lows and celebrated the highs with me over the last few years.

Chapter 1

Overview

Machine learning (ML) is transforming biomedical research, with applications ranging from protein structure and fitness prediction [62, 86, 58], computational drug design [44] to the identification of rare disease drivers [127] and clinical risk modeling [67]. Despite this progress, critical challenges—such as drug discovery [132], protein engineering [135], and understanding complex diseases [144, 143]—still require more robust and generalizable algorithms.

This dissertation addresses these challenges by advancing the theoretical understanding and practical performance of key ML methods applied to biomedical problems. It focuses on improving algorithmic stability, sampling efficiency, uncertainty quantification, and the efficeint integration of relevant biological information, with applications spanning clinical prediction, drug design, and genomics. A strong emphasis is placed on following the Predictability, Computability and Stability (PCS) framework [139, 138, 106], by conducting rigorous and extensive empirical validation.

In Chapter 2, we study Bayesian Additive Regression Trees (BART) [23], a Bayesian nonparametric regression algorithm used (among many other applications) for clinical risk prediction [115]. Access to BART's posterior is obtained via a Markov chain Monte Carlo sampler, which has been observed to have slow mixing times, leading to instability across multiple runs [54]. We provide the first theoretical analysis of BART's mixing time (through a simplified BART model), revealing a surprising trade-off between the number of training data points and the mixing time.

In Chapter 3, we explore Bayesian Optimization (BO) for computational drug discovery [44]. Previous BO techniques, which navigated molecular space using the latent space of a Variational AutoEncoder (VAE), often struggled to provide realistic solutions. We address this by developing Latent Exploration Score (LES), and a stable optimization method to constrain the search, leading to better and higher-quality solutions. Specifically, on the GuacaMol benchmark [18], using LES enables us to identify superior solutions that are free of PAINS [5]—molecules known to produce false positives in high-throughput experiments.

Chapters 4 and 5 are motivated by the goal of improving existing algorithms for protein engineering and epistasis discovery in genomic studies, by efficiently utilizing Protein

Language Models (PLMs [86, 107]). In Chapter 4, we study how embeddings from different PLMs can vary and how this difference can be exploited to improve fitness prediction and uncertainty quantification. In Chapter 5, we build on these insights and utilize embeddings from PLMs to improve the precision of iterative Random Forests (iRF) [9] for the discovery of epistasis interactions driven by rare genetic variants, using whole exome sequencing data from UK Biobank database [116]. Both improvements stem from following the PCS framework, which explicitly advocates using multiple data representations when there is no clear reason to favor a single one.

1.1 Understanding the Mixing Properties of Bayesian Additive Regression Trees (BART)

Chapter 2 studies the mixing time of a simplified version of BART [23]. BART's posterior is a distribution over sums of decision trees, and predictions are made by averaging approximate samples from the posterior. The combination of strong predictive performance and the ability to provide uncertainty estimates has led BART to be commonly used in the social sciences, biostatistics, and causal inference [54]. BART uses Markov chain Monte Carlo (MCMC) to obtain approximate posterior samples and it has often been observed that the chains are slow to mix [54], manifesting in different predictions across different MCMC chains. Our contribution is to conduct the first theoretical analysis for the mixing time of a simplified version of BART, which consists of a single tree model with a restricted set of moves. Our asymptotic lower bound of the mixing time is exponential in the size of the dataset, indicating that the mixing is slower when there are more data points. Our simulation studies show that this trend, in which mixing is slower with more data points, carries over to BART, though its manifestation is less severe. Simulations on four real-world datasets from the Penn Machine Learning Benchmark [111] show that both simplified BART and BART yield Gelman–Rubin diagnostic values [43] above 1.1—indicating a failure to mix—when more than 3000 data points are used.

This chapter is based on: Omer Ronen, Theo Saarinen, Yan Shuo Tan, James Duncan, and Bin Yu. "A mixing time lower bound for a simplified version of BART." arXiv preprint 2022.

An extended version of this work is currently being re-submitted to the Annals of Statistics: Yan Shuo Tan, Omer Ronen, Theo Saarinen, and Bin Yu (2024). The Computational Curse of Big Data for Bayesian Additive Regression Trees: A Hitting Time Analysis. arXiv preprint arXiv:2406.19958.

1.2 Mitigating Over Exploration in Bayesian Optimization for Drug Design

Bayesian Optimization (BO) is a widely used framework for tackling black-box optimization problems where evaluating the true objective function is computationally expensive [39]. These types of problems are common in fields like computational drug design, where the objective—for example, the binding affinity of a molecule to a given protein pocket—may only be measurable through costly and time-consuming wet-lab experiments [44].

BO operates iteratively, selecting a set of candidate solutions for evaluation at each step. It does so by employing a surrogate model—typically a Gaussian Process (GP), which defines a posterior distribution over the space of possible objective functions. Based on this posterior distribution, an acquisition function is designed to balance exploration and exploitation, and is then optimized to propose the next set of candidates for evaluation.

For discrete, high-dimensional objects such as small molecules—encoded as 4,200- and 10,512-dimensional arrays under the SMILES and SELFIES [69] representations (in our experiments), respectively—a common strategy is to optimize in the latent space of a pre-trained VAE [44], which maps the high-dimensional molecular representation to a lower-dimensional latent space. This strategy offers several key advantages:

- Dimensionality Reduction: The latent space is chosen to be lower-dimensional, which significantly reduces the statistical complexity of training an accurate GP.
- Continuous Representation: The continuous nature of the latent space enables the use of gradient-based optimization methods for the acquisition function, making the optimization process more efficient.
- Leveraging Unlabeled Data: VAEs can potentially learn meaningful representations from unlabeled data, which is often abundant.

Chapter 3 of this thesis specifically focuses on the optimization procedure of the acquisition function when this optimization is performed within the latent space of a VAE. It has been widely observed [120, 48] that a naive optimization of any reasonable acquisition function in this setting often leads to unrealistic solutions.

Our contribution is the development of a Latent Exploration Score (LES), which can be used as a constraint. This constraint encourages the optimization process to remain close to the regions within the latent space where the VAE training data are concentrated. The constraint itself is defined as the pushforward density of a latent vector under the VAE's decoder transformation. We develop a novel method to significantly speed up its computation by leveraging a connection between deep neural networks and continuous piecewise affine splines ([7, 59, 60]). Extensive experiments demonstrate the benefits of incorporating LES as a constraint in latent space optimization (LSO). Specifically, in 19 out of 30 LSO benchmark experiments, LES-constrained optimization either achieves the best solution or a solution within one standard error of the best, outperforming all six other optimization

methods we benchmarked.

This chapter is based on: Omer Ronen, Ahmed Imtiaz Humayun, Richard Baraniuk, Randall Balestriero, and Bin Yu. "Mitigating over-exploration in latent space optimization using LES." In International Conference on Machine Learning (ICML), 2025.

1.3 Stabilizing Protein Fitness Predictors via the PCS Framework

Computational predictors of protein fitness are an important component of protein engineering campaigns [135]. Inspired by empirical results on transfer learning in vision and text, embeddings from Protein Language Models (PLMs) are commonly used to represent protein sequences. However, the space of potential embedding vectors that can be derived from even a single PLM is huge [133]. While the most common choice are the mean-pooled (over the sequence dimension) embeddings from the penultimate layer, recent studies have shown that other choices, such as middle layers and different pooling strategies [75], can perform equally well in many cases. This makes the default choice a rather arbitrary one among a huge space of reasonable options.

Chapter 4 studies the representation level instability (that is, the particular choice of embedding) in the context of protein fitness prediction. Following the PCS framework, we propose a simple two-step procedure to add stability to various protein fitness predictors, such as neural networks, linear models, and Gaussian processes. Our procedure first weights embeddings according to their predictive performance and then ensembles models trained on each specific embedding. This simple procedure leads to significant improvements in prediction accuracy and uncertainty quantification through the ensemble variance on the supervised ProteinGym benchmark.

This chapter is based on: Omer Ronen, Alex Y. Zhao, Ron Boger, Chengzhong Ye, and Bin Yu. "Stabilizing protein fitness predictors via the PCS framework." In The Exploration in AI Today Workshop at ICML 2025.

1.4 Ultra-Stable iRF: Enhancing Epistasis Detection Precision in Rare Genetic Variants Studies

Identifying epistatic interactions, where the effect of a certain gene mutation is dependent on the presence or absence of another, presents both a computational and a statistical challenge [144]. The sheer size of human DNA makes computational discovery difficult, while the large number of potential interactions poses a significant statistical and computational hurdle.

While progress on epistasis discovery has been slow, decision trees have recently emerged as a promising method [10, 127]. This is due to their ability to model biochemical interactions and their inherent interpretability. Decision trees have successfully identified epistatic interactions for conditions such as red hair, multiple sclerosis, cardiac hypertrophy and hypertrophic cardiomyopathy.

A significant challenge remains in extending these methods to rare genetic variants, defined as those with a minor allele frequency (MAF) less than 0.1%. The aforementioned progress has primarily focused on common genetic variants (MAF greater than 5%).

In Chapter 5, we develop a procedure based on iterative Random Forests (iRF [9]) for discovering gene-gene epistatic interactions, driven by rare genetic variants, using large scale whole-exome sequencing data. Building on the results in Chapter 4, we further stabilize iRF by considering multiple numeric representations for SNP data. Specifically, leveraging the central dogma of molecular biology, for SNPs resulting in missense mutations, we derive numeric representations of the corresponding protein sequences using PLMs in addition to the traditional dose representation of the SNP data. The final score for an interaction is given by the average score obtained by iRF across these four representations.

Our study of 20 quantitative phenotypes (studied in previous works [85, 28]) from the UK Biobank [116] database demonstrates that by averaging iRF stability scores across these four representations we are able to significantly increase its precision in recovering previously known interactions from two databases: STRING [118] and BioGRID [97]. In particular, when testing for gene–gene interactions affecting 19 quantitative traits from the UK Biobank [116], 59% of the top 9% of interactions discovered by US-iRF had support in the STRING database, and 13% of the top 6% had experimental support in BioGRID. For comparison, the corresponding STRING support rates were 51% for iRF with the traditional dose representation, and 24% for MatrixEpistasis [142] (a linear interaction model); the corresponding BioGRID support rates were 7%, and 5%, respectively.

This chapter is based on: Omer Ronen, Chengzhong Ye, and Bin Yu. "Ultra-Stable iRF: Enhancing Epistasis Detection Precision in Rare Genetic Variants Studies." Manuscript in preparation (2025).

Chapter 2

Understanding the Mixing Properties of Bayesian Additive Regression Trees (BART)

2.1 Introduction

Decision tree models such as CART [17] and their ensembles such as Random Forests [16] and Gradient Boosted Trees [40, 21] have proved to be enormously successful supervised learning algorithms, because they are able to combine non-parametric model fitting with implicit dimension reduction. It is often difficult to quantify the uncertainty of their predictions and due to their greedy local splitting criteria, there is no guarantee for the optimality of the constructed decision trees.

To address these issues, Chipman et al. [24] proposed a Bayesian adaptation of CART, Bayesian CART, and later, a sum of Bayesian CART trees, which they called Bayesian Additive Regression Trees (BART) [23]. One perspective views these algorithms as non-greedy stochastic versions of their deterministic equivalents, where the randomness inside the fitting process allows the algorithm to explore the space of possible decision trees in ways the CART algorithm cannot. An alternative perspective views these algorithms as Bayesian non-parametric regression models, in which we put a prior on the space of decision trees, assume a likelihood for the observed data, and then obtain a posterior distribution over the possible decision trees based on the training data. The posterior distribution can be used to provide posterior predictive credible intervals and other forms of uncertainty quantification. Due to strong predictive performance [22, 23] and the ability to quantify the uncertainty of predictions, BART has spawned a number of variants [56, 114, 79, 52, 103, 88] and has become increasingly popular in fields such as the social sciences [45, 137], biostatistics [129, 115], and causal inference [56, 46, 65, 35, 51, 54].

Several recent works have theoretically analyzed BART variants from a frequentist perspective. Concentration results have been shown for the BART posterior under assumptions

on the smoothness of the underlying regression function and the prior used for the BART algorithm. Ročková et al. [109] show that when the BART prior is modified to decrease the mass on deeper trees as the number of training points increases, the posterior distribution of this BART variant concentrates around the true regression function at nearly the optimal rate. When the true regression function exhibits smoothness, Linero et al. [79] introduce a smoothing variant of BART and show that the modified BART posterior concentrates at nearly the optimal rate for that class of functions. In addition to concentration, there are several results for the feature selection consistency of the BART posterior. When the regression function is smooth and sparse, Ročková et al. [108] show that BART with a sparsity inducing prior can perform effective feature selection. Liu et al. [82] examine an approximate Bayesian computation algorithm to combine with BART for feature selection in high dimensional data. However, all of these theoretical results rely on the ability to sample from the BART posterior.

2.1.1 Prior Work on BART MCMC Mixing

Since there is no closed form expression for the BART posterior, the standard approach is to sample from it approximately via a Markov chain Monte Carlo (MCMC) algorithm designed by Chipman et al. [23]. Despite an abundance of empirical evidence, described following this, that the posterior samples from the BART algorithm do not mix well, researchers in many fields have regularly used the BART posterior for uncertainty quantification [46, 55, 125, 14, 35, 137, 141, 19]. Further, there has been very little theoretical analysis of how quickly the samples from the BART MCMC algorithm converge to the posterior distribution. This is problematic for several reasons: First, credible intervals obtained from the MCMC samples will not actually reflect the posterior, and are therefore of questionable meaning for inference. Next, it means that different runs of the algorithm may produce somewhat different results, thereby lacking stability and reproducibility. Finally, a necessary condition for the recent results on posterior concentration and model selection consistency, is the ability to sample from the posterior distribution. When the BART MCMC algorithm is slow to mix, it is difficult to satisfy this condition in practice.

Since the introduction of the BART algorithm, the problem of mixing has been observed, [23, 102, 53] leading to a number of works evaluating and attempting to address this issue. The difficulties in mixing have been explored empirically in several works [24, 131, 102] and mentioned in several recent survey papers [78, 54]. There have also been several algorithmic suggestions including modifying the MCMC proposal moves [102, 131], initializing the trees in the algorithm from greedily constructed trees as a "warm start" [53], and running multiple chains to quantify the uncertainty in the predictions [35, 20]. Despite the prevalence of works mentioning the mixing of the BART MCMC algorithm, little theoretical work has been done to understand why the mixing is slow.

Table 2.1: A summary of the different algorithms studied in this paper.

Algorithm \ Property	Number of Trees	MCMC moves
BART	200	**Grow**, **Prune**, **Change**, **Swap**
Bayesian CART	1	**Grow**, **Prune**, **Change**, **Swap**
Simplified BART	1	**Grow**, **Prune**

2.1.2 BART and Bayesian CART

In this chapter we will discuss several variants of the BART algorithm. The Bayesian CART algorithm was originally introduced as an algorithm for fitting a single decision tree in a Bayesian setting. The BART algorithm fits a sum of these Bayesian CART trees and when this sum has one term, devolves to the Bayesian CART algorithm. BART enjoys better prediction accuracy than Bayesian CART [22, 23, 54], and is hence used more often in practice. Because Bayesian CART and each individual tree in the BART sum share the same set of MCMC proposal moves, Bayesian CART has been used to study the empirical effect of these proposal moves on the mixing of both algorithms [131, 23, 102, 54]. In our theoretical work, we begin by using a simplified version of Bayesian CART towards the goal of explaining the properties of BART. For theoretical tractability, we restrict the MCMC proposals available to the Bayesian CART algorithm to a subset of the standard moves and call the resulting algorithm *simplified BART*. We present our theoretical mixing time lower bound for the simplified BART algorithm. We compare the mixing time of BART to simplified BART in simulations in Section 2.4.1, and compare Bayesian CART to simplified BART in Section 2.4.2 in order to compare the propensity of the two algorithms to select poor initial splitting features.

2.1.3 Our contributions

We present a theoretical result for the failure of the simplified BART MCMC algorithm to mix and show through extensive simulations that the BART MCMC algorithm has a similar mixing issue. As far as we know, this is the first attempt at theoretically analyzing the mixing behavior of a simplified version of the Bayesian CART or BART algorithms, with insights that also hold qualitatively for BART. Our specific contributions are as follows:

1. **Mixing time lower bounds:** We obtain a mixing time lower bound for a simplified version of the BART algorithm. Assuming that all features are discrete, we are able to show that the total variation mixing time of the simplified BART MCMC chain on the space of tree structures is bounded from below by $\exp(\Omega(n))$, where n is the number of training data points. This theoretical result holds for any data generating process with features from any random distribution on a discrete feature space of dimension at least 2 and a random outcome vector y. In addition, our proof identifies one reason

for the slow mixing: It is difficult for the chain to switch the split at the root of the tree.

2. **Simulations:** Taking several large real datasets from the Penn Machine Learning Benchmark repository (PMLB) [96], we run 8 independent instances of the MCMC chain for BART and for simplified BART and study the mixing of the chains according to several metrics. We use the root mean squared error (RMSE) from a held out test set as a summary statistic of each sampled regression function, and compute the distribution of this quantity over each chain after discarding a conservative number of burn-in samples. Through our simulations, we find that *the BART algorithm shows worse mixing as the number of training data points increases*, as indicated by mixing diagnostics as well as qualitative measures of the regression function. We also find that Bayesian CART is susceptible to the bottleneck that we exploit for simplified BART in the proof of our theoretical result: Difficulty in reversing the split at the root of the tree. We did not find strong evidence that this bottleneck affects the BART algorithm to the same degree.

To the extent of our knowledge, our theoretical result is the first that studies the mixing time of either the BART or Bayesian CART algorithms. Our simulations suggest that the problems identified theoretically for simplified BART still affect the original BART and Bayesian CART algorithms. In particular, the issue of mixing time increasing with the number of data points suggests that BART practitioners may benefit by running more chains for data with many observations, an observation not yet explored in the literature.

2.2 Preliminaries

In this section, we describe the elements of the Bayesian CART model underlying the simplified BART algorithm as well as the MCMC sampling algorithm, as described in the original paper [24]. We define the Bayesian CART model formally because it only differs from the simplified BART algorithm we analyze in the set of MCMC moves.

2.2.1 The Bayesian model specification

Parameter space: Recall that the Bayesian CART model is a non-parametric regression model, which means that the regression function f is a random function. Unlike Gaussian process regression, we constrain it to take values on the space of decision tree functions. We say that $f\colon \mathcal{X}^d \to \mathbb{R}$ is a decision tree function if it is a piecewise constant function on a partition of $\mathcal{X}^d$, where the partition is generated by recursively splitting $\mathcal{X}^d$ along the coordinate directions. Naturally, f can be parameterized by a tuple comprising the binary tree structure $\mathcal{T}$, and $\mathbf{\Theta} \in \mathbb{R}^b$, where $b = |\mathcal{T}|$ is the number of leaves in $\mathcal{T}$, and $\mathbf{\Theta}_i$ is the value of f on leaf i (after choosing a canonical ordering of the leaves). In turn, the tree structure $\mathcal{T}$ can be thought of as a labeled graph, and thus further parameterized by the

underlying ordered rooted binary tree, as well as labels on each internal node of the form (v_j, τ_j), which respectively denote the feature and threshold for the split associated with node j. We assume that our covariate space is discrete, i.e. $\mathcal{X} = \{1, \ldots, m\}$ for some integer m.[1] This implies that a tree structure $\mathcal{T}$ can have at most m^d leaves, which, together with a finite choice of labels at each node, implies that Ω, the collection of all tree structures, is discrete and finite.

Prior for $\mathcal{T}$: The prior on the tree structure is defined in terms of a stochastic process: Starting with a trivial tree with a single node, each newly generated node is split with probability $\alpha(1+\delta)^{-\beta}$, where δ is the depth of the node, while α and β are universal hyperparameters. If a node is split, the feature it splits on is drawn uniformly from all available features, and then the threshold is drawn uniformly from all available values of the feature, if it is discrete, and from all available values that have been observed in the training data, if it is continuous.

Prior for Θ: We put independent Gaussian priors for the leaf parameters: $\mathbf{\Theta}|\mathcal{T} \sim \mathcal{N}(\bar{\mu}\mathbf{1}, \sigma^2\mathbf{I}_b)$. In the original Bayesian CART algorithm, σ^2 is treated as a Bayesian parameter drawn from an inverse Gamma distribution. For the sake of theoretical tractability, we shall treat it instead as a fixed hyperparameter. We believe that this is relatively innocuous as, in practice, σ^2 is known to quickly concentrate around a fixed value.

Data likelihood: We put an independent Gaussian likelihood function on the errors in the responses. In other words, given observed data $\mathbf{X} = (\boldsymbol{x}_i)_{i=1}^n$, $\boldsymbol{y} = (y_i)_{i=1}^n$, we assume

$$\boldsymbol{y}|\mathbf{X}, \Theta, \mathcal{T} \sim \mathcal{N}\big((f(\boldsymbol{x}_i))_{i=1}^n, a\sigma^2\mathbf{I}_n\big),$$

where $a > 0$ is a fixed number.

The relationships between the variables described in this section can be summarized in a Bayesian network diagram (Figure 2.1).

2.2.2 Sampling from the Bayesian CART posterior

To sample from the posterior $p(\mathcal{T}, \mathbf{\Theta}|\mathbf{X}, \boldsymbol{y})$, we first decompose it as $p(\mathbf{\Theta}|\mathcal{T}, \mathbf{X}, \boldsymbol{y})p(\mathcal{T}|\mathbf{X}, \boldsymbol{y})$. The Gaussian specification of Θ and the data likelihood allow us to compute the first term in closed form, and so, we only need to use MCMC to sample from $p(\mathcal{T}|\mathbf{X}, \boldsymbol{y})$, the marginal posterior on the space of tree structures. We do this using the Metropolis-Hastings algorithm. When the chain is at a given tree $\mathcal{T}$, the proposed next tree $\mathcal{T}'$ is obtained by randomly choosing from one the following four moves: 1. **Grow** the tree by splitting a leaf node chosen uniformly at random. The split feature and threshold are also chosen uniformly at

[1] This is almost WLOG as in practice, splits for continuous features are chosen from a grid of possible values corresponding to the quantiles of the features in the training data.

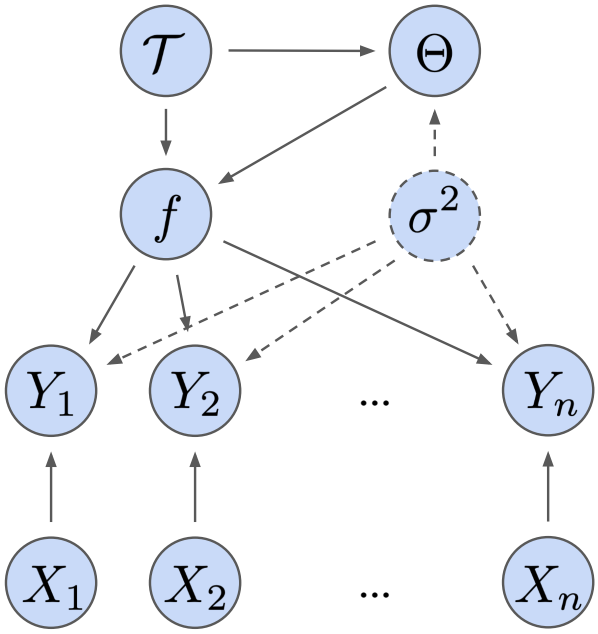

Figure 2.1: A Bayesian network showing the dependency relationships between the random variables in the simplified BART model. σ^2 is treated as a parameter in Bayesian CART, but will be treated as a fixed hyperparameter in our theoretical analysis.

random, as in the prior. 2. **Prune** the tree by collapsing a pair of adjacent leaf nodes chosen uniformly at random. 3. **Change** the split feature and threshold (draw them again from the uniform distribution) of an internal node selected uniformly at random. 4. **Swap** the split features and thresholds of a parent-child node pair chosen uniformly at random. An accept-reject filter is then applied to ensure that the chain has the marginal posterior as the stationary distribution. The original Bayesian CART paper proposed selecting each move with equal probability. Later when proposing BART, the authors updated their suggestion to use the probabilites $(0.25, 0.25, 0.4, 0.1)$.

2.2.3 Simplified BART

We define the simplified BART algorithm as the Bayesian CART algorithm defined in Section 2.2.2 with the MCMC moves restricted to **Grow** and **Prune** each of which we select with probability .5. This is the algorithm that we theoretically analyze in Section 2.3.

2.2.4 Contrasting Bayesian CART with BART

BART is Bayesian model for a tree sum that is built on top of Bayesian CART. The same priors on tree structure and leaf parameters are used, while the likelihood function of the noise in the response remains Gaussian. However, the regression function is now the sum of all the tree functions in the sum, and the parameter space thus comprises the product $(\mathcal{T}_i, \mathbf{\Theta}_i)_{i=1}^M$, where M is the number of trees. To sample from the resulting posterior, we use a combination of Gibbs sampling and the Metropolis-Hastings algorithm. More precisely, we cycle through the trees in the model and update them as follows: Given a tree $\mathcal{T}_i$ in the model, conditioned on the values of parameters from all the other trees, we update $\mathcal{T}_i$ using the same procedure as in Bayesian CART, except that we replace the responses with residuals in the likelihood function. We then make a single draw from the conditional posterior for $\mathbf{\Theta}_i$, and then repeat the process for the next tree in the sum. A single update of the MCMC chain comprises an update for all M trees in the sum.

2.3 Mixing Time Lower Bound

Notation on probabilities: We will use $p(-)$ to denote the marginal and conditional probabilities associated with the Bayesian model described in Section 2.2.1. We use $Q(-)$ to denote probabilities associated with the Markov chain for the simplified BART algorithm on Ω, the space of tree structures described in Section 2.2.2. As a shorthand, we will also denote the stationary distribution, the posterior marginal on Ω, as π. Finally, we assume that our observed data $(\mathbf{X}, \boldsymbol{y})$ comprise n i.i.d. data points from a generative regression model with regression function $f_0(\boldsymbol{x}) = \mathbb{E}\{y \mid \boldsymbol{x}\}$. We will denote probabilities, expectations, and variances with respect to the generative process using $\mathbb{P}$, $\mathbb{E}$, and Var. Note that the generative model is not necessarily the same as the data likelihood in the fitted Bayesian model.

The mixing time (see also Levin et al. [73]) of the Markov chain of the simplified BART algorithm, Q, is defined to be

$$t_{mix} \coloneqq \min\{t : \max_{\mathcal{T} \in \Omega} \|Q^t(-|\mathcal{T}) - \pi\|_{\text{TV}} \leq 0.25\}. \tag{2.1}$$

The quantity being maximized over in the right hand side of the equation is the total variation distance between the time t distribution of the chain initialized at a tree structure $\mathcal{T}$ and the stationary distribution. A larger mixing time means that the chain takes a long time to reach the stationary distribution from a worst case initialization.[2]

The main theoretical result of our paper is the following theorem.

Theorem 1 (Mixing time lower bound)**.** *Suppose $d \geq 2$ or $m \geq 2$. Also assume that y has a bounded distribution, i.e. $|y| \leq K$. Then with probability at least $1 - 1/n$, the mixing time*

[2]The choice of 0.25 as the threshold is by convention and does not affect the mixing time up to multiplicative constants.

of the simplified BART Markov chain, Q, as described in Section 2.2.3, satisfies

$$t_{mix} \geq \exp\left(\frac{n}{2a\sigma^2}\text{Var}\{f_0(\boldsymbol{x})\} - O(\sqrt{n\log n})\right). \tag{2.2}$$

The theorem says that under trivial assumptions, the mixing time of the simplified BART MCMC algorithm grows exponentially in the number of data points. This is surprising and unfortunate because we generally expect the performance of prediction algorithms to improve with more data points.

Our proof of the lower bound proceeds by constructing two tree structures $\mathcal{T}$ and $\mathcal{T}'$ that each give rise to a partition on which f_0 is piecewise constant, but whose splits at the root node differ from each other. This means that both $\mathcal{T}$ and $\mathcal{T}'$ possess large posterior mass, but are separated from each other in the state space by a bottleneck for the chain – the trivial tree with only a root node. This bottleneck becomes increasingly difficult to traverse as the number of data points increases.

While the result concerns mixing for the tree structure $\mathcal{T}$ and not the regression function f, in most cases we may select $\mathcal{T}$ and $\mathcal{T}'$ so that the partition associated with $\mathcal{T}$ is a refinement of that associated with $\mathcal{T}'$. In this case, the resulting conditional posterior distributions $p(f \mid \mathcal{T}, \mathbf{X}, \boldsymbol{y})$ and $p(f \mid \mathcal{T}', \mathbf{X}, \boldsymbol{y})$ are different, which strongly suggests that the stochastic process induced by Q on the space of regression functions[3] also mixes slowly with respect to Wasserstein distances. Such a result, while even more revealing, would be relatively technically difficult, and we leave it to future work.

The notion of a bottleneck is formalized by the definition of *conductance*. The conductance of a subset $S \subset \Omega$ is defined as the ratio

$$\Phi(S) := \frac{Q(S, S^c)}{\min\{\pi(S), \pi(S^c)\}},$$

where

$$Q(S, S^c) = \sum_{\mathcal{T}\in S, \mathcal{T}'\in S^c} \pi(\mathcal{T})Q(\mathcal{T}'|\mathcal{T})$$

is the probability of starting in S and ending in S^c over one step of Q when applied to the stationary distribution. A small conductance implies that it is difficult for Q to transit from S to S^c. This intuition can be used to derive the following well-known inverse relationship between mixing time and conductance.

Lemma 2 (Conductance and mixing time, Theorem 7.3 in Levin et al. [73])**.**

$$t_{mix} \geq \frac{1}{4\Phi^*}$$

[3]The transition kernel on the full parameter space is given by $\tilde{Q}(\mathcal{T}', \boldsymbol{\Theta}'|\mathcal{T}, \boldsymbol{\Theta}) = Q(\mathcal{T}'|\mathcal{T})p(\boldsymbol{\Theta}'|\mathcal{T}', \mathbf{X}, \boldsymbol{y})$. Since the tuple $(\mathcal{T}, \boldsymbol{\Theta})$ determines f, this chain induces a stochastic process on the space of regression functions.

where t_{mix} is the mixing time of the simplified BART algorithm defined in 2.1 and

$$\Phi^* := \min_{S \subset \Omega} \Phi(S).$$

By using the bottleneck at the trivial tree alluded to earlier, we can upper bound the conductance in terms of the minimum of two likelihood ratios, each of which compares the trivial tree to a nontrivial one making a different root split.

Lemma 3 (Conductance and likelihood ratio)**.** *Let $\mathcal{T}$ and $\mathcal{T}'$ be two tree structures whose splits at the root node differ from each other.*

Then

$$\Phi^* \leq C \min\left\{ \frac{p(\boldsymbol{y}|\mathcal{T}_0, \mathbf{X})}{p(\boldsymbol{y}|\mathcal{T}, \mathbf{X})}, \frac{p(\boldsymbol{y}|\mathcal{T}_0, \mathbf{X})}{p(\boldsymbol{y}|\mathcal{T}', \mathbf{X})} \right\}, \tag{2.3}$$

where $C = C(\mathcal{T}, \mathcal{T}', \alpha, \beta, m, d)$ is a constant not depending on n, and $\mathcal{T}_0$ is the trivial tree comprising only a root node.

To further bound each likelihood ratio in (2.3), we formulate a more general result for the log likelihood ratio between the trivial tree and *any* candidate tree $\mathcal{T}$. The result shows that the normalized log likelihood ratio concentrates around the population *impurity decrease* between $\mathcal{T}_0$ and $\mathcal{T}$.

Lemma 4 (Likelihood ratio and impurity decrease)**.** *For any tree $\mathcal{T}$, with probability at least $1 - 1/n$, we have*

$$\begin{aligned} &\left| \log \frac{p(\boldsymbol{y}|\mathcal{T}_0, \mathbf{X})}{p(\boldsymbol{y}|\mathcal{T}, \mathbf{X})} + \frac{n\Delta}{2a\sigma^2} \right| \\ &\leq CK^2 a^{-1} \sigma^{-2} \sqrt{\log n} \big(\sqrt{n} + b \big) + b \log(n/a). \end{aligned}$$

where

$$\Delta := \mathrm{Var}\{f_0(\boldsymbol{x})\} - \mathbb{E}\{\mathrm{Var}\{f_0(\boldsymbol{x}) \mid \mathcal{T}(\boldsymbol{x})\}\}$$

is the decrease in impurity of $f_0(\boldsymbol{x})$ when conditioning on the partition given by $\mathcal{T}$, and C is an absolute constant.

The empirical impurity decrease is used by CART to choose which splits to make and which to prune, while the likelihood ratio is used by Bayesian CART to accept or reject proposed grow, prune, change, or swap moves. This lemma thereby further illustrates the similarities and differences between the two algorithms. More importantly for this paper, it completes the toolbox we need to prove our main theorem.

Proof of Theorem 1. By assumption, there are at least two possible splits at the root node, (v, τ) and (v', τ'). Starting from either split, it is possible to grow a tree on whose leaves the true regression function is piecewise constant. Call these two trees $\mathcal{T}$ and $\mathcal{T}'$ respectively,

and notice that they automatically satisfy the hypotheses of Lemma 3 so that (2.3) holds. Furthermore, we have

$$\mathbb{E}\{\mathrm{Var}\{f_0(\boldsymbol{x})|\mathcal{T}(\boldsymbol{x})\}\} = \mathbb{E}\{\mathrm{Var}\{f_0(\boldsymbol{x})|\mathcal{T}'(\boldsymbol{x})\}\} = 0.$$

By Lemma 4, we therefore have

$$\log \frac{p(\boldsymbol{y}|\mathcal{T}_0, \mathbf{X})}{p(\boldsymbol{y}|\tilde{\mathcal{T}}, \mathbf{X})} = -\frac{n}{2\sigma^2}\mathrm{Var}\{f(\boldsymbol{x})\} + O(\sqrt{n \log n})$$

for $\tilde{\mathcal{T}} = \mathcal{T}, \mathcal{T}'$. Exponentiating and applying Lemma 3 followed by Lemma 2 then gives us (2.2). □

2.4 Simulations on Mixing for BART and simplified BART

In this section, we perform two simulation experiments to understand how the conclusions from our theoretical analysis of simplified BART in the previous section provide new insights for understanding BART and Bayesian CART. Specifically our simulations suggest that:

- The BART MCMC algorithm often mixes poorly, and the mixing quality decreases as the number of training data points n increases.
- The root split made by the Bayesian CART algorithm is often chosen suboptimally and yet is rarely reversed, thereby creating a bottleneck for the chain.

Data: We use real-world datasets in order to emulate how BART is used in practice. Specifically, we utilize the four largest datasets from the Penn Machine Learning Benchmarks (PMLB) [96] in order to study how the mixing time depends on the number of training data points. Table 2.2 details the dimensions of these datasets.

Table 2.2: PMLB datasets utilized in simulations

Name	Samples	Features
Breast tumor [110]	116640	9
California housing [98]	20640	8
Echo months [110]	17496	9
Satellite image [110]	6435	36

Algorithm settings and hyperparameters: We use the **dbarts** R package [34] with the following non-default hyperparameters. First, we increase the number of burn-in samples from 100 to 5000 (*nskip*=5000), in order to highlight that mixing does not occur within a reasonable number of iterations. Second, we run 8 chains (*nchain*=8) to facilitate the analysis of the mixing time. Last, we define the proposal distribution of simplified BART to select grow and prune with equal probability (*probs*=(1-1e-5,1e-5,0,.5)[4]). All other hyperparameters are kept at their default values. In particular, for simplified BART we use *ntree*=1 and for BART we use *ntree*=200.

2.4.1 Experiment 1: Mixing time increases with n

Choice of mixing metric: Each sample in a chain is a function, and we calculate its RMSE on a held-out evaluation set, comprising 10% of the dataset, as a one-dimensional summary statistic. To quantify the degree of mixing, we compute the Gelman-Rubin convergence diagnostic (GR) [43], which compares the within-chain variation to the across-chain variation.[5] It is widely accepted that a GR value which exceeds 1.1 indicates failure to mix [43]. We implicitly assume that the degree of mixing at a fixed number of iterations is indicative of the mixing time.

Varying n: To vary n, we draw random sub-samples without replacement from each dataset of sizes $n = 200$ and $n = 2000$ in addition to using the full dataset.

Results: Across 20 experimental replicates, we compute the GR diagnostic value for BART at each configuration described previously, and summarize our results in Figure 2.2. For comparison, we also display the corresponding results for the simplified BART algorithm that we analyzed theoretically. We observe that the mixing time increases with n for both BART and simplified BART. Furthermore, even when running the chain with more burn-in samples than recommended, we observe that BART fails to mix on all of the original datasets. This failure to mix for large n is also visually demonstrated for BART and simplified BART respectively in Figure 2.3 and Appendix A.2.1, through RMSE density plots. As n increases the RMSE densities show visible differences. Observing such differences is extremely unlikely if these chains are comprised of samples from a distribution that concentrates around a fixed function. Furthermore, these plots illustrate a subtle point that mixing reflects the relative magnitude of the between-chain variation compared with the within-chain variation, and

[4] 1e-5 is used avoid a numeric error, this term does not affect the proposal probabilities.

[5] The Gelman-Rubin diagnostic is a statistic that uses the observation that multiple MCMC chains should realize similar values under convergence to give a measure of convergence for a collection of MCMC chains. Suppose we have L samples from J different MCMC chains, let x_{ij} denote the ith sample from the jth chain, $\bar{x}_j$ denote the mean sample from the jth chain, and $\bar{x}$ denote the overall mean sample. The Gelman-Rubin diagnostic is defined as the ratio $R = \frac{\frac{L-1}{L}W + \frac{1}{L}B}{W}$ where W denotes the within-chain variance: $W := \frac{1}{J}\sum_{j=1}^{J} \frac{1}{L-1}\sum_{i=1}^{L}(x_{ij} - \bar{x}_j)^2$ and B denotes the between-chain variance: $B := \frac{L}{J-1}\sum_{j=1}^{J}(\bar{x}_j - \bar{x})^2$.

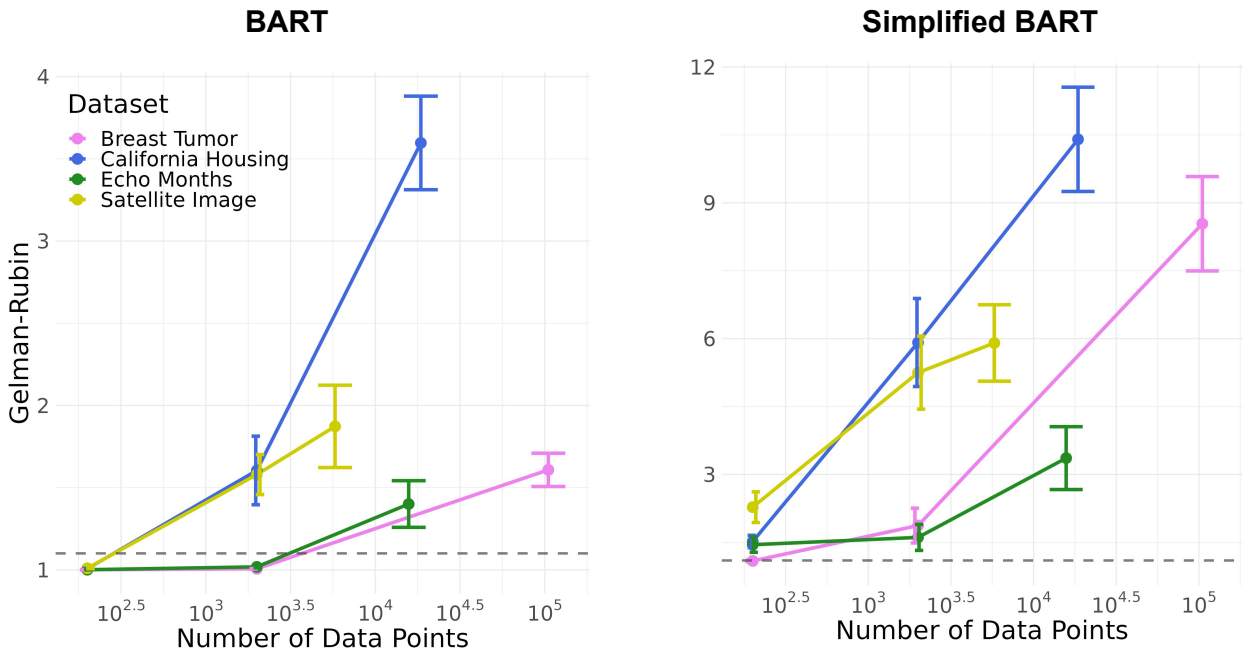

Figure 2.2: Mixing quality for BART and simplified BART decreases with the number of data points. Mixing quality is measured in terms of the Gelman-Rubin diagnostic applied to the test set RMSE using 8 chains. Error bars are calculated over 20 different Monte carlo runs using the same training data. The left column shows the results for BART, while the right for simplified BART . The horizontal dashed line represent the recommended mixing threshold of 1.1.

not its absolute magnitude. In particular, the between-chain variation for the Breast Tumor dataset is no more than 0.1% of the RMSE value, while the GR value is 1.74. Appendix A.2.2 visualizes the same information using the cusum path plots [140], which have also been used for MCMC diagnostics. We observe that the chains follow a smoother path as the number of data points increases, which is an indication of slow mixing (see Appendix A.2.2 for more details).

2.4.2 Experiment 2: Root split stuck on suboptimal feature

Additional set-up: In contrast to the previous experiment, we study Bayesian CART instead of BART, in order to understand to what extent the two additional moves, **Change** and **Swap**, can help avoid bottlenecks. For each of the 6000 iterations of each chain (including the burn-in iterations), we record the index of the feature on which the root node

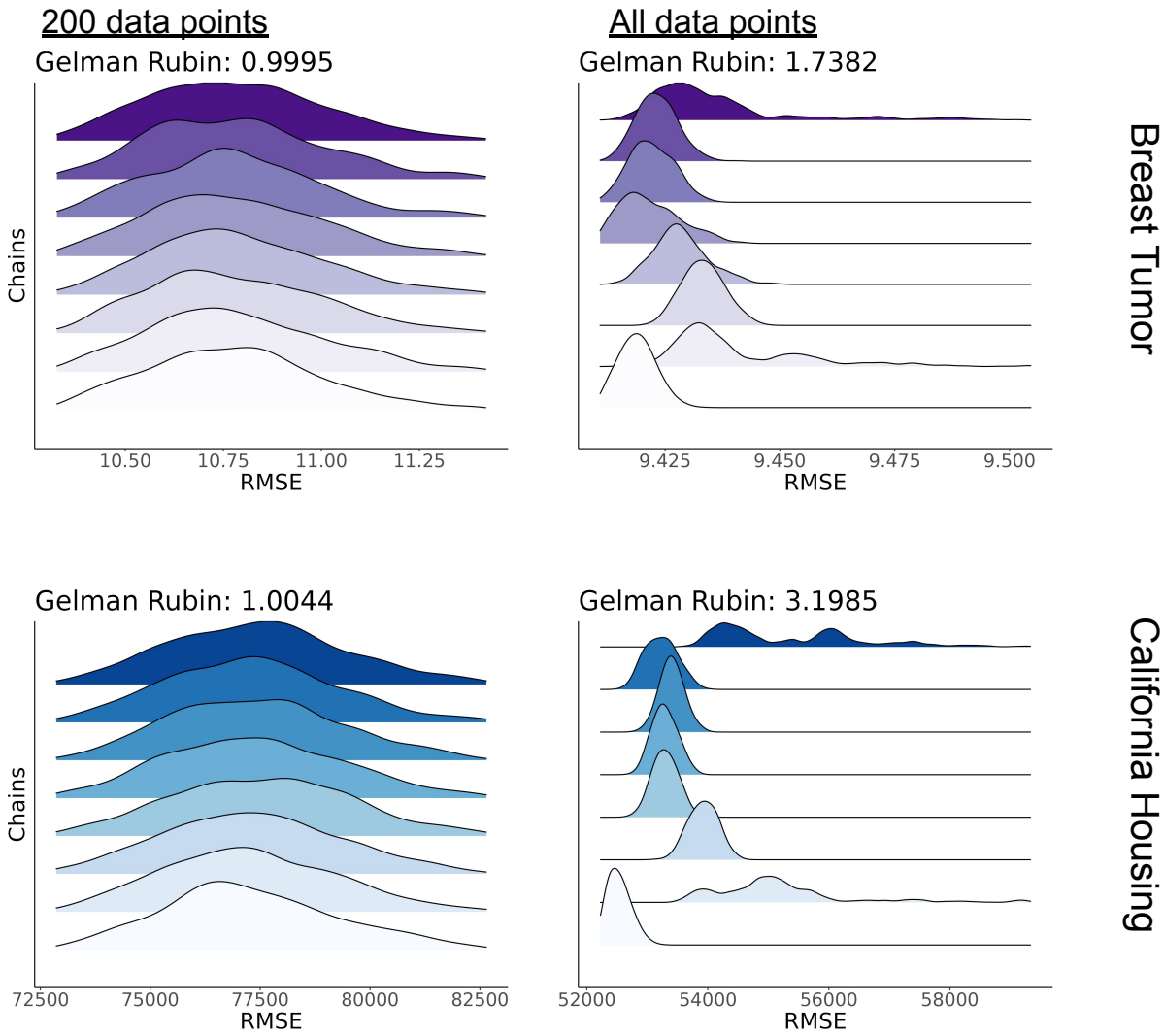

Figure 2.3: Kernel density plots of BART RMSE values across different chains and data sets and sample sizes. The number of data points used increases from the left to right column. Different rows correspond to different datasets.

splits.

Results: The results suggest that the root node changes in only a tiny fraction of the iterations, which decreases as n increases. Specifically, when each full dataset is used, the root split changes in less than 0.2% of the samples on average across 160 chains. Appendix A.2.3 plots the number of iterations during which the feature on which the root node splits changes (via a prune, change, or swap move) against the number of data points, and demonstrates

that the probability of such change decreases with n. Therefore, even with the full set of moves, changing the root split remains a bottleneck for the chain.

In Figure 2.4, we summarize the recorded split indices by counting the number of iterations on which the root node splits on each feature. In each subplot, we display the results separately for each of the 8 independent chains. We observe that for full datasets, an overwhelming majority of the root splits occur on the same feature, and furthermore, this feature is different for different chains. Similar analysis for simplified BART is deferred to Appendix A.2.4.

This further supports our claim about the bottleneck, while demonstrating the instability of the algorithm at the level of tree structures. Moreover, it implies that the chain can get stuck with a root split that is suboptimal.

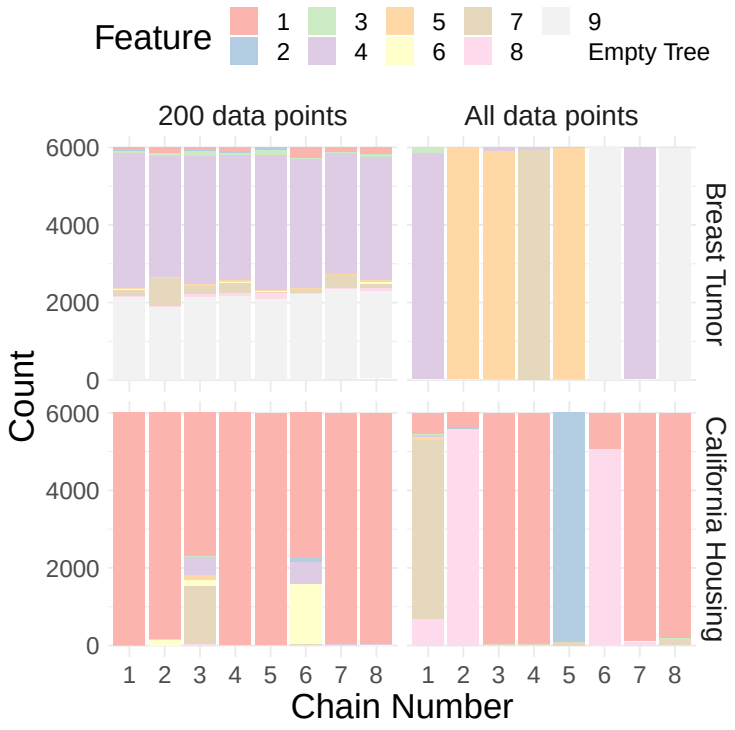

Figure 2.4: Number of iterations on which the root node splits on each feature for different chains on two datasets. When n is large (right column), almost all MCMC samples for a single chain of Bayesian CART have a root split on the same feature, whose index varies across chains. The different colors correspond to the different features, and empty tree refers to a single leaf (no splits).

2.5 Discussion

Poor mixing of the BART and Bayesian CART MCMC algorithms has been observed since these algorithms were introduced. In this chapter we provide the first theoretical analysis that proves a mixing time lower bound scaling with the number of training data points for a simplified version of the BART MCMC algorithm. Our analysis of simplified BART can partially explain why this occurs: The first split in the tree forms a bottleneck for the mixing of the chain, and the bottleneck becomes more difficult to traverse as the number of training data points increases. While we do not provide theoretical results for BART, our simulations show that the trend of mixing degradation with the number of data points suggested by our theory also holds for BART. We also learn from our simulations that Bayesian CART also encounters a bottleneck when splitting on the first feature; the restricted move set of simplified BART seems not solely responsible for this bottleneck.

The bottleneck formed by the first split in the tree helps explain one cause of the poor mixing of Bayesian CART and possibly BART. This split, which may be sub-optimal, forms a bottleneck for the chain and makes it difficult for it to transition to a tree with a differing root split. Furthermore, both Bayesian CART and BART choose the proposed feature and threshold uniformly at random, and are not guaranteed to make an optimal split initially.

Future Directions: We have made four simplifying assumptions to make the theoretical analysis of BART tractable. Most significantly, we studied a single tree instead of a sum, and only allowed **Grow** and **Prune** moves for the MCMC proposal (thereby excluding the **Change** and **Swap** moves). If either **Change** or **Swap** moves are allowed, the conductance computations would become more complicated and we may not be able to use the same bottleneck set used in the proof of Theorem 1. In future work, we aim to expand our result to hold for a single tree with an unrestricted set of MCMC moves. Analyzing the behavior of a sum of trees would require developing a new set of analytical tools, and appears prima facie to be more difficult. Despite these challenges, a result in this vein would be insightful as it would offer a better understanding of the BART algorithm as used in practice.

The remaining simplifying assumptions are mostly technical, and do not greatly affect the generality of our results. Our assumption of discrete features is relatively benign as a continuous feature can be approximated by its discretization on a sufficiently fine grid. Relaxing this assumption would entangle the prior and the likelihood, since in practice for continuous features, splits are selected from a grid of values dependent on the data. Lastly, our assumption that the variance of the Gaussian prior placed on the leaf parameters, σ^2, is fixed is not divergent from practice because it converges to a fixed value within a small fraction of the iterations that the algorithm runs for [23]. Allowing σ^2 to be random, as is done in practice, would complicate the form of the integrated likelihood and make an equivalent version of Lemma 4 more difficult to prove.

We believe our work should be viewed as only a first step towards rigorously understanding the mixing properties of BART and suggesting principled improvements to this class of MCMC algorithms. Our work suggests several natural ways to improve the mixing of

BART in practice: The proposal distribution for selecting splits should be data-adaptive rather than uniform at random, the chain should be initialized at an intelligent guess for the possible trees, and the number of MCMC chains run for BART should increase with the number of training data points.

Chapter 3

Mitigating Over-Exploration in Bayesian Optimization for Drug Design

3.1 Introduction

Many important tasks in scientific fields, such as small molecule discovery and protein engineering, can be framed as discrete black-box optimization problems.

Optimization is particularly effective when the goal is to improve a specific property, such as increasing the binding affinity of a molecule to a given target, or reducing it's toxicity.

Latent space optimization (LSO) was recently developed to enhance the sample efficiency of discrete optimization algorithms, such as genetic algorithms, in the black-box setting [44]. LSO transfers the optimization problem to the domain of the latent space of a Varitional auto-encoder (VAE), which can be efficiently explored using continuous optimization methods. However, when the original discrete space is structured, ensuring that LSO solutions respect the structure of the original space remains a challenge. To illustrate this issue, we provide some examples.

Example 5 (Arithmetic expressions)**.** *An expression built up using numbers, arithmetic operators and parentheses is called an arithmetic expression. However, not every sequence of the above elements correspond to a valid expression. For instance, the expression "*`sin(x) + x`*" is a valid expression while "*`sin(xxx`*" is not.*

Example 6 (Simplified molecular-input line-entry system (SMILES))**.** *SMILES provides a syntax to describe molecules using short ASCII strings. Atoms are represented by letters (e.g., water:"O"), bonds are represented by symbols (e.g., triple: "#", double: "=", . . .), branches are represented in parentheses and cyclic structures are represented by inserting numbers at the beginning and the end. Like the arithmetic expressions case, not every combination of the*

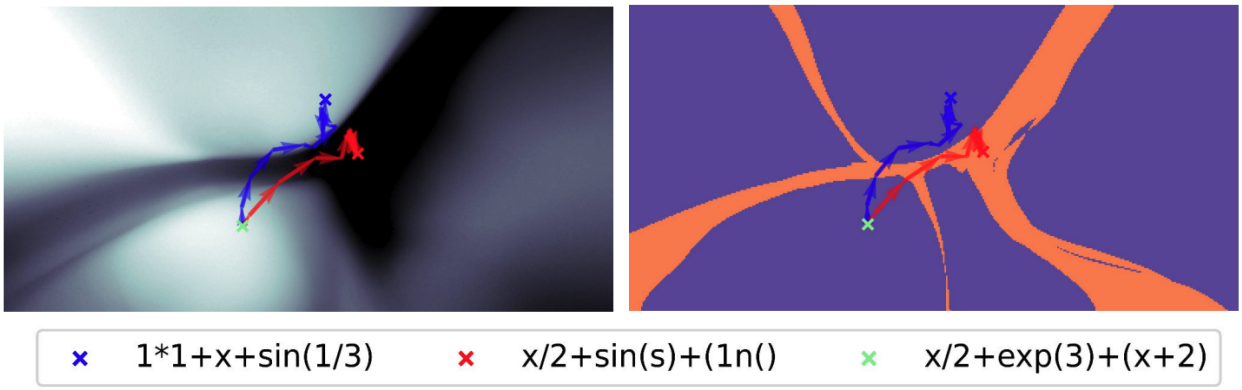

Figure 3.1: Incorporating LES promotes valid solutions. We consider the task of approximating the expression `1/3 + x + sin(x * x)`, using LSO. Optimization trajectories with (blue) and without (red) LES constraint in the latent space of a VAE are projected onto a two-dimensional subspace that contains the starting point and the end-points obtained after 10 gradient ascent steps. In the left panel, we show the LES score for latent vectors on the two-dimensional subspace, with darker shades corresponding to lower LES. In the right panel, we show the validity of the decoder outputs for each latent vector, with orange denoting invalid generations. High LES values correlate with valid areas, and incorporating LES in LSO produce an expression that adheres to the grammatical rules of Example 5.

elements described above corresponds to a valid molecule. For example, while "`C1CCCCC1`" is valid, both "`C1CCCCC2`" and "`C1CCCCC)`" are not.

Example 7 (Quality filters for molecules)**.** *Chemists seek molecules that not only optimize desired chemical properties but are also stable and easy to synthesize. This has led to the development of rules such as Lipinski's Rule of Five (RO5 [80]), which helps determine if the bioavailability (i.e., the proportion of a drug or other substance that enters the circulation when introduced into the body) of a given compound meets a certain threshold. For example, RO5 suggests that poor absorption is more likely when the octanol-water partition coefficient (logP) exceeds 5. Similarly, the Pan Assay Interference Compounds [5] filter helps in identifying false positives in assay screenings. The* `rd_filters` *[126] package has curated many such rules and is considered a "high precision, low recall surrogate measure" [18]. Following [90] we consider a sample valid if it passes the* `rd_filters` *filters* [1]

Numerous directions have been explored to overcome the challenge of providing valid solutions, including specialized VAE architectures [70, 61] or robust representations for discrete data [69]. Additionally, constrained objectives can be formulated under the assumption that one has access to a function which quantifies the validity of any point in the latent space

[1] We use the default Inpharmatica rule set comprised of 91 alerts

[48]. However, in many realistic scenarios, such as Example 7, these solutions may not be directly applicable, as the structure of the sequence space may not be sufficiently well understood. To address this, Notin et al. [90] proposed using an estimator of the uncertainty of the decoder, based on the variational approximation to a posterior distribution over the VAE parameters, encouraging LSO to respect the sequence space structure (details are provided Appendix B.4). Although this approach proved effective, the non-differentiable nature of the uncertainty score required its integration into LSO through heuristic approaches. Additionally, the computation of the uncertainty score is not exact (i.e., it relies on variational approximation and Monte Carlo sampling) and requires significant amount of time to compute. Therefore, there is a need for robust methods that work across different VAEs and sequence space structures, and can be easily integrated into existing LSO pipelines.

To achieve this goal, we develop LES, a score that can be used as a constraint in LSO optimization to increase the number of solutions that respect a given structure. The distinctive characteristics of LES are differentiability and robustness that allow its easy integration into existing LSO pipelines. Specifically, our contributions are as follows:

- We introduce LES, a score that achieves higher values in regions of the latent space closer to the training data. Our results demonstrate that LES is highly effective at identifying regions that preserve the structure of the sequence space. Although LES' computation scales cubically with the latent dimension, it is up to 80% faster than the current state-of-the-art for identifying out-of-distribution data points in the latent space of generative models for discrete sequences (Tables 3.1 and B.19).
- We develop a numerically stable optimization procedure to incorporate LES as a constraint in LSO.
- We evaluate LES-constrained LSO across thirty optimization tasks, including twenty-two VAEs and five benchmark problems, demonstrating its robustness in generating valid solutions and achieving high objective values. Specifically, in 19 out of the 30 LSO experiments, our method either finds the best solution on average or achieves a solution within 1 standard deviation of the best solution across 10 independent runs. This outperforms the six alternative methods we considered by 19% (Tables B.17 and B.18).

3.2 Background: Latent Space Optimization

LSO is a method for solving black-box optimization problems in discrete and structured spaces, such as the space of valid arithmetic expressions. Formally, let $\mathcal{V} \subset \mathbb{R}^{\mathrm{L}\times\mathrm{D}}$ be a discrete and structured space, represented as a sequence of L one-hot vectors of dimension D. We represent sequences of length L of categorical variables with D categories. L is set as the maximum sequence length that we are optimizing for, and one of the D categories is used as an "empty" category (which enables generating sequences shorter than the maximal

length). For instance, in the case of valid arithmetic expressions, $\mathcal{V}$ would be the set of sequences that define such expressions. Let $\mathcal{M} : \mathcal{V} \to \mathbb{R}$ be the objective function. LSO solves,

$$\arg\max_{x \in \mathcal{V}} \mathcal{M}(x). \tag{3.1}$$

In this setting, we assume that evaluations of the objective function ($\mathcal{M}$) are expensive to conduct. For example, the objective may be the binding affinity of a compound to a given protein, measured through a wet lab experiment.

A popular approach to solve Equation (3.1) is Bayesian Optimization (BO), which utilizes first order optimization of a surrogate model for $\mathcal{M}$. However, since the space is discrete, first order optimization cannot be directly applied.

In an attempt to make BO applicable for solving Equation (3.1), [44] proposed to transfer the optimization problem into that over a domain of the latent space of a deep generative model and subsequently perform BO in this space. The main idea is to (1) learn a continuous representation of the discrete objects (e.g., using a VAE) and (2) perform BO in the latent space while decoding the solution at each step. Formally, given a pre-trained encoder ($\mathbf{E}_\theta$) and decoder ($\mathbf{G}_\theta$) the initial labelled dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ is first encoded into the latent space $\mathcal{D}^{\boldsymbol{z}} = \{\boldsymbol{z}_i = \mathbf{E}_\theta(\boldsymbol{x}_i), y_i\}_{i=1}^n$. Using the encoded dataset, a BO procedure is conducted, which we describe in Algorithm 1. Most commonly, a Gaussian process is used as the surrogate model for $\mathcal{M}$, and the acquisition function is the expected improvement [39], defined as:

$$\mathcal{A}_{\hat{f}}(\boldsymbol{z}) = \mathbb{E}_{\hat{f}} \max(\hat{f}(\boldsymbol{z}) - \max_i y_i, 0) \tag{3.2}$$

where the expectation is with respect to the distribution of the function $\hat{f}$, conditioned on $\mathcal{D}^{\boldsymbol{z}}$.

Algorithm 1 Latent Space Optimization

for $t = 1$ **to** T **do**

1. Fit a surrogate model $\hat{f}$ to the encoded dataset, $\mathcal{D}^{\boldsymbol{z}}$
2. Generate a new batch of query points by optimizing a chosen acquisition function ($\mathcal{A}$)

$$\boldsymbol{z}^{(\text{new})} = \arg\max_{\boldsymbol{z}} \mathcal{A}_{\hat{f}}(\boldsymbol{z}) \tag{3.3}$$

3. Decode $\boldsymbol{x}^{(\text{new})} = \mathbf{G}_\theta(\boldsymbol{z}^{(\text{new})})$, evaluate the corresponding true objective values ($y^{(\text{new})} = \mathcal{M}(\boldsymbol{x}^{(\text{new})})$) and update $\mathcal{D}^{\boldsymbol{z}}$ with $(\boldsymbol{z}^{(\text{new})}, y^{(\text{new})})$.

Over-exploration in LSO Multiple studies [90, 70] have found that unconstrained latent space optimization (LSO) often yields solutions that disregard the aforementioned structures. For example, when searching for arithmetic expressions, invalid equations like "$ssin(xxx$" frequently occur. Similarly, many solutions in molecule searches fail to pass basic quality filters (Example 7), limiting their practical utility [84].

While acquisition functions such as expected improvement (Equation (3.2)) are designed to balance exploration and exploitation based on the estimated uncertainty from the Gaussian process model for $\mathcal{M}$. The frequent generation of invalid solutions during acquisition optimization, which implies that the estimated uncertainty can be problematic in this setting, underscores the need for additional regularization [120], which we aim to address.

To mitigate over-exploration, we propose adding a penalty to Equation (3.3). The penalty uses a new score, giving higher values over the latent space valid set, defined as:

$$\{\boldsymbol{z}; \mathbf{G}_\theta(\boldsymbol{z}) \in \mathcal{V}\}, \tag{3.4}$$

where $\mathbf{G}_\theta : \mathcal{Z} \to \mathbb{R}^{\mathrm{L}\times\mathrm{D}}$ is the decoder network, and $\mathcal{V} \subset \mathbb{R}^{\mathrm{L}\times\mathrm{D}}$ is the set of valid sequences.

The derivation of our score leverages the Continuous Piecewise Affine (CPA) representation of neural networks, which we briefly review below.

Deep generative networks as CPA Following Humayun et al. [60, 59] and Balestriero et al. [7, 8], we consider the representation of Deep Generative Networks (DGNs) as Continues Piecewise Affine (CPA) Splines operators. Let f_θ be any neural network with affine layers and piecewise affine activations then it holds that

$$f_\theta(\boldsymbol{z}) = \sum_{\omega\in\Omega} (\boldsymbol{A}_\omega \boldsymbol{z} + \boldsymbol{b}_\omega)\, 1_{\{\boldsymbol{z}\in\omega\}}, \tag{3.5}$$

where Ω is the input space partition induced by f_θ, ω is a particular region and the parameters $\boldsymbol{A}_\omega$ and $\boldsymbol{b}_\omega$ defines the affine transformation depending on ω.

In cases where the neural network f_θ is not composed solely of piecewise affine layers and activations, we leverage the result from Daubechies et al. [32] to assert that Equation (3.5) either exactly represents f_θ or provides a sufficiently accurate approximation for our practical purposes [60]. We therefore argue that all the decoder neural networks included in our study (i.e., GRU, LSTM, and Transformers) can be approximated with high accuracy as continuous piecewise affine (CPA) functions.

3.3 A Latent Exploration Score to Reduce Over-Exploration in LSO

In this section, we introduce Latent Exploration Score (LES), our new score to reduce over-exploration in LSO. We begin by motivating LES and proceed to formally derive it in Section 3.3.1. In Section 3.3.2, we provide empirical evidence that LES gives higher values in the latent space valid set. The use of LES to regularize LSO is left for Section 3.4.

Motivation Our goal is to develop a meaningful constraint for optimizing the acquisition function within a latent space of a given VAE. Specifically, we aim to construct a constraint that is a continuous function of $\boldsymbol{z}$ with higher values, indicating that it is more likely that $\boldsymbol{z}$ resides within valid regions of the latent space (Equation (3.4)).

Such a score should be higher in regions near training data points, assuming most of VAE training data is valid. To achieve this, we treat the latent space of the VAE as a probability space, i.e. $\boldsymbol{z} \sim p_{\boldsymbol{z}}$, for some prior distribution p (for example standard Gaussian). The prior should reflect our best guess for the distribution of the observed data in the latent space. Solutions are mapped back to sequences by the decoder through a deterministic (we do not consider $\boldsymbol{x}$ to follow a conditional distribution given $\boldsymbol{z}$) transformation of the latent vectors. Therefore, any distribution on the latent space defines a distribution over the space of sequences. Our score uses the density function of the push-forward measure of $\boldsymbol{x} = \mathbf{G}_\theta(\boldsymbol{z})$, which we call the sequence density. Consequentially, our score depends only on the decoder network, not the encoder, and can potentially be applied to other generative models like GANs or diffusion models.

Why use the sequence density function? We argue that for a well-trained decoder network, the density should be higher in areas of the sequence space close to the training data. To see why, consider a decoding model $\mathbf{G}_\theta$ trained on a dataset $\{(\boldsymbol{z}_i, \boldsymbol{x}_i)\}_{i=1}^n$. The average loss (L) at $\boldsymbol{z}$ is

$$\ell(\mathbf{G}_\theta(\boldsymbol{z})) = \mathbb{E}_{\boldsymbol{x}|\mathbf{E}_\theta(\boldsymbol{x})=\boldsymbol{z}} L(\mathbf{G}_\theta(\boldsymbol{z}), \boldsymbol{x}). \tag{3.6}$$

As the training process is designed to minimize the population loss: $\mathbb{E}\ell(\mathbf{G}_\theta(\boldsymbol{z}))$, if successful, we hypothesize that the distribution of $\mathbf{G}_\theta(\boldsymbol{z})$ puts higher weight in the areas where $\ell(\mathbf{G}_\theta(\boldsymbol{z}))$ is low. Since we expect most of the training data to be valid and to achieve low expected loss, the sequence density should put higher weight on the latent space valid set. In, Section 3.3.2 we provide an empirical validation for this hypothesis, for Examples 5 to 7. We highlight that this relationship between the valid set and the sequence density depends on how well the decoder fits the data.

3.3.1 Derivation of LES

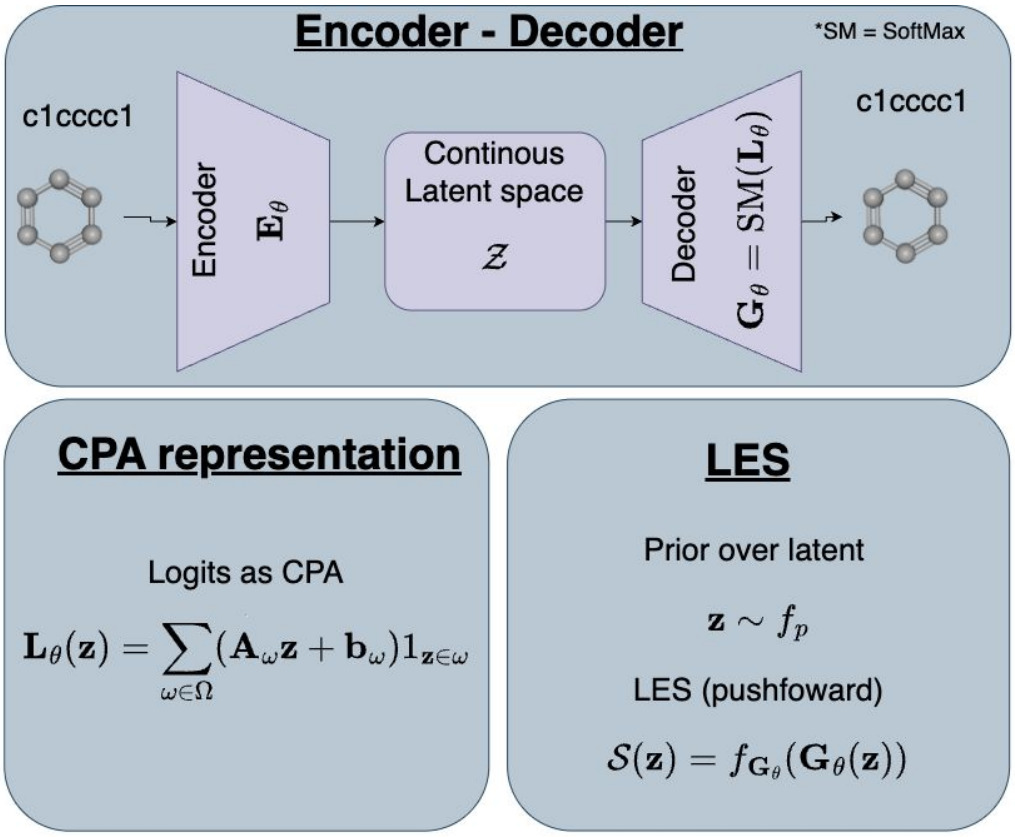

Figure 3.2: Derivation of LES. The decoder network ($\mathbf{G}_\theta$), which maps from the latent space to the output space, is assumed to be the composition of a softmax operation over a continuous piecewise affine (CPA) spline operator. LES is the density of a random variable ($\boldsymbol{z}$) in the latent space, under the decoder transformation. Calculating LES only requires a pre-trained decoder.

Analytical formula for LES DGNs for discrete sequences typically output a matrix of logits, transformed into normalized scores by the softmax function:

$$\mathbf{G}_\theta(\boldsymbol{z}) = \mathrm{Softmax}(\mathbf{L}_\theta(\boldsymbol{z})). \tag{3.7}$$

$\mathbf{L}_\theta(\boldsymbol{z})$ is a D $\times$ L logits matrix (D - vocabulary size, L - sequence length) and Softmax is the softmax operation applied to every column of $\mathbf{L}_\theta(\boldsymbol{z})$. In order to avoid a violation of the assumption that $\mathbf{G}_\theta$ is bijective, we extend the function's output to include the normalizing constant for each column. We find that parametrizing the output to include the inverse of

the normalizing constant (Equation (3.8)), helps in avoiding numerical instabilities that are caused by the constant being potentially very large. With this formulation, we can now derive the sequence density function.

Theorem 8 (DGN sequence density)**.** *Let*

$$\mathbf{G}_\theta(\boldsymbol{z}) = \left(\boldsymbol{p}_{\boldsymbol{z}}^{(1)}, (c_{\boldsymbol{z}}^{(1)})^{-1}, \ldots, \boldsymbol{p}_{\boldsymbol{z}}^{(L)}, (c_{\boldsymbol{z}}^{(L)})^{-1}\right) = \boldsymbol{x}_{\boldsymbol{z}} \tag{3.8}$$

where $\boldsymbol{p}_{\boldsymbol{z}}^{(i)} = Softmax(\mathbf{L}_\theta(\boldsymbol{z}))_{.i}$ *and* $c_{\boldsymbol{z}}^{(i)} = \sum_{j=1}^{D} e^{\mathbf{L}_\theta(\boldsymbol{z})_{ji}}$*. Assume that* $\mathbf{L}_\theta$ *is bijective and can be expressed as a CPA (Equation (3.5)), and that* $\boldsymbol{z} \sim p_{\boldsymbol{z}}$*, then the density function of* $\mathbf{G}_\theta(\boldsymbol{z})$ *is given by:*

$$f_p(\boldsymbol{z})\sqrt{\det\left(\sum_{i=1}^{\mathrm{L}} (\boldsymbol{A}_i^\dagger)^T (\boldsymbol{B}_i)^T \boldsymbol{B}_i \boldsymbol{A}_i^\dagger\right)} \tag{3.9}$$

for

$$\boldsymbol{B}_i = \left(diag\left(\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_1}, \ldots, \frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_{\mathrm{D}}}\right), -\mathbf{1}\frac{1}{c_{\boldsymbol{z}}^{(i)}}\right)^T \tag{3.10}$$

$$\boldsymbol{A}_i^\dagger = \left(\boldsymbol{A}_\omega^{(1)}, \ldots, \boldsymbol{A}_\omega^{(L)}\right)^\dagger_{(i\cdot \mathrm{D}):(i+1\cdot \mathrm{D}).}, \tag{3.11}$$

where $\left(\boldsymbol{A}_\omega^{(1)}, \ldots, \boldsymbol{A}_\omega^{(L)}\right)^\dagger$ *is the Moore–Penrose inverse of* $\left(\boldsymbol{A}_\omega^{(1)}, \ldots, \boldsymbol{A}_\omega^{(L)}\right)$*, and* f_p *is the density function of* $p_{\boldsymbol{z}}$*.*

The proof is provided in Appendix B.1. We define LES to be the logarithm of the determinant term,

$$\mathcal{S}(\boldsymbol{z}) = \log\left(\sqrt{\det\left(\sum_{i=1}^{\mathrm{L}} (\boldsymbol{A}_i^\dagger)^T (\boldsymbol{B}_i)^T \boldsymbol{B}_i \boldsymbol{A}_i^\dagger\right)}\right), \tag{3.12}$$

as the contribution of the prior is negligible in magnitude in all the decoders we study.

Remark 9. *LES can be calculated directly without assuming the decoder logits follow a CPA function [11]. However, using the expressions derived in Equation (3.12) has two computational benefits for calculating the derivative of LES. First, using Jacobi's formula and observing that* $\sum_{i=1}^{\mathrm{L}} (\boldsymbol{A}_i^\dagger)^T (\boldsymbol{B}_i)^T \boldsymbol{B}_i \boldsymbol{A}_i^\dagger$ *is a quadratic formula of the softmax probabilities, we can calculate the derivative of LES in closed form. Second, by the CPA assumption, the matrices* $\boldsymbol{A}_\omega^{(i)}$ *are a constant function of* $\boldsymbol{z}$ *and therefore* $\frac{\partial \boldsymbol{A}_\omega^{(i)}}{\partial \boldsymbol{z}} = 0$*. As a result, we avoid the need to calculate the hessian of the decoder when taking the derivative of LES.*

Limitations of Theorem 8 Our derivation relies on the decoder logits being (i) a CPA operator and (ii) bijective between the latent space and the generated manifold in the ambient space. We argue that (i) is not a restrictive assumption, as approximation theory has already demonstrated that any continuous model can be approximated by a CPA network. Therefore, one always recovers Theorem 8 even when using non-CPA models. Specifically, [32] show that ReLU networks (which are CPA spline functions) can approximate refinable functions—a class of non-smooth functions—in addition to known results on other classes of functions (e.g., Weierstrass, polynomials, and analytic functions).

On the other hand, (ii) is a stronger assumption that practitioners should be mindful of, as it would invalidate LES as a meaningful metric for comparing different samples. For (ii) to be violated, i.e., for Equation (B.14) to be incorrect, the Lebesgue measure of the set $C_{\mathbf{G}_\theta} = \{\boldsymbol{z} |\ \exists \boldsymbol{z}^*; \mathbf{G}_\theta(\boldsymbol{z}) = \mathbf{G}_\theta(\boldsymbol{z}^*)\}$ must be larger than 0 (see Lemma 11 for proof). This would suggest some degeneracy in the decoder function, where large regions of the latent space map to the same output, resulting in a zero gradient of the decoder with respect to its input. However, we believe this is rare in practice. In our experiments, where we compute the gradient of the decoder with respect to the input to calculate LES, we did not encounter instances where the gradient was zero.

Although we do not formally validate (ii) (the bijectivity assumption), we argue through our empirical analysis in Table B.19, conducted across 22 VAEs (including pre-trained models), that (ii) may hold or, at the very least, serve as a reasonable approximation for real-world VAEs.

Lastly, it is crucial to highlight that the ability of LES to detect out-of-distribution data is closely tied to the decoder's capacity to accurately model the data. Although Theorem 8 holds for poorly trained decoders under the given conditions, we advise against relying on LES in such cases.

Computing LES LES is a function of the matrices $\boldsymbol{B}_i$ and $\boldsymbol{A}_\omega$. The matrices $\boldsymbol{B}_i$ are a function of the logits and can be calculated using a single forward run of $\mathbf{G}_\theta$. The matrix $\boldsymbol{A}_\omega$ is equal to the derivative of $\mathbf{L}_\theta$ at $\boldsymbol{z}$, and can therefore can be obtained using automatic differentiation ([99]). To avoid the (pseudo) inversion of the matrix $\boldsymbol{A}_\omega$ we can exploit the inverse function theorem which states that $J\mathbf{G}_\theta^{-1} = (J\mathbf{G}_\theta)^{-1}$ and we can take $\mathcal{S}(\boldsymbol{z}) = -0.5 \log\det((A_\omega \frac{\partial \mathbf{G}_\theta(\mathbf{L}_\theta)}{\partial \mathbf{L}_\theta})(A_\omega \frac{\partial \mathbf{G}_\theta(\mathbf{L}_\theta)}{\partial \mathbf{L}_\theta})^T)$, where $\frac{\partial \mathbf{G}_\theta(\mathbf{L}_\theta)}{\partial \mathbf{L}_\theta}$ (derivative of the softmax w.r.t the logits) admits a simple closed form solution. Ideally, LES can be computed by performing all of the above calculations in parallel using a single forward call to the $\mathbf{G}_\theta$ network. In addition, the computation of the determinant is done via SVD on a square matrix whose dimension is the latent dimension of the decoder (d) with complexity of $\mathcal{O}(d^3)$.

In Table 3.1 we provide the wall clock times for calculating LES for a batch of 20 latent vectors across all architectures and datasets studies in this work. LES is compared with the Bayesian uncertainty score proposed by Notin et al. [90] (with the default configuration: 10 sampled models and 40 sampled outcomes), which was previously used to regularize LSO.

We also compare with a *Likelihood* score:

$$\ell(\boldsymbol{z}) = \max_{\boldsymbol{x}} p_{\mathbf{G}_\theta}(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Z} = \boldsymbol{z}) \tag{3.13}$$

where $p_{\mathbf{G}_\theta}(\boldsymbol{z})$ is the distribution defined by the decoder softmax probabilities, which reflects the likelihood of the most likely $\boldsymbol{x}$ conditioned on the latent vector $\boldsymbol{z}$.

For 9 out of the 10 models, particularly when the latent dimension is larger (e.g., SMILES and SELFIES), LES is computed faster than the Uncertainty score, achieving reductions of up to 85% in some cases. As the Likelihood score requires only a single forward pass of the decoder, it offers a more computationally efficient alternative, which comes with some performance trade-offs (Sections 3.3.2 and 3.4).

Table 3.1: Wall clock times in seconds (lower is better) for calculating LES, the Bayesian uncertainty and the Likelihood scores for a sample of 20 latent vectors on a single A100 GPU.

Dataset	Arch. (dim.)	LES	Uncertainty	Likelihood
Expressions	GRU (25)	0.730	0.823	0.025
	LSTM (25)	0.164	0.857	0.049
	Transformer (25)	0.526	0.481	0.029
SMILES	GRU (56)	0.663	3.157	0.103
	LSTM (56)	0.696	3.990	0.123
	Transformer (56)	0.581	0.726	0.185
SELFIES	GRU (75)	0.498	1.925	0.064
	LSTM (75)	0.525	2.442	0.071
	Transformer (75)	0.451	0.583	0.085
	Transformer (256)	7.422	42.882	0.410
Average	–	1.226	5.787	0.114

3.3.2 Validating the Relationship Between LES and Valid Generation

To assess LES's ability to identify valid regions (as defined in Examples 5 to 7), we sample data points in the latent space using the twenty-two VAEs studied in Section 3.4. Specifically, we sample 500 data points from three distributions: train, prior ($\mathcal{N}(\mathbf{0}, \boldsymbol{I})$), and out-of-distribution ($\mathcal{N}(\mathbf{0}, \boldsymbol{I} \cdot 5)$). We decode each data point and determine if the decoded sequence is valid.

Identifying if a point in the latent space decodes into a valid sequence can be viewed as a classification problem, in which the different scores (i.e., LES or the Bayesian uncertainty score) provide (unnormalized) probabilities for a sequence being valid. We measure the performance of these scores using the AUROC metric. Besides LES, the Bayesian uncertainty

score and the Likelihood score, we add three additional baseline scores for comparison. The first is the density of a standard Gaussian (p), which is the distribution the latent vectors are regularized to follow during VAE training. The second is the polarity score (**Polarity**) [60], based only on the derivative of the decoder logits with respect to the latent vector, which shows the gains due to accounting for the softmax non-linearity in the derivation of LES (Theorem 8). We also consider the average distance to the closest three data points within a random sample of 1000 points from the training data in the latent space (**Train distances**).

The results are shown in Tables 3.2 and B.19. LES provides the best performance in 18 out of the 22 VAEs in this analysis, and in all cases provides a clear signal for identifying valid regions, as indicated by AUROC values that are at least 0.75. This is while being much faster to compute than the Uncertainty score (Table 3.1) and without the need to store a potentially large array of latent vectors.

3.4 LES-Constrained LSO

In Section 3.3.2 we showed that LES is a robust score that obtains higher values in the latent space valid set (Equation (3.4)). Furthermore, LES is differentiable, which means it can easily be used to constrain any optimization problem. Therefore, we propose adding an explicit constraint to Equation (3.3), encouraging the solution to achieve a high LES value. We modify Algorithm 1 by penalizing step (2):

$$\boldsymbol{z}^{\text{new}} = \arg\max_{\boldsymbol{z}} \mathcal{A}(\hat{f}(\boldsymbol{z})) + \lambda \mathcal{S}(\boldsymbol{z}). \tag{3.14}$$

3.4.1 Experimental Setup

VAE models To evaluate the effectiveness of LES as a regularization method for LSO, we trained twenty-two VAEs, focusing on varying the decoder architectures and the β parameter, which controls the trade-off between reconstruction loss and alignment with the prior (KL divergence term). All models use a convolutional encoder based on the architecture proposed by Kusner et al. [70], and were trained for 300 epochs using the Adam optimizer [66] with a learning rate of 1e-3 and batch size of 256.

The VAEs for the Expressions dataset, sourced from Kusner et al. [70], were trained on 80k data points with a latent dimension of 25. The SMILES VAEs were trained on the ZINC250k dataset, consisting of approximately 250k drug-like molecules in SMILES format. Following Kusner et al. [70] and Notin et al. [90], a latent dimension of 56 was used. For the SELFIES VAEs, we used a subset of approximately 200k molecules from the ZINC250k dataset that passed a set of quality filters [126], using the SELFIES representation [69], with a latent dimension of 75. Additionally, the pre-trained VAE by Maus et al. [84] had a latent dimension of 256.

Table 3.2: AUROC values (higher is better) for identifying valid data points within the latent space, across datasets and decoder architectures. Data points are sampled from the training data, the VAE prior (standard Gaussian), and out-of-distribution data (Gaussian with std of 5). LES achieves the best performance in most cases (18 out of 22) and on average.

Dataset	Arch.	β	LES	Prior	Uncertainty	Likelihood
SMILES	GRU	0.05	0.93	0.09	0.85	**0.94**
		0.1	**0.94**	0.12	0.84	**0.94**
		1.0	0.91	0.16	0.87	**0.92**
	LSTM	0.05	**0.99**	0.07	**0.99**	0.98
		0.1	0.89	0.21	0.89	**0.90**
		1.0	**0.97**	0.12	0.95	0.96
	Transformer	0.05	**0.93**	0.14	0.84	0.92
		0.1	**0.94**	0.14	0.87	0.93
		1.0	**0.97**	0.10	0.89	0.95
Expressions	GRU	0.05	**0.96**	0.38	**0.96**	0.88
		0.1	**0.94**	0.42	**0.94**	0.80
		1.0	**0.94**	0.57	**0.94**	0.89
	LSTM	0.05	**0.96**	0.38	0.90	**0.96**
		0.1	**0.95**	0.37	0.91	**0.95**
		1.0	**0.95**	0.56	0.91	0.91
	Transformer	0.05	**0.91**	0.43	0.86	0.90
		0.1	**0.91**	0.53	0.87	0.89
		1.0	0.86	0.70	**0.92**	**0.92**
SELFIES	Transformer	0.05	**1.0**	0.02	0.99	0.97
		0.1	**0.99**	0.03	0.96	0.98
		1.0	**0.95**	0.06	0.85	0.93
SELFIES ([84])	Transformer	–	**0.75**	0.69	0.33	0.70
	Average		**0.93**	0.29	0.88	0.91

LSO setup We begin by training a single-task Gaussian Process on an initial dataset. Each sample is mapped to a latent space and paired with its true objective value. For Expressions and SMILES VAEs, we use 500 data points. For SELFIES VAEs, we use 1500. Across all tasks, we generate 500 candidate solutions per problem, aligned with prior work [70, 90] and reflective of real-world wet-lab constraints [42]. Solutions are proposed in batches of 20.

For the SELFIES VAEs (both from Maus et al. [84] and those trained by us), we employ a deep kernel that reduces the latent space to 12 dimensions before fitting the Gaussian Process. To mitigate vanishing gradients, we use log expected improvement [2] as our acquisition

function, which is sequentially maximized.

Optimization tasks The Expressions dataset consists of arithmetic expressions that are functions of a single variable (e.g., $\sin(x)$, $1+x*x$). Our objective is to find an expression that approximates `1/3 + x + sin(x * x)`, as described by Kusner et al. [70]. The optimization target is defined as $\mathcal{M}(\boldsymbol{x}) = -\log(1 + \text{MSE}(\boldsymbol{x}))$, where $\text{MSE}(\boldsymbol{x})$ is the mean-squared error between the expression $\boldsymbol{x}$ and `1/3 + x + sin(x * x)`, evaluated over the range -10 to 10 using a grid of 1000 equally spaced points.

For the SMILES dataset, our goal is to maximize the octanol-water partition coefficient, which is calculated using the prediction model developed by Wildman et al. [130]. In the case of the SELFIES dataset, following Maus et al. [84], we focus on three objectives: Perindopril MPO, Ranolazine MPO, and Zaleplon MPO, all of which are part of the Guacamol benchmarks [18]. While the SELFIES syntax is 100% robust, we consider only solutions that pass a set of quality filters for evaluation (see Example 7 for more details).

LES-constrained LSO For ease of implementation and numerical stability, when applying LES regularization, we adopt a simple optimization procedure in which the acquisition function is optimized using 10 steps of normalized gradient ascent. This is because we empirically find that the norm of the derivative of the constraint (i.e., $\mathcal{S}(\boldsymbol{z})$) is typically much larger than the norm of the derivative of the acquisition function. As a result, using the gradient ascent update rule $\boldsymbol{z}^{(i+1)} = \partial\mathcal{A}(\hat{f}(\boldsymbol{z}^{(i)})) + \lambda\partial\mathcal{S}(\boldsymbol{z}^{(i)})$ leads to a numerically unstable optimization process. To address this issue, we propose the following update rule:

$$\boldsymbol{z}^{(i+1)} = \frac{\partial\mathcal{A}(\hat{f}(\boldsymbol{z}^{(i)}))}{\|\partial\mathcal{A}(\hat{f}(\boldsymbol{z}^{(i)}))\|_2} + \lambda\frac{\partial\mathcal{S}_\rho(\boldsymbol{z}^{(i)})}{\|\partial\mathcal{S}_\rho(\boldsymbol{z}^{(i)})\|_2}. \tag{3.15}$$

We set $\lambda = 0.05$ for Expressions, $\lambda = 0.1$ for our SELFIES models, and $\lambda = 0.5$ for SMILES and the pre-trained SELFIES-VAE. For Ranolazine MPO with pre-trained SELFIES-VAE, we use $\lambda = 0.1$ to prevent over-regularization, as regularized methods already yield a high percentage of valid solutions (Table B.21). We select these values because, without regularization, the SMILES and pre-trained SELFIES models tend to produce a lower percentage of valid solutions (Table B.21). Based on our findings, we suggest $\lambda = 0.5$ as a reasonable default value, while leaving the exploration of an optimal choice for future work.

Benchmark methods We compare our LES-constrained method (**LES**) with five alternative approaches for optimizing the acquisition function. First, we evaluate a non-regularized version of Equation (3.15) , where $\lambda = 0$ (**GA**), and a two regularization methods that uses the prior density (i.e., ℓ_2 regularization) and likelihood (Equation (3.13)) instead of LES (**p**, and **Likelihood** respectively), using similar λ values described above.

Additionally, we consider optimizing the acquisition function using the Limited-memory BFGS method within a symmetric hypercube centered at 0 (**L-BFGS**), which is the default approach in the `BoTorch` package [6]. Since recent state-of-the-art LSO pipelines [84, 71]

have utilized trust regions centered around the best observed value [36], we also compare with this approach (**TuRBO**). Lastly, we implement the Uncertainty Censoring (**UC**) method proposed by Notin et al. [90], which suggests early stopping of the optimization when the estimated uncertainty exceeds a certain threshold. For this threshold, we use the 99th percentile of the observed uncertainty values in the training data, as recommended by Notin et al. [90].

Hyperparameters We calibrate the step size, which affects **LES**, **UC**, **p**, **Likelihood**, and **GA**, to ensure our gradient ascent procedure (with $\lambda = 0$) improves the acquisition function values across different initializations. The same step size is applied to all models within the same dataset: Expressions = 0.8, SMILES = 0.003, SELFIES = 0.03, and SELFIES pre-trained = 0.3. The L-BFGS method has a single hyperparameter, the facet length, which is set to 5. For TuRBO, there are three primary hyperparameters: the initial length, which we set to 0.8, along with the success and failure tolerances, determining when to expand or shrink the trust region, set at 10 and 2, respectively. An ablation study is provided in Appendix C.3.

3.4.2 Results

Optimization results The experimental results, presenting the average across 10 independent LSO runs for the top 20 and best solutions found, are summarized in Tables B.17 and B.18 respectively. In both cases, LES achieves the average best performance most frequently (22 and 17 out of 30 times, respectively). Furthermore, LES outperforms other methods in both the average ranking across LSO tasks and the frequency with which it falls within one standard deviation of the best-performing method (Table 3.3). These findings demonstrate that using LES as a regularization technique generally enhances optimization performance.

Figures 3.3 and B.1 show the cumulative average top-20 and the best objective values during BO with the pre-trained SELFIES-VAE [84], respectiveley. Results align with [84] under our realistic evaluation budget. For the average of the top-20 solutions on the Ranolazine MPO and Zaleplon MPO tasks, LES outperforms TuRBO on valid solutions but underperforms overall, highlighting that LES effectively constrains the optimization.

Table B.21 shows the percentage of valid solutions found by each method across datasets and VAEs. LES improves upon the non-regularized version of gradient ascent by 7% on average and upon TuRBO and L-BFGS by 24% and 36% on average respectively. While UC achieves a 2% higher percentage of valid solutions on average, we show in Table B.20 that for the Expressions datasets, where UC excels, the number of valid produced by LES solutions can be increased by setting a higher λ value. However, this did not improve optimization performance.

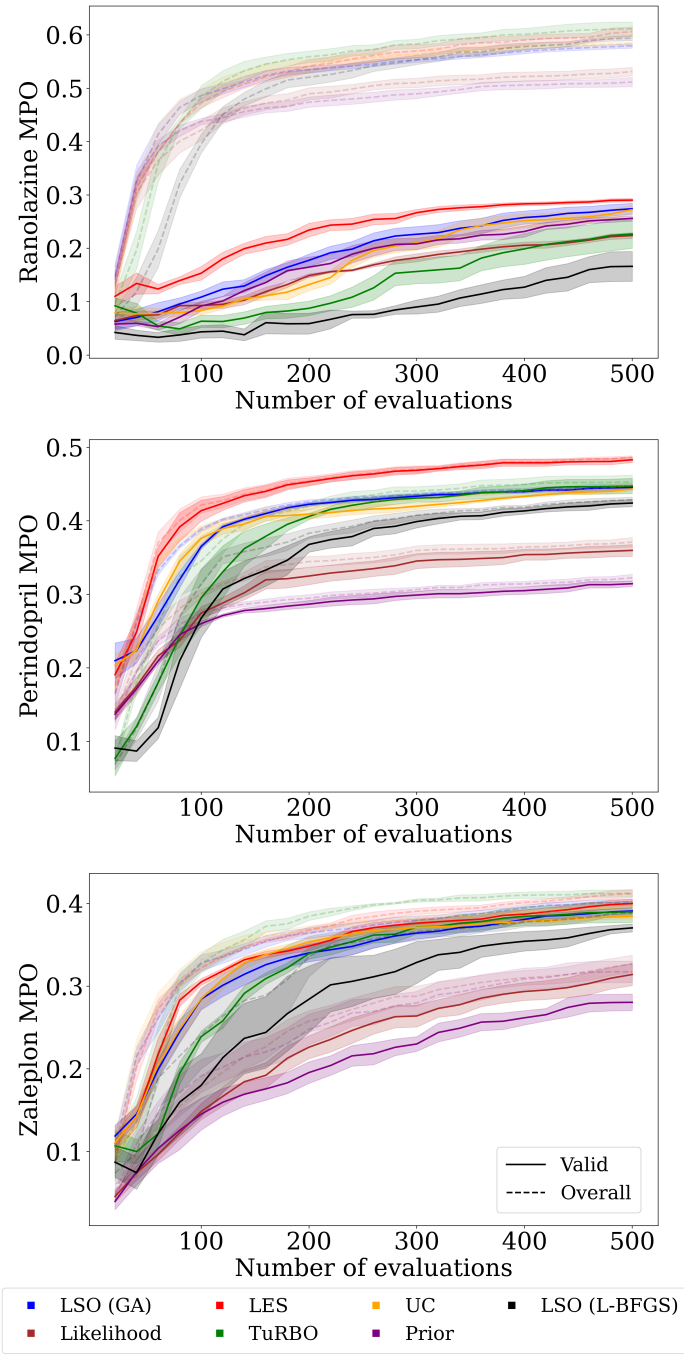

Figure 3.3: Cumulative objective for the top-20 solutions found during Bayesian optimization with the pre-trained SELFIES-VAE ([84]). Each method is shown in a distinct color. Solid lines represent solutions passing quality filters, while dashed lines include all evaluations. LES outperforms all baselines on Ranolazine MPO and Perindopril MPO, achieving competitive results on Zaleplon MPO.

Table 3.3: Summary metrics for 30 Bayesian Optimization experiments from Tables B.17 and B.18. We report the average rank (lower is better) and the count of times each method is within one standard deviation of the best. Results cover both the best solution and the top-20 average. LES outperforms alternatives on both metrics.

	Top 1		Top 20	
Method	**Avg Rank**	**# within std**	**Avg Rank**	**# within std**
LES	**2.15**	**19**	**1.9**	**22**
Likelihood	2.76	16	2.53	13
GA	3.15	8	2.87	7
Prior	3.86	10	3.8	5
UC	4.17	5	4.3	3
TuRBO	5.7	2	6.1	0
L-BFGS	6.17	1	6.4	0

3.5 Discussion

In this chapter we proposed LES to mitigate over-exploration in latent space optimization (LSO). LES is differentiable and fully parallelizable. Extensive evaluations demonstrate that incorporating LES as a penalty in LSO consistently enhances solution quality and objective outcomes. Moreover, LES outperforms alternative regularization techniques, and proves to be most robust across diverse datasets and varying definitions of validity. In addition, LES has only a single hyperparameter (the regularization strength), and we observe empirically that deploying LES can provide significant performance gains. We therefore believe LES offers a powerful approach for discovering more realistic solutions, when the criteria for realism are difficult to define or validate.

While LES is fully parallelizable, it requires the calculation of the derivative of the decoder as well as the determinant of the change-of-variables term, which can be computationally expensive. This step can become a bottleneck when the size of the output and the latent dimension are both large. It is left for future work to develop a fast approximation for this operation in order to enable the use of LES in applications involving large generative models.

Chapter 4

Stabilizing Protein Fitness Predictors via the PCS Framework

4.1 Introduction

Improving the ability of machine learning (ML) models to predict the effects of mutations on protein fitness is a central challenge in computational biology, with far-reaching implications for protein design, disease understanding, and beyond. In protein engineering, ML has shown promise in reducing the experimental cost associated with discovering high-fitness protein sequences—by prioritizing which sequences to test in costly wet-lab experiments [133].

However, successfully utilizing ML methods to guide wet lab experiments poses a few unique challenges. First, as the goal is to identify new protein sequences (typically mutations of a given reference sequence) with desired functional properties, these methods must be able to provide useful predictions for out-of-distribution (OOD) sequences. Second, uncertainty quantification (UQ) is essential for assessing the reliability of predictions, as no existing model reliably generalizes across the vast, combinatorial space of protein sequences.

Existing efforts to improve OOD generalization [119] and provide reliable UQ for protein engineering [47] primarily focus on the modeling stage of the data-science life cycle (DSLC [139]). For UQ, Bayesian methods offer uncertainty estimates via the posterior distribution [47, 93], while ensembling neural networks with random initializations captures variability due to randomness of the training procedure [50]. Additionally, frequentist approaches estimate predictive variance through probabilistic modeling [89, 47]. Conformal prediction techniques, which quantify uncertainty in the form of prediction intervals, have been used for model selection for protein design and function prediction [38, 15, 37]. While the use of pre-trained protein language models (PLMs) and ensembling has shown promising results for OOD generalization in antibody design [119], there remains a critical need to further strengthen these capabilities to ensure their utility across a broader range of protein engineering tasks [132].

The goal of this work is to highlight the importance of stability under *reasonable* (for the

purpose of prediction) perturbations to data processing choices, with a particular focus on protein engineering. Following the Predictability-Computability-Stability (PCS) framework for veridical data science [139, 138], we introduce a simple and broadly applicable procedure to improve UQ and OOD generalization by leveraging multiple reasonable representations of the same protein sequence. For example, embeddings derived from different pre-trained PLMs, or zero-shot evolutionary scores obtained by different methods. Our key contributions are:

- We identify a previously underappreciated but critical source of instability in protein fitness prediction: the choice of data representation (i.e., embeddings). Our findings reveal that the performance of state-of-the-art models can vary substantially depending on these choices (Section 4.4 and Appendix C.3).
- We propose a simple and intuitive two-step procedure (Figure 4.1 and Section 4.3) to leverage multiple reasonable data processing choices, guided by the PCS framework. Our procedure has two steps, (1) **Pred-Check** step to select and weight representations which are predictive for the fitness of interest, and (2) an **ensembling** step that fits a given base method on each representation that passes the Pred-Check, yielding a Stable and Pred-Checked (**SP**) fitness predictor.
- On the supervised ProteinGym substitution benchmark, SP predictors consistently outperform their base models—including Gaussian Processes (GPs), linear models, and CNNs—in both predictive accuracy and uncertainty estimation (Section 4.4.1). Among them, SP Kermut improves over the standard Kermut (the current state-of-the-art) by reducing MSE by up to 20% and increasing Spearman correlation by up to 10%, with the largest improvements observed under distribution shifts. For uncertainty, SP Kermut also achieves up to 70% stronger correlation between predicted uncertainty, and the true absolute errors compared to standard Kermut, that relies on a single representation.
- Finally, we show that our SP versions of Bayesian Ridge and Kermut methods improve performance in in-silico iterative protein design, leading to 6% improvement in recovering the highest fitness sequences over the prior best performing base method Kermut (Section 4.4.3).

4.2 Background and related works

Supervised fitness prediction We represent a protein sequence as $p = (a_1, \ldots, a_{\mathrm{L}^{(p)}})$, where $\mathrm{L}^{(p)}$ is the sequence length and each a_i is one of the 20 canonical amino acids. We define a *mutated sequence* relative to p as a sequence of the same length that differs from p at one or more positions, and is denoted by $m^{(p)}$. We denote the set of all such mutated sequences as $\mathbb{M}_p$. The goal of supervised fitness prediction is to learn a predictor $\hat{f}^{(p)} : \mathbb{M}_p \to \mathbb{R}$ that maps

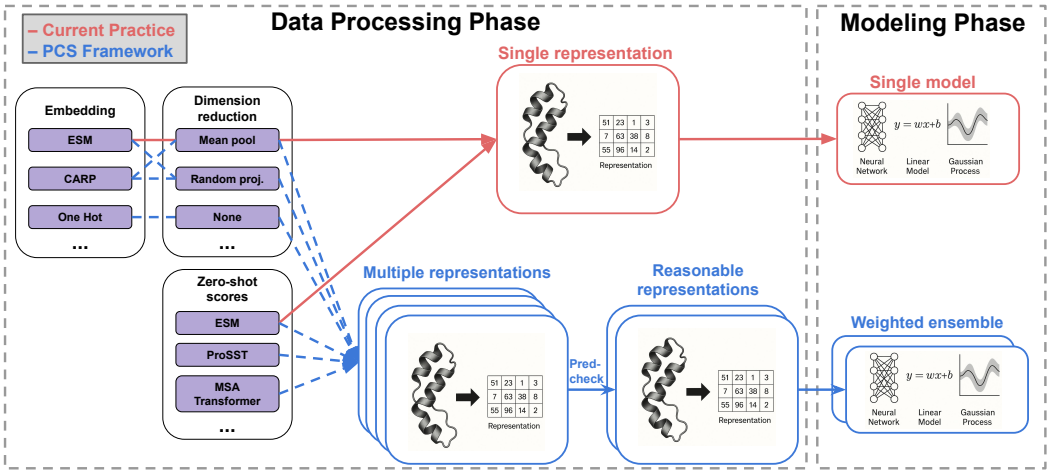

Figure 4.1: We stabilize protein fitness prediction by following the PCS framework. Our approach leverages variability at the representation level by considering multiple representations, filtering out those that are less predictive through a pred-check step, and ensembling across the remaining ones. This leads to substantial gains in both prediction accuracy and uncertainty quantification, highlighting the importance of considering multiple reasonable representations for a protein sequence.

a mutated sequence to its corresponding fitness value y—a scalar quantity that may reflect properties such as thermostability or binding affinity to a given ligand. We assume access to a labeled dataset of n mutations, $(m_1^{(p)}, y_1), \ldots, (m_n^{(p)}, y_n)$, where y_i is the experimentally measured fitness of $m_i^{(p)}$.

Representations of protein sequences To apply ML methods for fitness prediction, protein sequences must first be converted into numerical representations, or *embeddings*, which serve as inputs to models trained to predict observed fitness values. The simplest and still commonly used approach is the one-hot encoding, where a sequence of length L is embedded as a vector of dimension $20 \times \mathrm{L}$ by concatenating L one-hot vectors corresponding to the 20 canonical amino acids [58].

More recently, embeddings derived from pre-trained PLMs [107, 105, 77, 134] have demonstrated superior performance in some prediction tasks [75] due to their ability to transfer information learned from large-scale unsupervised pre-training. These models typically embed a protein sequence as a matrix of size $\mathrm{L} \times h$, where h is the hidden dimension of the PLM. This matrix is often reduced to a fixed-size vector, most commonly via mean pooling across the sequence dimension.

In addition to embeddings, PLMs can produce scalar representations of mutated sequences by approximating the log-likelihood ratio between a mutant and a reference se-

quence, using the sequence distribution learned during training. These likelihood-based scores—known as zero-shot scores—have been shown to correlate with various measures of fitness without requiring supervised training [86, 91].

Beyond PLMs, structure-based representations have also been explored for protein fitness prediction [49]. These include features derived from predicted 3D structures using methods like AlphaFold2 [62] or from inverse folding models such as ESM-IF and ProteinMPNN [57, 33].

It is important to emphasize that the space of *reasonable*[1] choices for representing a single protein sequence is prohibitively large. For example, [75] show that different protein language models can produce embeddings that perform equally well in prediction tasks, and that increasing model size or pre-training data does not necessarily lead to better results. In addition, while many computational methods incorporate a zero-shot score into the representation, there is no universally optimal score: different scores perform best for different tasks [91]. Moreover, although the penultimate layer is often used by default, [75] report that embeddings from alternative layers can offer competitive performance. Finally, while mean pooling is the most common dimension reduction strategy, [74] have shown that alternative pooling methods can perform similarly well. Table 4.1 provides a non-exhaustive list of different and reasonable choices for embedding a single protein sequence.

Methods for fitness prediction The simplest approach for predicting protein fitness is to use a linear model trained on the one-hot encoding of sequences. Such models learn a distinct coefficient for each specific mutation, preventing them from generalizing to mutations unseen in training (whose coefficient value is zero). Additionally, as linear models, they are unable to capture non-linear, epistatic interactions between mutations. Alternatively, zero-shot scores—such as those from PLMs that estimate the density of their training data—typically naturally occurring sequences—have been empirically shown to correlate with fitness without additional training [86, 83]. These scores generalize across mutation sites and can capture epistasis. But as scalar values, they cannot be improved using labeled data, and they perform poorly when natural sequence density does not align with the definition of fitness of interest.

The hybrid approach proposed by [58] combines supervised learning with zero-shot scores. It does this by concatenating a one-hot sequence embedding with a chosen zero-shot score, then fitting a Ridge regression model. This simple method strikes a strong balance between bias and variance, and often beats more complex approaches like deep neural networks or fine-tuned PLMs. In some cases, using embeddings from pre-trained PLMs instead of one-hot encodings can further boost performance [136]. While ESM2 [77] is a popular default for such embeddings, other models—such as CARP [134]—can match or even surpass its performance in specific settings [75].

ProteinNPT [93] is a non-parametric transformer [68] model that combines embeddings from the pre-trained MSA Transformer [105] with zero-shot scores, achieving substantial

[1]We define a representation as reasonable if there is no clear justification, a priori, to expect it will perform poorly for the purpose of fitness prediction.

Table 4.1: A list of reasonable choices for the representation of a single protein. A single protein sequence can be represented by any combination of zero-shot scores, embeddings from a PLM (with any choice of dimension reduction, layer or model) as well as structure and inverse folding information based on this structure. This represents a huge space of potential reasonable ways to represent a single protein sequence.

Category	Component	Reasonable choices
Zero-shot score	Method / Model	ESM1v [86], ESM2 [77], MSA Transformer [105], ProSST [76]
Embedding	Method / Model	ESM1v [86], ESM2 [77], CARP [134], MSA Transformer [105], One-hot [58]
	Layer	Penultimate, first/middle layers, or learned combinations across attention layers [13]
	Dimension reduction	Mean pooling, max pooling, pool over sequence dimension, pool over hidden dimension, mean pooling over mutated sites, flattening [30, 74].
Structure	3D coordinates	AlphaFold2 [62], RosettaFold [4], Experimental (PDB, [12])
	Inverse folding	ESM-IF [57], ProteinMPNN [33]

gains over the hybrid approach, at a high computational cost [93, 49]. Kermut [49] is a GP model for the distribution of the protein fitness conditioned on sequence. Its kernel is the sum of two main components (see Appendix C.5 for details): (1) a sequence kernel, defined by an RBF kernel applied to mean-pooled ESM2 embeddings, and (2) a structure kernel, which is itself the product of three terms—one based on AlphaFold2-predicted distances between mutation sites, another based on the Hellinger distance between inverse folding probabilities computed by ProteinMPNN [33] and a third one based on the likelihood of the sequence under an inverse folding model. Kermut outperforms ProteinNPT while being more computationally efficient, and currently achieves state-of-the-art results on the supervised ProteinGym benchmark [49].

We emphasize that each one of these methods (one-hot, hybrid, ProteinNPT and Kermut) considers a single representation of the protein through some combination of the choices outlined in Table 4.1.

Uncertainty quantification for fitness prediction Existing approaches for UQ in fitness prediction can be broadly categorized into three groups. The first is probabilistic modeling, where assumptions are made about the conditional distribution of fitness given the sequence. Uncertainty estimates are derived from this distribution, whose parameters are estimated from data. These methods rely heavily on correct model specification (e.g., the choice of kernel in Gaussian Processes). A second category involves randomization based uncertainty, for example, ensembling neural networks with different initializations, with the uncertainty defined as the variance of the prediction across the ensemble. This approach has seen empirical success in some tasks [50, 47], but may be suboptimal when the underlying neural network does not provide good predictions. Finally, conformal prediction provides uncertainty estimates in the form of prediction intervals with guaranteed marginal coverage. However, these intervals are not locally adaptive unless combined with an estimate of predictive variance—such as in studentized conformal prediction [72, 1].

It is important to note that all of these approaches quantify uncertainty only at the **modeling** stage of the DSLC. They do not account for uncertainty introduced in **earlier** stages—such as problem formulation, data cleaning, or data processing—which is the primary focus of this work.

4.3 Stable and Pred-Checked (SP) fitness predictors via the PCS framework

Setup We study the supervised fitness prediction task described in Section 4.2, where the goal is to learn a predictor $\hat{f}^{(p)}$ that maps a mutated protein sequence $m^{(p)}$ to a scalar fitness value y, such as thermostability or binding affinity to a given ligand. We assume access to a dataset of n labeled examples $(m_1^{(p)}, y_1), \ldots, (m_n^{(p)}, y_n)$, where y_i denotes the experimentally measured fitness of $m_i^{(p)}$.

We consider base models that output both point predictions and uncertainty estimates. In this work, we study Bayesian Ridge regression, Kermut (a GP) and an ensemble of CNNs, each one of these models has been used for fitness prediction in previous works [50, 49, 47]. Details are provided in Appendix C.5.

We define the representation of the protein as the following triplet: (1) an embedding from a pre-trained PLM, (2) a dimensionality reduction method, and (3) a zero-shot score. To keep the study tractable, we focus on a representative subset. For embeddings, we use ESM1v [86], ESM2 [77], and CARP [134]. For dimensionality reduction, we apply standard mean pooling [30] and three randomized variants. Each variant perturbs the mean-pooling weights: for a sequence of length l, we project the (l by d) embedding matrix using inner product with $\mathbf{m} = (1/l + \epsilon_1, \ldots, 1/l + \epsilon_l)$, where $\epsilon_i \sim \mathcal{N}(0, 1/l^2)$ independently, resulting in a h-dimensional vector. For zero-shot scores, we use MSA Transformer [105], ProSST [76], ESM2 [77] and TranceptEVE [94]. We provide a detailed description of the zero-shot and embedding methods in Appendices C.1.1 and C.1.2 respectively.

Stabilizing and Pred-Checking fitness predictors via the PCS framework We describe our training procedure for exploring the space of reasonable protein representations while also quantifying the uncertainty associated with these choices. Our approach consists of two main steps, mirroring the first two steps of the procedure described in Agarwal et al. [1]. First, we apply a **Pred-Check step** to select a subset of informative zero-shot scores, that vary substantially in predictive power across datasets. To identify the most useful scores, we evaluate their fit to the training data. Of the methods we tested (Appendix C.2), the most effective selected the top three zero-shot features, ranked by their Mean Decrease in Impurity (MDI) scores from a random forest trained on the fitness labels. We then divided each of the three MDI scores by their total sum to obtain non-negative weights that sum to one. These weights were assigned to all models associated with each of the selected scores, and used to compute a weighted average of their predictions (Appendix C.3). We emphasize that the Pred-Check step is partial, as it focuses solely on the zero-shot score. Developing analogous checks for the embedding and dimensionality reduction steps is an important direction for future work. In the second step (**the ensembling step**), we consider the Cartesian product of the selected zero-shot scores (3 in total), the embeddings (3 in total), and the dimensionality reduction methods (4 in total) as our space of reasonable representations ($3 \times 3 \times 4 = 36$ representations in total). On each representation within this set, we train a single base model on the training dataset, resulting in an ensemble of 36 models. Final predictions are computed as a weighted (by the normalized MDI values) average of the ensemble outputs. Uncertainty is estimated by combining two sources: (1) the average uncertainty reported by the base models, and (2) the empirical standard deviation of the predictions across different representations.

This procedure builds on the PCS-UQ framework introduced by [1], but shifts the focus from algorithmic variability to sensitivity with respect to data representation choices. Therefore, our approach includes two key modifications. First, the Pred-Check step filters representations (i.e., zero-shot scores) rather than algorithms. Second, instead of using bootstrap resampling, we perturb reasonable data representations. These changes reflect the central goal of this work: to highlight the often-overlooked impact of representation-level sensitivity on model performance.

4.4 Experiments

Choice of experiments We design our experimental evaluation aiming to cover a wide range of proteins and notions of fitness, and aligns with prior work. Specifically, the supervised ProteinGym benchmark experiments in Section 4.4.1 follow the setup used in [93, 49]. The analysis of prediction interval width and empirical coverage for protein fitness was previously carried out for fluorescent proteins in [38]. In Section 4.4.2, we carry this analysis to the ProteinGym dataset, that includes a much larger set of proteins. The iterative protein design experiment in Section 4.4.3 also follows the protocol introduced in [93].

4.4.1 ProteinGym benchmark performance

Setup We evaluate our methods on the supervised prediction benchmark from ProteinGym [91], which comprises 217 experimentally measured assays spanning a diverse set of proteins, organisms, and fitness definitions. Each dataset contains single-substitution mutations—i.e., amino acid substitutions at a single position—with corresponding experimental fitness measurements, as described in Section 4.2. Results on datasets containing double mutants are deferred in Appendix C.7.2. While there are other benchmarks for protein design, such as FLIP [30], we select ProteinGym because it is the largest and most diverse benchmark.

For each dataset, we use 80% of the mutated sequences for training and evaluate predictive performance on the remaining 20%. ProteinGym provides three types of data splits:

- **Random split**: sequences are randomly assigned to the training and test sets.
- **Modulo split**: sequences with mutations at every fifth residue are assigned to the test set.
- **Contiguous split**: the sequence is divided into five equal segments, and each fold contains sequences with mutations in residues from one segment.

We emphasize that both the modulo and contiguous splits introduce distribution shifts by ensuring that train and test mutations affect disjoint sets of positions along the protein sequence, thereby representing a form of covariate shift.

Methods As baselines, we consider three models adopted from prior work: (1) a Bayesian Ridge regression model [47], (2) a Kermut regressor [49], and (3) an ensemble of four CNNs [50, 47] (details for the models are provided in Appendix C.5). For each of these models, we use ESM2 embeddings with mean pooling across the sequence dimension, combined with the ESM2 zero-shot score as additional input. These same models also serve as the base for our SP predictors, where they are trained separately on each selected representation before being ensembled. We also report results from ProteinNPT [93] that is a strong baseline, though we do not reimplement a SP version of it due to its high computational cost [49]. For each of the three models (Bayesian Ridge, Kermut and CNN), we implement a SP version following the procedure described in Section 4.3. These SP versions are weighted ensembles of the base models fitted on the same training data with different representations of the protein sequences. Each SP version consists of 36 models.

Results The results, presented in Figure 4.2, show that SP predictors consistently outperform their base estimators across all models and data splits. Among them, SP Kermut achieves the best performance, followed by SP Bayesian Ridge. The improvements are particularly notable on the more challenging *modulo* and *contiguous* splits, that introduce a covariate shift between training and test sets. On average, across the two evaluation splits, SP CNN increases Spearman correlation by 22%, SP Bayesian Ridge by 30%, and SP Kermut

by 9% relative to their respective base models. Remarkably, SP Bayesian Ridge outperforms ProteinNPT across all splits and metrics, achieving performance on par with Kermut. These results demonstrate that mitigating instability due to data representation can yield improvements comparable to—or even greater than—those achieved through the development of new algorithms, underscoring the critical role of representation choice. In addition to improved accuracy, SP estimators also provide more reliable uncertainty quantification, as evidenced by stronger Spearman correlations between predicted uncertainty and absolute error of a model—exceeding a 50% gain for Kermut, and yielding several-fold improvements for CNN and Bayesian Ridge. Appendix C.3 presents a detailed ablation of the SP pipeline. Each component—ensembling, Pred-Check, and others—shows statistically significant gains in at least one setting (i.e., base model and split), and none reduce performance in any case.

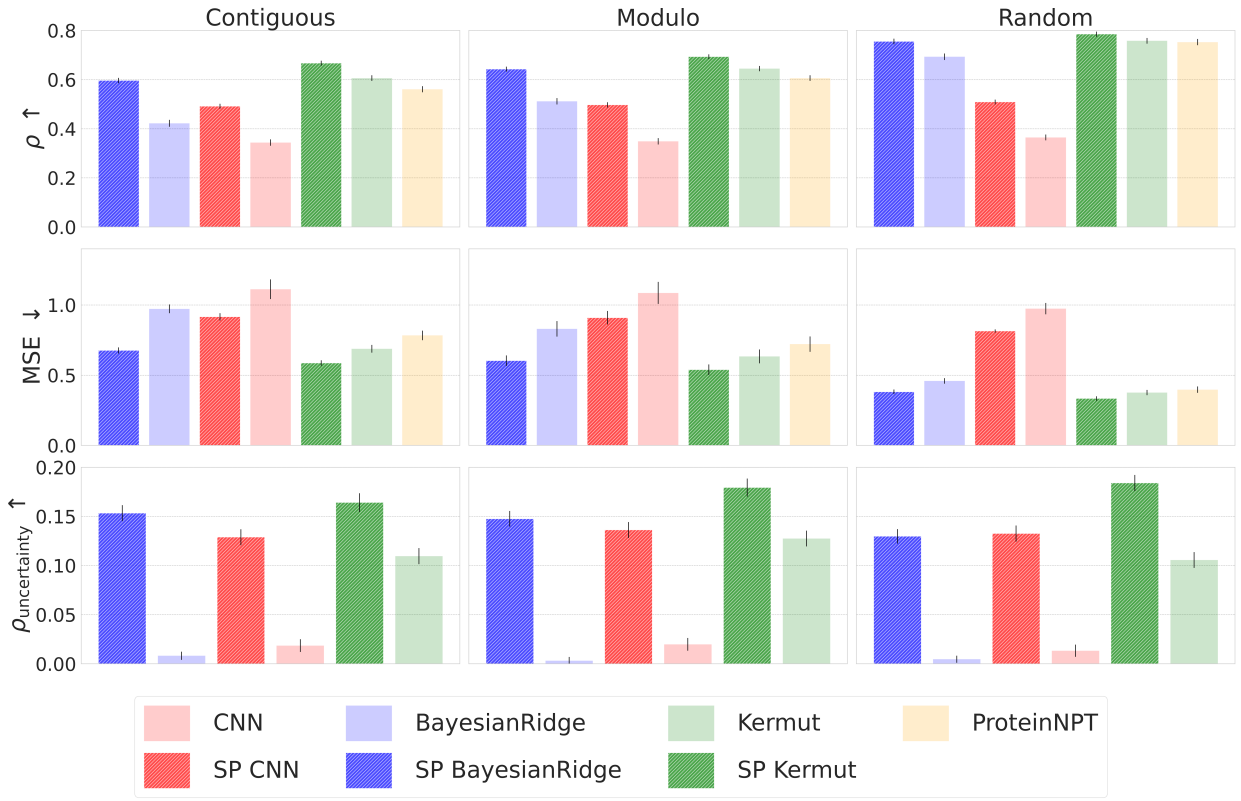

Figure 4.2: SP regressors outperform their base estimators on ProteinGym. We report the average Spearman correlation (ρ), mean squared error (MSE), and the Spearman correlation between the uncertainty scores and the absolute errors of the predictions ($\rho_{\text{uncertainty}}$). Each column corresponds to a different ProteinGym split. SP Kermut outperforms all other methods across all metrics and splits, while SP Bayesian Ridge is on par with Kermut.

4.4.2 Sharper prediction intervals

Setup To evaluate the quality of uncertainty quantification, we measure the coverage and width of prediction intervals on a held-out test set. The intervals are first calibrated on a separate validation set to achieve a target coverage level. We compare three calibration techniques: (1) split conformal [72], (2) studentized conformal [72], and (3) PCS-UQ [1]. In our setting, all methods guarantee the same marginal coverage, assuming the calibration and test sets are exchangeable. Detailed descriptions of these approaches are provided in Section C.4. For each dataset, under the random train-test split, we reserve 50 data points from the test set for calibration (14 datasets whose test set included less than 50 data points were removed) and evaluate prediction interval coverage and width on the remaining test points.

Results The results for Kermut and SP Kermut are shown in Figure 4.3, with corresponding results for Bayesian Ridge and CNN provided in Appendix C.4.2, in Figures C.5 and C.4. We find that SP regressors produce sharper prediction intervals, achieving approximately 10% narrower median widths across assays at the same coverage level. Notably, for the strongest Kermut predictor, the PCS-UQ approach proposed by [1] significantly outperforms conformal methods, reducing the median interval width by up to 20% while maintaining the same coverage.

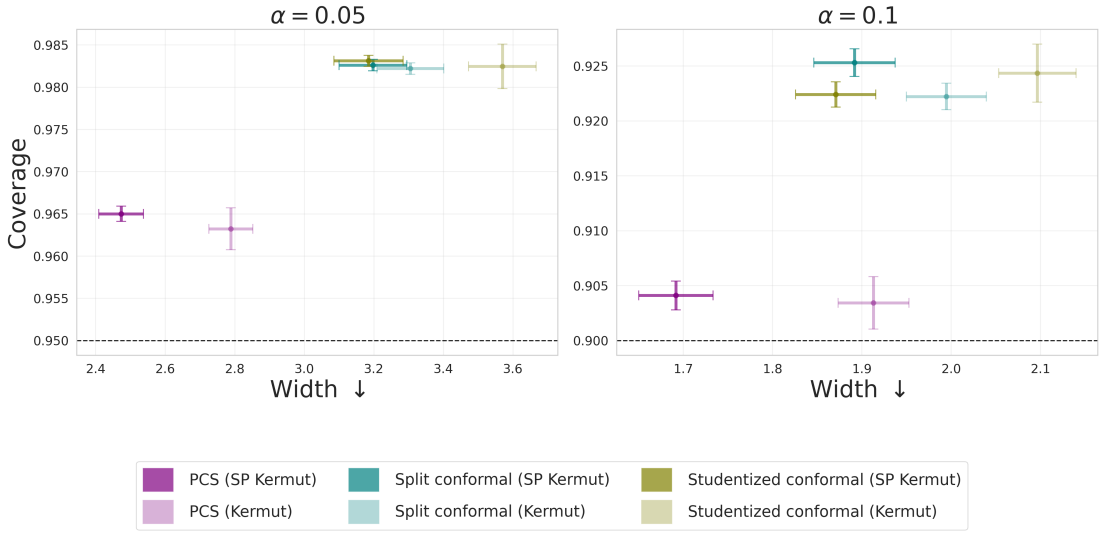

Figure 4.3: SP Kermut achieves sharper prediction intervals across calibration methods. We report the median interval width and the average empirical coverage with standard errors for Kermut and SP Kermut using three prediction interval techniques. The left and right panels correspond to target coverages of 95% and 90% , respectively.

4.4.3 In-silico protein engineering

Setup We conduct an in-silico protein engineering campaign with the goal of identifying high-fitness sequences using as few evaluations as possible. We implement an iterative Bayesian Optimization (BO) procedure, beginning with an initial batch of labeled sequences used to train a predictive model. At each iteration, we select the next batch of k sequences $(m_1, \ldots, m_k)$ using the upper confidence bound (UCB) acquisition function (which is commonly used in these settings [50, 93, 47, 133]), defined as:

$$\hat{f}(m) + \lambda \hat{\sigma}(m), \tag{4.1}$$

where $\hat{f}(m)$ denotes the predicted fitness, $\hat{\sigma}(m)$ represents the model's uncertainty and λ controls the exploration-exploitation tradeoff. We evaluate two choices for λ, $\{0.1, 2\}$, as proposed by [93, 133] respectively, and find that $\lambda = 2$ yields stronger performance in recovering the highest-fitness sequences, while $\lambda = 0.1$ yields better performance in recovering a larger number of high fitness sequences (i.e., their fitness values belong to 70th or 90th precentile of all measured fitness values within their assay).

Because the uncertainty estimates $\hat{\sigma}$ vary in scale across models, we normalize $\hat{\sigma}$ for each method so that it has the same ℓ_2 norm as $\hat{f}$, across the sequences we evaluate UCB on. Since this is an in-silico evaluation, we use the measured fitness values from the ProteinGym benchmark as ground truth.

For each dataset, we initialize the BO loop with 50 randomly selected sequences (two datasets with less than 50 sequences are removed) and acquire 50 additional sequences per round over 5 rounds, resulting in a total of 250 labeled sequences. This setup mirrors the structure and scale of real-world protein engineering campaigns [133]. We evaluate three base predictors—CNN ensemble, Bayesian Ridge, and Kermut—along with their SP variants.

Results SP variants of all base methods—Kermut, Bayesian Ridge, and CNN—consistently outperform their base counterparts. Figure 4.4 reports the cumulative fraction of assays in which the highest-fitness sequence is recovered across five steps of Bayesian optimization (BO), averaged over three independent runs with different random selection of the initial dataset. At every step, SP Kermut and SP Bayesian Ridge recover the top sequence in more assays than their base versions. By step five, SP Kermut shows a 6% absolute improvement over base Kermut.

Appendix C.6 presents additional results, including comparisons at $\lambda = 0.1$ and further analysis on recovering multiple high-fitness sequences, measured using quantiles of each assay's fitness distribution. At $\lambda = 0.1$, SP Kermut recovers on average across assays 3% more sequences in the top 70th percentile and 5% more in the top 90th percentile, compared to base Kermut. While both methods perform similarly in recovering the single top sequence at this setting, their performance remains below that of SP Kermut and SP Bayesian Ridge at $\lambda = 2$.

At $\lambda = 2$, SP Bayesian Ridge achieves the strongest performance in recovering sequences above the top 70th and 90th percentiles. However, its ability to recover the absolute top sequence lags behind Kermut, SP Kermut, and its own $\lambda = 0.1$ variant.

In summary, for both top-sequence recovery and high-percentile recovery, SP Kermut achieves the best overall performance—with $\lambda = 2$ performing best for identifying the top sequence, and $\lambda = 0.1$ performing best recovering a large number of high fitness sequences.

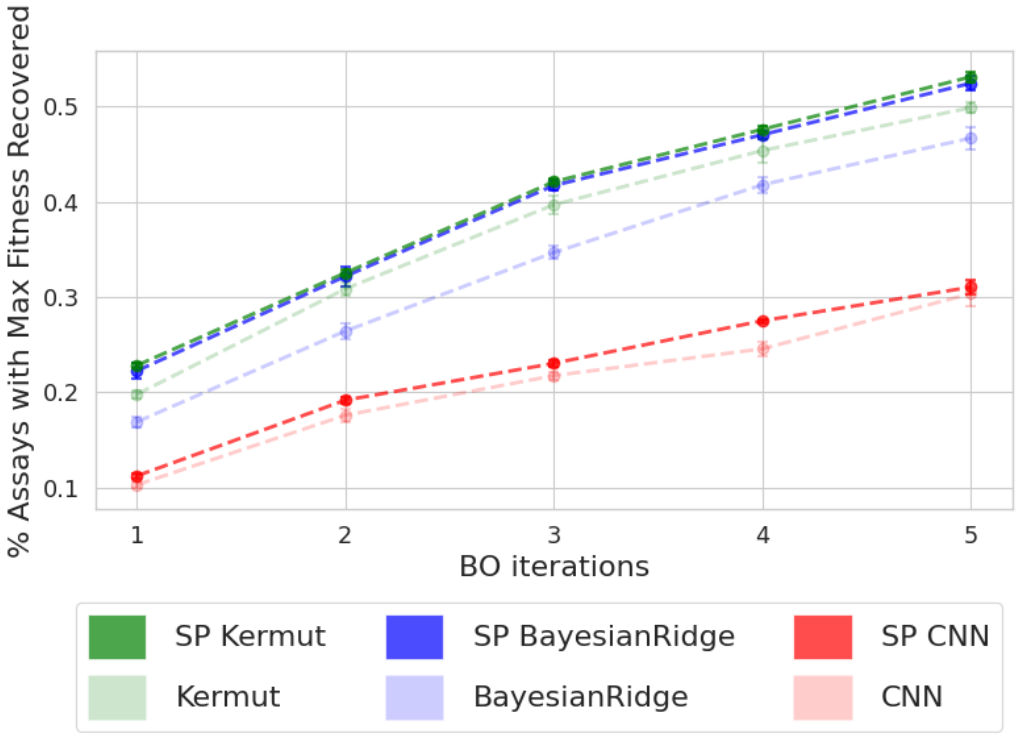

Figure 4.4: SP predictors recover the highest-fitness sequence more frequently than their base models, for BO with $\lambda = 2$. The y-axis shows the percentage of experiments in which the highest-fitness sequence is recovered, and the x-axis shows the number of Bayesian optimization steps. Results are averaged over three random initializations of the training set. Error bars indicate standard errors across runs, though they may be too small to see.

4.5 Discussion

This chapter highlights the often-overlooked impact of instability from representation choices in protein fitness prediction, and shows that leveraging this variability—guided by the PCS framework—can yield substantial performance gains, particularly on out-of-distribution

(OOD) data such as covariate shift. While we examined a subset of reasonable representations, such as zero-shot scores and embeddings from pre-trained models, many parts of the representation space remain unexplored—for example, the choice of network layer. Stabilizing and Pred-Checking predictions with respect to these choices presents a promising direction for future improvement. Although SP predictors yield improved performance, they come with increased computational cost: in our experiments, we used an ensemble of 36 models, which is relatively modest in size. Incorporating additional Pred-Check steps to filter embeddings and dimensionality reduction methods could reduce this cost.

Chapter 5

Ultra-Stable iRF: Enhancing Epistasis Detection Precision in Rare Genetic Variants Studies

5.1 Introduction

Identifying epistatic interactions, where the phenotypic effect of one genetic variant depends on the genetic context of others, remains a formidable challenge at the frontier of functional human genomics. Large-scale genetic studies typically prioritize additive effects of common variants or marginal gene-level effects of rare variants. This strategy, which largely overlooks epistasis, is a natural first step aimed at reducing computational costs and increasing statistical power. As a result, the discovery of epistatic interactions has progressed slowly, contributing to the persistent "missing heritability" gap [144, 143].

Decision trees and their ensembles are well-suited for modeling epistasis due to their ability to capture interactions through thresholding along decision paths, as well as their interpretability. Recently, iterative Random Forests (iRF) [9] have shown empirical success in identifying epistatic interactions in large-scale studies. Specifically, iRF has successfully identified known epistatic interactions for red hair and multiple sclerosis [10]. iRF has also successfully discovered interactions regulating genetic control of cardiac hypertrophy, which were subsequently validated in gene-silencing experiments [127]. However, these efforts have primarily focused on common genetic variants, occurring with Minor Allele Frequency (MAF) larger than 5%, as they are statistically easier to study. The next and more challenging step is to study the role of rare genetic variants.

Extending epistasis detection to rare genetic variants is critical because these variants are believed to have larger effect sizes and significantly influence human phenotypes [27, 143]. However, their infrequent occurrence makes interactions between rare variants exceptionally rare, and therefore challenging to study and understand. This necessitates computational methods that are both efficient and biologically meaningful, capable of reliably detecting

subtle signals from limited data. In particular, useful biological information, such as that captured by Protein Language Models (PLMs) [86, 107], must be effectively utilized to achieve the goal of identifying epistatic interactions among rare variants.

This chapter focuses on developing an ML algorithm for gene-gene epistasis discovery using rare genetic variants with iRF, by following the Predictability, Computability, and Stability (PCS) framework [139], to best leverage the biological information capture by PLMs. Our core contribution is enhancing iRF with another layer of stability, by considering multiple reasonable numeric representations of the single nucleotide polymorphism (SNP) data. These representations are derived by leveraging the central dogma of molecular biology utulize information captured at the protein level by PLMs (throught their pre-training data) as follows. We use three distinct PLMs to embed missense variants into the PLMs latent spaces, offering an alternative to the conventional dose representation. Considering all of these four different representations in our analysis instead of a single one, results in our proposed Ultra-Stable iRF (US-iRF) algorithm.

Like most epistasis discovery methods [142, 127], we begin with a filtering step. This step reduces the number of genes (which is the most commonly used biological unit for studying rare variants [101]) under consideration to a smaller, computationally feasible set for running iRF. Specifically, for a given phenotype, we employ two filtering steps: the first is to select genes that have significant associations in genome-wide gene-level rare variants association tests. The second further filters this list based on prediction performance, to obtain a small set of genes whose SNPs are predictive of the phenotype. Given this small set of genes, an iRF model is fitted to predict the phenotype based on all the SNPs belonging to these genes and interactions are extracted from the fitted model. US-iRF uses multiple representation in this step and averages the interaction stability scores across these representations to enhance its stability.

Our study demonstrates that US-iRF accurately recovers interactions from established databases with enhanced precision compared to iRF that relies on a single representation or to MatrixEpistasis [142] that is a linear model with multiplicative interaction terms and still a leading method for interaction detection [127]. Indeed, incorporating this representation-level stability substantially improves US-iRF's accuracy in identifying interactions previously reported in both the STRING database (the largest available dataset [118]) and BioGRID (manually curated experimental evidence [97]), compared to iRF.

Our contributions We summarize our main contributions below:

- **Algorithm development** — We introduce **US-iRF**, a method for detecting epistatic interactions in large-scale rare genetic variant studies. Its distinctive features are:

    1. An additional prediction-based gene-filtering step, which prioritizes genes more likely to be affect the phenotype, thereby reducing computational complexity and increasing the signal-to-noise ratio.

2. The use of **PLM embeddings** to integrate biological knowledge captured in protein sequences.
3. The use of **multiple representations** rather than a single one, substantially improving stability.

- **Evaluation on large-scale data** — We evaluate US-iRF's ability to recover previously reported gene–gene interactions using whole-exome sequencing data from 200,000 individuals in the UK Biobank. Our evaluation focuses on recovering interactions from two human gene–gene interaction databases: STRING and BioGRID.
- **Performance gains** — US-iRF substantially outperforms both iRF (using the dose representation) and MatrixEpistasis (using either single or multiple representations). In tests for gene–gene interactions affecting 20 quantitative traits from the UK Biobank [116], **59%** of the top 9% of interactions discovered by US-iRF had support in the STRING database, and **13%** of the top 5% had experimental support in BioGRID. For comparison, the corresponding STRING support rates were **51%** for iRF with the dose representation, **18%** for US-MatrixEpistasis, and **24%** for MatrixEpistasis; the corresponding BioGRID support rates were **7%**, **1%**, and **5%**, respectively. Notably, both US-iRF and iRF with the dose representation outperformed US-MatrixEpistasis and MatrixEpistasis in recovering biologically validated interactions in both STRING and BioGRID.

5.2 Methods

5.2.1 Notation

For a cohort of n individuals, we denote the n-dimensional phenotype (e.g., cholesterol level or height) vector as $\mathbf{y}$, the $n \times d$ covariates matrix, containing sex, age, sex^2, sex $\times$ age and the first 10 population principal components [104], as $\mathbf{X}$. We consider the $n \times l$ genotype for gene g matrix as $\mathbf{G}^{(g)}$ (where l can vary depanding on the context such as the specific gene or numeric representation of the SNPs).

5.2.2 Phenotype and genotype data from the UK Biobank

We follow a standard data processing steps, aligned with previous work [3].

UK Biobank whole-exome sequencing data We used the UK Biobank WES 200k interim release, which includes 200,580 participants and 15,891,882 variants. We study the 21 quantitative traits from blood biochemistry marker measurements (Category 17518) and standing height (Data-field 50) as the main phenotypes in our study (these traits were previously studied in McCaw et al. [85] and Clarke et al. [28]). In the association analyses, sex, age, sex^2, sex $\times$ age and the first 10 genetic PC are used as covariates. Our analysis

was restricted to unrelated individuals, which were identified using the official precomputed genetic kinship classification (Data field 22021) to select individuals with no kinship found. Overall, this includes 140,290 participants.

Variant data preprocessing We accessed the project-level VCF data and performed the following data processing procedures on the UK Biobank Research Analysis Platform (RAP). We applied the variant-level quality control (QC) using bcftools v1.2 [31] following similar procedures to previous studies [28]. We first set genotype calls to be missing when they did not meet minimum depth thresholds (<7 reads for SNPs or 10 for indels). We then removed variants with over 10% missing genotypes. Next, variants exhibiting anomalous allele balance, defined here as containing alternate-homozygous calls or heterozygous calls with abnormally high alternate allele fractions (exceeding 15% for SNPs or 20% for indels) were excluded. Finally, we filtered out those variants deviating significantly from Hardy–Weinberg equilibrium (p-value $\leq$ 1e-15). This resulted in 13,204,155 variants after QC.

Variant annotation and conversion to protein changes In order to generate protein embeddings for our analysis, the genomic sequence variants need to be converted to protein sequence variants. We used SnpEff [25] and SnpSift [26] v5.0 to annotate and filter the variants with the default GRCh38.99 databases in the software. When running SnpEff, we restricted the annotations to only protein-coding sequences, and to only the canonical isoform of each gene defined in the database. We then use SnpSift to filter the variants retaining only those annotated to be in the "MISSENSE" functional class, or classified as "HIGH" in impact prediction, which includes loss-of-function variants such as protein-truncating, frameshift, stop gain, stop loss, start loss, and splice donor/acceptor variants. Among the missense variants, we extracted the inferred protein change, which is further used to embed the SNPs using PLMs in our procedure.

5.2.3 Numeric representations of SNPs

In any genome-wide association study the SNP data is converted into numerical representations to facilitate statistical analysis. We first describe below the dose representation, that is most commonly used, and then describe representations based on PLMs which have not been used before in genomic studies to the best of our knowledge.

Gene-Level dose representation Most commonly, SNPs are numerically encoded using an additive dosage model (also referred to as *dose* representation). Under this representation, an individual's genotype at a biallelic SNP is assigned a numerical value based on the count of the alternate allele (or effect allele): 0 for the homozygous reference genotype, 1 for the heterozygous genotype, and 2 for the homozygous alternate genotype. This representation can be conceptualized in terms of a genetic "distance" in an embedding space, where the base

pairs are represented by one-hot encodings and the distance is an ℓ_1-norm, it fundamentally captures the incremental effect of allele dosage. In this work, the gene-level representation derived from the additive dosage model is obtained by concatenating the individual SNP encodings for all variants within that gene, and summed at the individual level. We emphasize that this representation is not restricted to missense variants.

Gene-Level representation derived from PLMs The motivation for using PLMs to extract embeddings stems from their demonstrated success in predicting the functional consequences of sequence variations and their ability to infer information about protein three-dimensional structure [92, 75]. This supports the belief that PLMs are well-suited to inform the analysis of rare missense variants and their effects at the protein level.

Following the central dogma of molecular biology, each SNP annotated to cause a missense mutation is translated into the corresponding protein sequence. This protein sequence is then embedded using a PLM into a $\mathrm{L} \times h$ matrix (for its penultimate layer), where L represents the number of amino acids in that protein and h denotes the hidden dimension of an embedding matrix from the PLM. To derive a SNP-specific representation, an element-wise difference is computed between the $\mathrm{L} \times h$ matrix embedding of the variant protein sequence and that of its corresponding reference protein sequence. The absolute values of this difference matrix are then averaged across the hidden dimension (i.e., the columns of the embedding matrix), resulting in an L-dimensional representation for each SNP.

As language models, PLMs also define a log of the likelihood ratio score for each mutation, conditional on the rest of the amino acid sequence. Formally, let p be the likelihood over protein sequences defined by a given PLM. Then for a mutated sequence s^{mut} and a reference sequence s^{ref} (sequence defined by the reference allele), both consisting of L amino acids, the log of the likelihood ratio score given by that PLM is defined as [86]

$$\sum_{i=1}^{\mathrm{L}} \log(p(s_i = s_i^{\mathrm{mut}} | s_{/i}) / p(s_i = s_i^{\mathrm{ref}} | s_{/i})), \tag{5.1}$$

where $s_{/i}$ is the reference sequence masked at the i-th amino acid. As is common in protein fitness prediction [58], we concatenate this scalar to the L-dimensional representation of each SNP (derived from the embeddings matrices as described above), resulting in $(\mathrm{L}+1)$-dimensional representation for each SNP.

The final aggregated gene-level representation is an $(\mathrm{L}+1)$-dimensional vector, obtained by summing all SNP representations that describe mutations in that gene at the individual level, weighted by genotype: 1 for a heterozygous variant and 2 for a homozygous variant. Following the analysis in Chapter 4, we considered three different PLMs: ESM2 [77], ESM1v [107] and CARP [134] to derive a distinct representation from each one, in our Ultra-Stable procedure.

5.2.4 Three-step gene-gene epistatic interaction detection with prediction guided dimensionality reduction

Epistasis discovery methods typically incorporate an initial screening (or filtering) step, a necessity driven by the substantial computational and statistical challenges inherent in identifying interactions at a genome-wide scale [128, 29, 142, 127]. Following this established paradigm, our new algorithm US-iRF employs two sequential screening (or filtering) stages. The first stage involves rare variant association testing conducted across the entire genome (excluding the sex chromosomes, similar to previous studies [28]). This step reduces the pool of candidate genes to a few hundred genes per phenotype. To further refine this selection, we then implement a prediction-based screen (following the PCS framework for veridical data science [139, 138, 106]), which further narrowed the number of candidate genes to a maximum of a few dozens per phenotype (full details are provided in Appendix D.1). In the final step, the genes that successfully passed both screening stages are then used for interaction detection, where only interactions between these genes are considered, significantly reducing the computational burden and increasing to signal to noise ratio. For the purpose of interaction detection, we evaluate both iRF [9] that is a decision tree based method, and MatrixEpistasis [142] that is a linear multiplicative interaction method. Both are the leading pairwise epistatic interaction methods [127] and are described in detail in Section 5.2.4.1.

The US-iRF algorithm takes as input the full genotype data (i.e., VCF file) and the phenotype data. In Step 1, an initial gene screening is performed via rare variant association testing. In Step 2, the list of genome-wide associated genes is further filtered using a prediction-based check. In the final step, iRF searches for interactions among the genes that passed the first two steps, considering four possible SNP representations based on the dose encoding and three PLMs. The algorithm outputs a stability score for every possible interaction between gene pairs that pass both screening steps.

These three steps are described in detail below.

Step 1 - Initial Gene Screening: For each phenotype, an initial screening of the entire genome was conducted to identify marginally associated genes. Similar to Clarke et al. [28], we used the genome-wide significantly associated genes discovered in one of two full 500k UK Biobank WES studies [3, 64].

This preliminary step reduced the pool of candidate genes for each phenotype to at most 211 per phenotype (full details are provided in Table D.1).

Step 2 - Prediction-Based Gene Screening: To further refine the set of genes considered for subsequent interaction detection, a marginal gene prediction-based screen was performed. For each gene-phenotype pair, the corresponding embedding matrices were obtained. We denote the three distinct PLM embedding (for the penultimate layer) matrices as $\mathbf{G}_{\text{ESM2}}$,$\mathbf{G}_{\text{ESM1v}}$, and $\mathbf{G}_{\text{CARP}}$. Additionally, a gene dosage embedding matrix is denoted as $\mathbf{G}_{\text{Dose}}$. The phenotype vector ($\mathbf{y}$) was first regressed on the covariate matrix ($\mathbf{X}$) using

ordinary least squares to derive a residual vector ($\mathbf{r} = \mathbf{y} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{y}$). This residual vector, representing the proportion of phenotype variance unexplained by covariates, was then utilized to assess the predictive power of each embedding matrix. For each embedding matrix, the dimensionality was first reduced to 20 using Principal Component Analysis (PCA), unless the original dimensionality was already 20 or less. We found that using 20 dimensions performed equally well compared to larger dimensions. For PLM embeddings the dimensionality reduction is only applied to the features derived from the hidden representation of the penultimate layer and not to the scalar log of the likelihood ratio scores. Following this, the test-set Spearman correlation was computed from a Ridge regression model, executed across 10 random 80/20 train-test data splits. The regularization parameter for this model was selected via 5-fold cross-validation. Genes for which the calculated Spearman correlation, averaged across these 10 runs, was significantly greater than zero at a nominal $\alpha = 0.05$ for at least one embedding matrix were advanced to the interaction discovery stage. A Bonferroni correction for the number of tested genes was applied for each phenotype.

Step 3 - Gene-Gene Interaction Discovery: At the final step, for each phenotype, we gather all genes that passed the first two steps. Their individual embeddings are then concatenated into a single matrix. For instance, when considering the dose embedding, the concatenated matrix for genes $g_1, \ldots, g_k$ would be:

$$\mathbf{G}_{\text{Dose}}^{(g_1,\ldots,g_k)} = [\mathbf{G}_{\text{Dose}}^{(g_1)}, \ldots, \mathbf{G}_{\text{Dose}}^{(g_k)}]. \tag{5.2}$$

This combined matrix serves as the input for an interaction discovery method. We employ two such methods: iRF and MatrixEpistasis (Section 5.2.4.1 provides a detailed description of these methods). Each method subsequently outputs a score for every potential gene-gene interaction, with iRF providing a Stability score and MatrixEpistasis a p-value. Prior to running iRF, we apply PCA dimensionality reduction to each gene and each representation, consistent with Step 2. For MatrixEpistasis, we explore two approaches: performing dimensionality reduction analogous to iRF, and utilizing the full dose matrix as originally proposed by Zhu et al. [142].

Adding Representation-Level-Stability To further enhance the Stability of iRF and MatrixEpistasis, we introduce an additional layer of stability by averaging results across different SNP representations (embeddings from PLMs or does). Specifically, because Step 3 is dependent on a particular choice of SNP representation, and there is no inherent reason to prefer one representation over another, we execute this step using all four distinct representations detailed in Section 5.2.3. The final score for each interaction is then calculated as the mean stability score (for iRF) or the Cauchy combination [81] of the p-values (for MatrixEpistasis, to preserve the null distribution) obtained from these four separate runs (one using each representation) of Step 3. We refer to iRF and MatrixEpistasis, when enhanced with this representation-level stability, as **Ultra-Stable iRF (US-iRF)** and **Ultra-Stable MatrixEpistasis (US-MatrixEpistasis)**, respectively.

5.2.4.1 Methods for Interaction Discovery

This section reviews the leading computational methods for identifying interactions between genes: MatrixEpistasis and iRF [9, 142, 127], used in our study.

MatrixEpistasis MatrixEpistasis [142] is a statistical method designed to identify epistatic interactions at the SNP level. It employs a linear model that incorporates multiplicative interaction terms.

Specifically, for any given pair of SNPs (or more generally, features derived from genetic data in our analysis), $\mathbf{G}_i^{(a)}$ from gene a and $\mathbf{G}_j^{(b)}$ from gene b, MatrixEpistasis models their relationship with a phenotype ($\mathbf{y}$) as follows:

$$\mathbf{y} = \mathbf{X}\beta + \gamma_1 \mathbf{G}_i^{(a)} + \gamma_2 \mathbf{G}_j^{(b)} + \gamma_3 \mathbf{G}_j^{(b)} \cdot \mathbf{G}_i^{(a)} + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim N(\mathbf{0}_n, \boldsymbol{\sigma}^2 I_n). \tag{5.3}$$

Interaction detection between $\mathbf{G}_j^{(b)}$ and $\mathbf{G}_i^{(a)}$ is performed by statistically testing whether the interaction coefficient, γ_3, is significantly different from zero. This test relies on the normality assumption of the error term, as stated in equation (5.3).

In the context of our analysis, where our goal is to identify interactions between genes (rathar than SNPs), we define two genes, gene a and gene b, as interacting if at least one pair of their associated SNPs or features ($\mathbf{G}_j^{(b)}$ and $\mathbf{G}_i^{(a)}$) demonstrates a statistically significant interaction. To formalize this we denote the event gene a and gene b are interacting as $p_{a \longleftrightarrow b}$. We deonte the pvalue for the SNP/feature-wise interactions as $p_1, \ldots, p_n$. Under the null hypothesis, assuming the model is correctly specified, all p-values for potential interactions between the two genes follow a uniform distribution (an assumption we do not verify). We use the Cauchy combination method [81] to aggregate SNP- (or feature-) level p-values into a single gene-level p-value. The Cauchy combination preserves the uniform distribution under the null given the linear model assumptions and does not introduce any additional assumptions.

iRF (Iterative Random Forests) iRF [9] builds a sequence of random forests and uses their interpretability to discover important features and their interactions.

The process unfolds iteratively. In each iteration, a new random forest is constructed. A key aspect is how features are selected for growing individual trees within this forest: their probability of selection is weighted. This weighting is determined by the Mean Decrease in Impurity (MDI) values calculated from the random forest of the previous iteration. MDI quantifies how much a feature contributes to improving the homogeneity (purity) of the data within the decision trees. Features with higher MDI values from the previous iteration are given a higher chance of being selected in the current iteration, allowing the algorithm to focus on more relevant features.

Once the iRF model has been trained, it leverages Random Intersection Trees (RIT) [112]. RIT is a technique designed for efficiently identifying combinations of features that

frequently appear together along the decision paths within the fitted random forests. These frequently co-occurring features are indicative of potential interactions.

To further enhance the reliability and robustness of the identified interactions, iRF incorporates a stability assessment. It performs the RIT analysis not just once, but across multiple bootstrap samples of the original dataset. By repeating the RIT analysis on these varied samples, iRF can gauge the consistency of interaction detection. The stability score for a particular interaction is then reported as the percentage of these bootstrap samples in which that interaction was successfully identified.

Similar to our approach with MatrixEpistasis, when defining gene-level interactions using iRF, we consider two genes to be interacting if any of their associated SNPs or features show an interaction. Consequently, we define the gene-level stability score as the maximal stability score observed among all individual SNP-level or feature-level interactions associated with those two genes. This means that if even one strong, stable interaction is found between any features belonging to the two genes, the gene-level interaction is deemed stable.

5.3 Results

5.3.1 Evaluation Through Recovery of Known Gene-Gene Interactions

To evaluate how well the different methods identify true gene–gene interactions, we measure their ability to recover interactions previously reported in the literature. This serves as a surrogate metric, that been used before for validating interactions discovery [**gao2010classification**], but is imperfect for two main reasons. First, there may be *false negatives*—interactions that truly exist but are not reported in these databases, for example because they have never been studied. Second, the interactions in these databases are not specific to a given phenotype, meaning there may also be *false positives*—interactions that exist but do not affect the phenotype under consideration. We adopt this surrogate metric because we do not perform follow-up experiments for each interaction, in contrast to the more rigorous evaluation performed by Wang et al. [127].

To find interactions that were previously reported and to ensure our results are not specific to a particular curation method, we consider two protein-protein interaction databases: STRING [118] and BioGRID [97].

BioGRID BioGRID is a manually curated database containing protein-protein interactions across various species, although in our work, we only use interactions reported in humans. Interactions are defined as either physical interactions (i.e., physical binding of the proteins coded by the genes), co-existence on a stable protein complex, and genetic interaction (e.g., increased expression of one gene impairs growth in a strain deficient in another). The reported interactions in BioGRID are backed primarily by experimental evidence in literature, from both low- and high-throughput experiments.

STRING STRING is a database that integrates known and predicted protein-protein interactions, encompassing both direct (physical) and indirect (functional) associations [118]. While STRING covers interactions across various species, our work focuses solely on human interactions. The database compiles and scores evidence from multiple sources. Specifically, STRING incorporates seven evidence channels for genetic interactions, with higher scores indicating more frequent occurrences. *Neighborhood* scores are assigned when two genes occur in close proximity in prokaryotic genomes. *Co-occurrence* scores capture the presence or absence of genes across species. *Gene fusion* scores are based on cases where two genes appear fused in some genomes. *Co-expression* scores indicate correlated expression patterns in the same species. Additional scores are derived from *experimental* evidence and from curated *databases* such as BioGRID. Finally, *text mining* uses machine learning models to identify co-mentions of gene names in the abstracts of scientific publications.

All interactions within STRING undergo critical assessment and scoring. Each interaction channel is assigned a confidence score, ranging from 0 to 1, which indicates the likelihood of the interaction being present in a small more carefully curated database such as KEGG [63]. The final aggregated score, based on the seven channels described above, denoted as $s_1, \ldots, s_7$, is calculated as Von Mering et al. [124]:

$$s = 1 - \prod_{i=1}^{7}(1 - s_i). \tag{5.4}$$

5.3.2 Representation Level Stability Improves Known Interaction Recovery

To evaluate the ability how well each method identifies previously reported interactions, we consider a classification setup in which reported interactions in BioGRID and STRING constitute the positive class and the rest constitute the negative class. Specifically, for each phenotype, we obtain a list of genes following the two filtering steps described in Section 5.2.4. The potential gene-gene interactions include all pairwise combinations of genes in that list, and the positive cases include those previously reported. While BioGRID provides a binary indicator for the existence of an interaction, STRING assigns a confidence score between 0 and 1. We consider three thresholds for binarizing the STRING score: 0.4, 0.7, and 0.9. These thresholds reflect increasing confidence in the interaction being true, with 0.4 representing medium confidence, 0.7 high confidence, and 0.9 very high confidence. The predictions of iRF and US-iRF were their stability scores, while the predictions for MatrixEpistasis and US-MatrixEpistasis were one minus their p-value.

In our evaluation, we constructed datasets by pooling interactions from all phenotypes. The resulting dataset contains 1,748 potential interactions in total. Of these, 48 are labeled as positive in BioGRID. In STRING, the number of positives is 101, 187, and 315 for the respective confidence thresholds of 0.9, 0.7, and 0.4 (Appendix D.1). To ensure our evaluation is not influenced by the choice of a specific cutoff for the predictions, we compared

the performance using the Area Under the Receiver Operating Characteristic (ROC) curve (AUC-ROC).

The results are shown in Figure 5.1. For both the STRING and BioGRID databases, and across different confidence thresholds for STRING, Ultra-Stable iRF (US-iRF) consistently achieves the best AUC-ROC performance in classifying reported versus non-reported interactions.

Notably, US-MatrixEpistasis performs similarly to, or worse than, random selection of gene pairs, as indicated by AUC-ROC values close to 0.5. We believe this occurs because the Cauchy combination method for aggregating p-values is designed to maximize power under the assumption that all p-values are valid. This method behaves similarly to a minimum operator, but its poor performance here suggests that the p-values are not valid. We speculate that this invalidity may arise from misspecification of the linear interaction model.

An ablation study across all the single representation choices that is provided in Table D.2. US-iRF outperforms every single-representation variant across all databases, emphasizing the value in considering multiple representations. Notably, MatrixEpistasis outperformed every US-MatrixEpistasis (i.e., those that apply dimentionality reduction), while both version subsentially underperformed compared to iRF.

In Appendix D.3 we provide per-phenotype evaluation for AUROC scores. Across the 21 phenotypes, US-iRF achieves the highest AUROC most frequently in both BioGRID (6 out of 12 phenotypes that have at least one interaction in BioGRID) and for STRING (15 out of 20, 12 out of 17, and 9 out of 15 for the thresholds 0.4, 0.7, and 0.9, respectively).

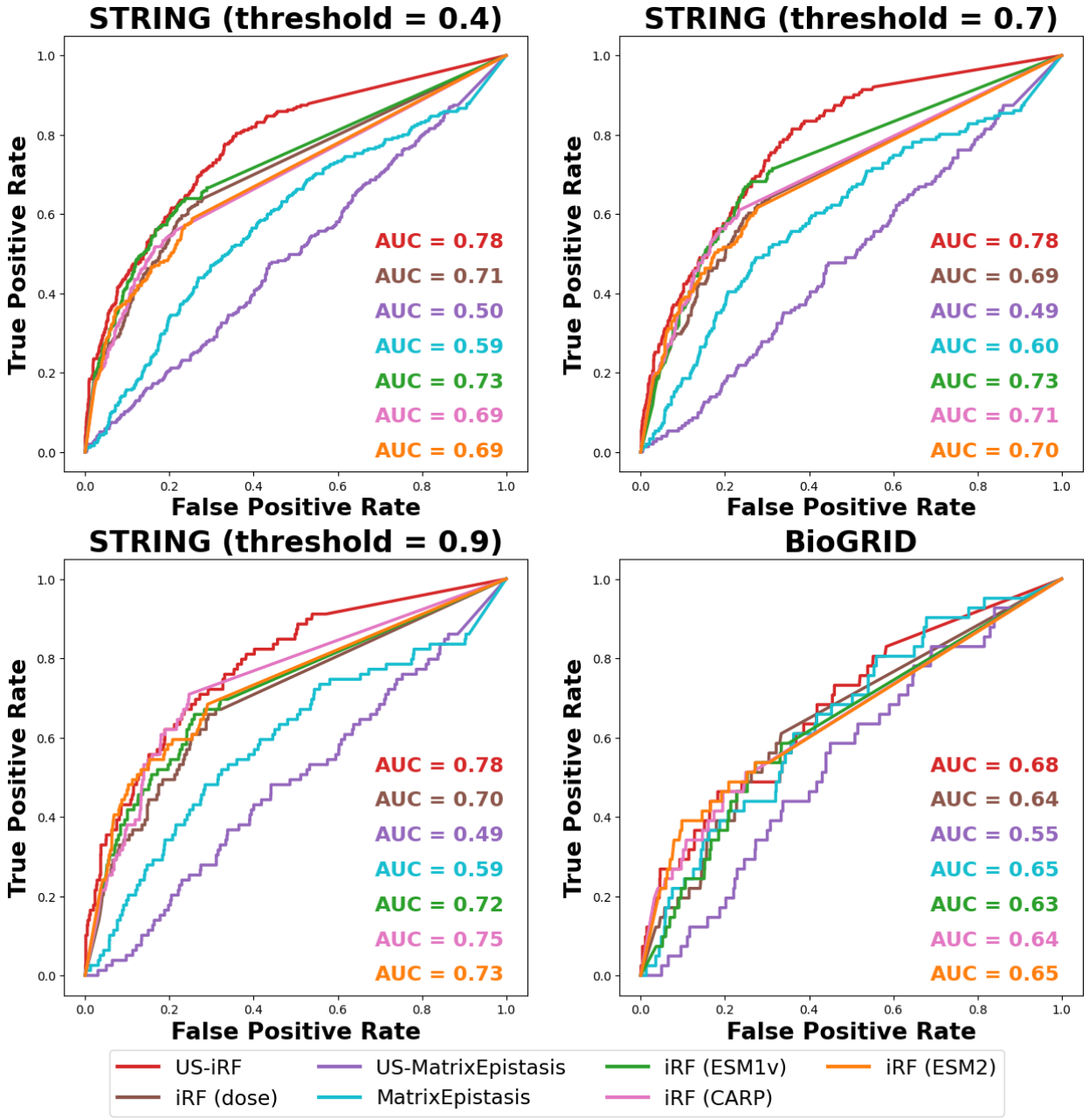

Figure 5.1: AUROC values for the classification problem of recovering known gene-gene interactions. Each panel represents a different set of positive interactions. The top row and bottom left represent the recovery of interaction from the STRING database with different thresholds. The bottom right panel represents the BioGRID database.

5.3.3 Ultra-Stable iRF Achieves High Precision for Top-Ranked Interactions

Obtaining a reliable ranking of discovered interactions is critical, under a constrained experimental budget. The ranking is used for selecting and prioritizing the most promising candidates for subsequent wet-lab experimental validation. While our previous analysis in Section 5.3.2 demonstrates that the Ultra-Stable procedure generally improves the ability to classify known from unknown interactions, it is also important to examine how the precision of these methods behaves specifically at high-confidence thresholds.

To this end, we report the precision for each method when restricted to the top percentiles of predicted interactions. Precision is defined as the number of true positive interactions divided by the total number of positive predictions in the selected set. We pick the percentiles as four equally spaced point on a grid between 0 and two times the percentage of positives at each dataset.

The results are reported in Figure 5.2. Indeed, US-iRF mostly outperforms all other methods. Across both STRING and BioGRID iRF methods that use PLM embeddings outperform the variant which uses the dose representation. Appendix D.4 provides full details of this analysis, including the number of discoveries and true discoveries for each database and method. Results for BioGRID are shown in Table D.7, while results for STRING are reported in Tables D.8 to D.10 for thresholds of 0.4, 0.7, and 0.9, respectively.

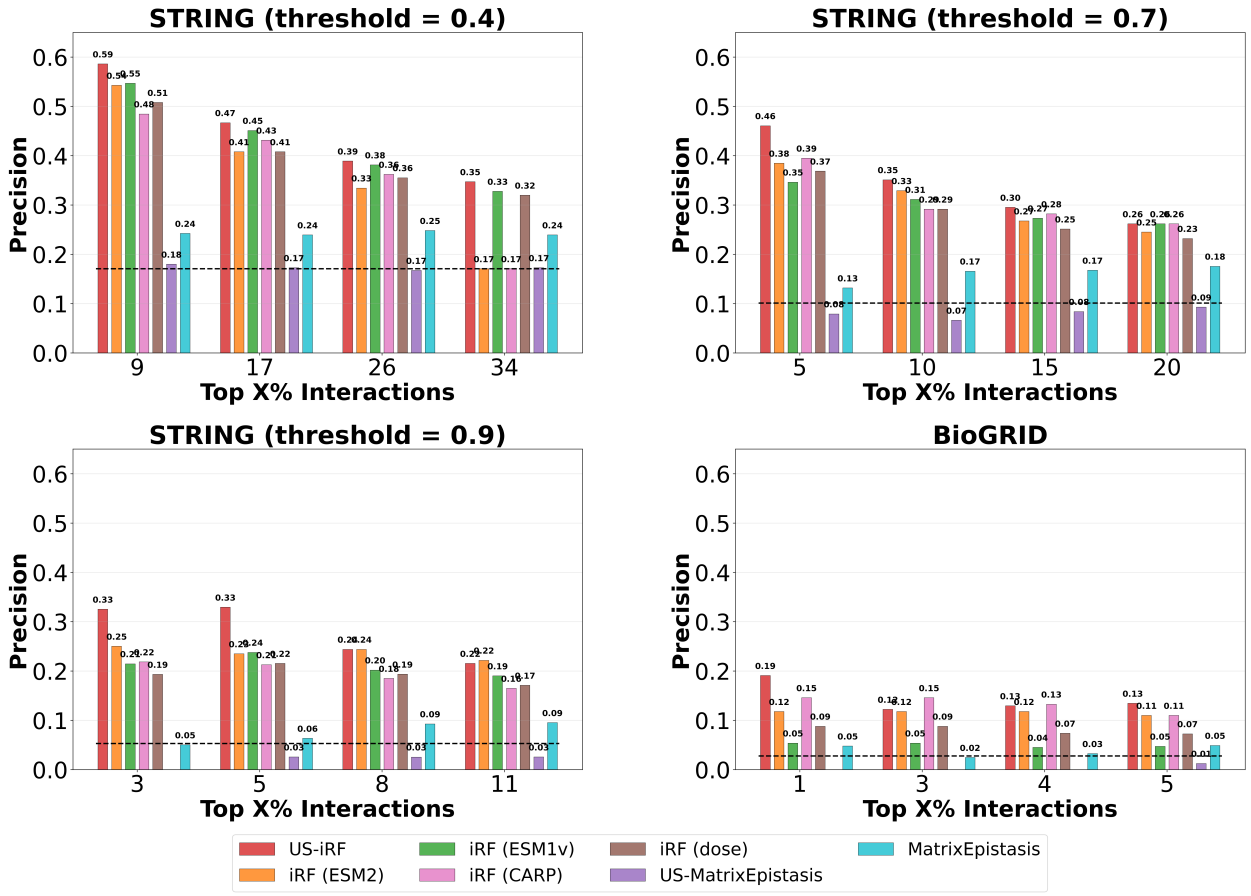

Figure 5.2: Precision (number of true positive interactions divided by the total number of predicted positives) across different thresholds and databases. Each panel represents a different set of positive interactions. The top row and bottom left represent the recovery of interaction from the STRING database with different thresholds. The bottom right panel represents the BioGRID database. Dashed lines indicate the proportion of positive interactions in each database.

5.4 Discussion

In this chapter, we build on the previous success of iRF for epistasis discovery in large-scale studies and propose methodological advancements to enhance its performance in the analysis of rare genetic variants. Our core contribution is to introduce a representation-level stability step, in which we follow the PCS framework for veridical data science and consider multiple representations of the SNP data.

Our evaluation, which was based on recovering known gene-gene interactions, demonstrated that this added stability significantly improved the performance of iRF. This was

especially evident in that 59% of the top 9% of interactions discovered using US-iRF had at least some previous evidence in the literature (STRING score ≥ 0.4).

While our results demonstrated impressive improvement of both US-iRF and iRF over MatrixEpistasis, and of US-iRF over iRF, it is important to emphasize that recovering known interactions is only a surrogate for discovering true causal relationships with respect to the phenotype of interest. It could very well be that interactions not present in BioGRID or STRING are in fact real and could be discovered if experimentally tested. In addition, it is possible that known gene-gene interactions within BioGRID or STRING do not affect the phenotypes of interest and could therefore be considered false positives in the context of a specific study. Therefore, a natural direction for future work is to experimentally verify some of the interactions proposed by iRF.

Therefore, a natural direction for future work is to experimentally verify some of the interactions proposed by iRF.

Bibliography

[1] Abhineet Agarwal et al. "PCS-UQ: Uncertainty Quantification via the Predictability-Computability-Stability Framework". In: *arXiv preprint arXiv:2505.08784* (2025).

[2] Sebastian Ament et al. "Unexpected improvements to expected improvement for bayesian optimization". In: *Advances in Neural Information Processing Systems* 36 (2024).

[3] Joshua D Backman et al. "Exome sequencing and analysis of 454,787 UK Biobank participants". In: *Nature* 599.7886 (2021), pp. 628–634.

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

[5] Jonathan B Baell and Georgina A Holloway. "New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays". In: *Journal of medicinal chemistry* 53.7 (2010), pp. 2719–2740.

[6] Maximilian Balandat et al. "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization". In: *Advances in Neural Information Processing Systems 33*. 2020. URL: http://arxiv.org/abs/1910.06403.

[7] Randall Balestriero and Richard Baraniuk. "Mad max: Affine spline insights into deep learning". In: *arXiv preprint arXiv:1805.06576* (2018).

[8] Randall Balestriero, Ahmed Imtiaz Humayun, and Richard Baraniuk. "On the geometry of deep learning". In: *arXiv preprint arXiv:2408.04809* (2024).

[9] Sumanta Basu et al. "iterative Random Forests to discover predictive and stable high-order interactions". In: *Proceedings of the National Academy of Sciences* (2018), p. 201711236.

[10] Merle Behr et al. "Learning epistatic polygenic phenotypes with Boolean interactions". In: *Plos one* 19.4 (2024), e0298906.

[11] Adi Ben-Israel. "The change-of-variables formula using matrix volume". In: *SIAM Journal on Matrix Analysis and Applications* 21.1 (1999), pp. 300–312.

[12] Helen M Berman et al. "The protein data bank". In: *Nucleic acids research* 28.1 (2000), pp. 235–242.

[13] Nicholas Bhattacharya et al. "Single layers of attention suffice to predict protein contacts". In: *Biorxiv* (2020), pp. 2020–12.

[14] James Bisbee. "Barp: Improving mister p using bayesian additive regression trees". In: *American Political Science Review* 113.4 (2019), pp. 1060–1065.

[15] Ron S Boger et al. "Functional protein mining with conformal guarantees". In: *Nature Communications* 16.1 (2025), p. 85.

[16] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[17] Leo Breiman et al. *Classification and regression trees*. Chapman and Hall/CRC, 1984.

[18] Nathan Brown et al. "GuacaMol: benchmarking models for de novo molecular design". In: *Journal of chemical information and modeling* 59.3 (2019), pp. 1096–1108.

[19] Colin J Carlson. "embarcadero: Species distribution modelling with Bayesian additive regression trees in r". In: *Methods in Ecology and Evolution* 11.7 (2020), pp. 850–858.

[20] Nicole Bohme Carnegie. "Comment: Contributions of Model Features to BART Causal Inference Performance Using ACIC 2016 Competition Data". In: *Statistical Science* 34.1 (2019), pp. 90–93. DOI: 10.1214/18-STS682. URL: https://doi.org/10.1214/18-STS682.

[21] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[22] Hugh Chipman, Edward George, and Robert McCulloch. "Bayesian ensemble learning". In: *Advances in neural information processing systems* 19 (2006).

[23] Hugh A Chipman, Edward I George, and Robert E McCulloch. "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* 4.1 (2010), pp. 266–298.

[24] Hugh A Chipman, Edward I George, and Robert E McCulloch. "Bayesian CART model search". In: *Journal of the American Statistical Association* 93.443 (1998), pp. 935–948.

[25] P. Cingolani et al. "A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3". In: *Fly* 6.2 (2012), pp. 80–92.

[26] P. Cingolani et al. "Using Drosophila melanogaster as a model for genotoxic chemical mutational studies with a new program, SnpSift". In: *Frontiers in Genetics* 3 (2012).

[27] Elizabeth T Cirulli and David B Goldstein. "Uncovering the roles of rare variants in common disease through whole-genome sequencing". In: *Nature Reviews Genetics* 11.6 (2010), pp. 415–425.

[28] Brian Clarke et al. "Integration of variant annotations using deep set networks boosts rare variant association testing". In: *Nature Genetics* (2024), pp. 1–10.

[29] Lorin Crawford et al. "Detecting epistasis with the marginal epistasis test in genetic mapping studies of quantitative traits". In: *PLoS genetics* 13.7 (2017), e1006869.

[30] Christian Dallago et al. "FLIP: Benchmark tasks in fitness landscape inference for proteins". In: *bioRxiv* (2021), pp. 2021–11.

[31] Petr Danecek et al. "Twelve years of SAMtools and BCFtools". In: *Gigascience* 10.2 (2021), giab008.

[32] Ingrid Daubechies et al. "Neural network approximation of refinable functions". In: *IEEE Transactions on Information Theory* 69.1 (2022), pp. 482–495.

[33] J. Dauparas et al. "Robust deep learning–based protein sequence design using ProteinMPNN". In: *Science* 378.6615 (2022), pp. 49–56. DOI: 10.1126/science.add2187. eprint: https://www.science.org/doi/pdf/10.1126/science.add2187. URL: https://www.science.org/doi/abs/10.1126/science.add2187.

[34] Vincent Dorie. *dbarts: Discrete Bayesian Additive Regression Trees Sampler*. R package version 0.9-22. 2022. URL: https://CRAN.R-project.org/package=dbarts.

[35] Vincent Dorie et al. "Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition". In: *Statistical Science* 34.1 (2019), pp. 43–68.

[36] David Eriksson et al. "Scalable global optimization via local Bayesian optimization". In: *Advances in neural information processing systems* 32 (2019).

[37] Clara Fannjiang and Ji Won Park. "Reliable algorithm selection for machine learning-guided design". In: *arXiv preprint arXiv:2503.20767* (2025).

[38] Clara Fannjiang et al. "Conformal prediction under feedback covariate shift for biomolecular design". In: *Proceedings of the National Academy of Sciences* 119.43 (2022), e2204569119.

[39] Peter I Frazier. "Bayesian optimization". In: *Recent advances in optimization and modeling of contemporary problems*. Informs, 2018, pp. 255–278.

[40] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[41] 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.

[42] Wenhao Gao et al. "Sample efficiency matters: a benchmark for practical molecular optimization". In: *Advances in neural information processing systems* 35 (2022), pp. 21342–21357.

[43] Andrew Gelman and Donald B Rubin. "Inference from iterative simulation using multiple sequences". In: *Statistical science* (1992), pp. 457–472.

[44] Rafael Gómez-Bombarelli et al. "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2 (2018), pp. 268–276.

[45] Donald P Green and Holger L Kern. "Modeling heterogeneous treatment effects in large-scale experiments using bayesian additive regression trees". In: *The annual summer meeting of the society of political methodology*. 2010, pp. 100–110.

[46] Donald P Green and Holger L Kern. "Modeling heterogeneous treatment effects in survey experiments with Bayesian additive regression trees". In: *Public opinion quarterly* 76.3 (2012), pp. 491–511.

[47] Kevin P Greenman, Ava P Amini, and Kevin K Yang. "Benchmarking uncertainty quantification for protein engineering". In: *PLOS Computational Biology* 21.1 (2025), e1012639.

[48] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. "Constrained Bayesian optimization for automatic chemical design using variational autoencoders". In: *Chemical science* 11.2 (2020), pp. 577–586.

[49] Peter Mørch Groth et al. "Kermut: Composite kernel regression for protein variant effects". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 29514–29565.

[50] Nate Gruver et al. "Effective surrogate models for protein design with bayesian optimization". In: *ICML Workshop on Computational Biology*. Vol. 198. 2021.

[51] P Richard Hahn, Vincent Dorie, and Jared S Murray. "Atlantic causal inference conference (acic) data analysis challenge 2017". In: *arXiv preprint arXiv:1905.09515* (2019).

[52] P Richard Hahn, Jared S Murray, and Carlos M Carvalho. "Bayesian regression tree models for causal inference: Regularization, confounding, and heterogeneous effects (with discussion)". In: *Bayesian Analysis* 15.3 (2020), pp. 965–1056.

[53] Jingyu He and P Richard Hahn. "Stochastic tree ensembles for regularized nonlinear regression". In: *Journal of the American Statistical Association* (2021), pp. 1–20.

[54] Jennifer Hill, Antonio Linero, and Jared Murray. "Bayesian additive regression trees: A review and look forward". In: *Annual Review of Statistics and Its Application* 7.1 (2020).

[55] Jennifer Hill and Yu-Sung Su. "Assessing lack of common support in causal inference using Bayesian nonparametrics: Implications for evaluating the effect of breastfeeding on children's cognitive outcomes". In: *The Annals of Applied Statistics* (2013), pp. 1386–1420.

[56] Jennifer L Hill. "Bayesian nonparametric modeling for causal inference". In: *Journal of Computational and Graphical Statistics* 20.1 (2011), pp. 217–240.

[57] Chloe Hsu et al. "Learning inverse folding from millions of predicted structures". In: *International conference on machine learning*. PMLR. 2022, pp. 8946–8970.

[58] Chloe Hsu et al. "Learning protein fitness models from evolutionary and assay-labeled data". In: *Nature biotechnology* 40.7 (2022), pp. 1114–1122.

[59] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. "Magnet: Uniform sampling from deep generative network manifolds without retraining". In: *International Conference on Learning Representations*. 2021.

[60] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. "Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10641–10650.

[61] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Junction tree variational autoencoder for molecular graph generation". In: *International conference on machine learning*. PMLR. 2018, pp. 2323–2332.

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

[63] Minoru Kanehisa. "The KEGG database". In: *'In silico'simulation of biological processes: Novartis Foundation Symposium 247*. Vol. 247. Wiley Online Library. 2002, pp. 91–103.

[64] Konrad J Karczewski et al. "Systematic single-variant and gene-based association testing of thousands of phenotypes in 394,841 UK Biobank exomes". In: *Cell Genomics* 2.9 (2022).

[65] Holger L Kern et al. "Assessing methods for generalizing experimental impact estimates to target populations". In: *Journal of research on educational effectiveness* 9.1 (2016), pp. 103–127.

[66] Diederik P Kingma. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[67] Aaron E Kornblith et al. "Predictability and stability testing to assess clinical decision instrument performance for children after blunt torso trauma". In: *PLOS digital health* 1.8 (2022), e0000076.

[68] Jannik Kossen et al. "Self-attention between datapoints: Going beyond individual input-output pairs in deep learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28742–28756.

[69] Mario Krenn et al. "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation". In: *Machine Learning: Science and Technology* 1.4 (2020), p. 045024.

[70] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. "Grammar variational autoencoder". In: *International conference on machine learning*. PMLR. 2017, pp. 1945–1954.

[71] Seunghun Lee et al. "Advancing Bayesian optimization via learning correlated latent space". In: *Advances in Neural Information Processing Systems* 36 (2024).

[72] Jing Lei et al. "Distribution-free predictive inference for regression". In: *Journal of the American Statistical Association* 113.523 (2018), pp. 1094–1111.

[73] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006. URL: http://scholar.google.com/scholar.bib?q=info:3wf9IU94tyMJ:scholar.google.com/&output=citation&hl=en&as_sdt=2000&ct=citation&cd=0.

[74] Francesca-Zhoufan Li et al. "Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes". In: *bioRxiv* (2024), pp. 2024–10.

[75] Francesca-Zhoufan Li et al. "Feature reuse and scaling: Understanding transfer learning with protein language models". In: *bioRxiv* (2024), pp. 2024–02.

[76] Mingchen Li et al. "Prosst: Protein language modeling with quantized structure and disentangled attention. bioRxiv". In: (2024).

[77] Zeming Lin et al. "Evolutionary-scale prediction of atomic-level protein structure with a language model". In: *Science* 379.6637 (2023), pp. 1123–1130.

[78] Antonio R Linero. "A review of tree-based Bayesian methods". In: *Communications for Statistical Applications and Methods* 24.6 (2017), pp. 543–559.

[79] Antonio R Linero and Yun Yang. "Bayesian regression tree ensembles that adapt to smoothness and sparsity". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 80.5 (2018), pp. 1087–1110.

[80] Christopher A Lipinski et al. "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings". In: *Advanced drug delivery reviews* 23.1-3 (1997), pp. 3–25.

[81] Yaowu Liu and Jun Xie. "Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures". In: *Journal of the American Statistical Association* 115.529 (2020), pp. 393–402.

[82] Yi Liu, Veronika Ročková, and Yuexi Wang. "Variable selection with ABC Bayesian forests". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 83.3 (2021), pp. 453–481.

[83] Benjamin J Livesey and Joseph A Marsh. "Updated benchmarking of variant effect predictors using deep mutational scanning". In: *Molecular systems biology* 19.8 (2023), e11474.

[84] Natalie Maus et al. "Local latent space Bayesian optimization over structured inputs". In: *Advances in neural information processing systems* 35 (2022), pp. 34505–34518.

[85] Zachary R McCaw et al. "An allelic-series rare-variant association test for candidate-gene discovery". In: *The American Journal of Human Genetics* 110.8 (2023), pp. 1330–1342.

[86] Joshua Meier et al. "Language models enable zero-shot prediction of the effects of mutations on protein function". In: *Advances in neural information processing systems* 34 (2021), pp. 29287–29303.

[87] Kevin P Murphy. "Conjugate Bayesian analysis of the Gaussian distribution". In: *def* $1.2\sigma 2$ (2007), p. 16.

[88] Jared S Murray. "Log-linear bayesian additive regression trees for multinomial logistic and count regression models". In: *Journal of the American Statistical Association* 116.534 (2021), pp. 756–769.

[89] David A Nix and Andreas S Weigend. "Estimating the mean and variance of the target probability distribution". In: *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*. Vol. 1. IEEE. 1994, pp. 55–60.

[90] Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. "Improving black-box optimization in VAE latent space using decoder uncertainty". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 802–814. URL: https://proceedings.neurips.cc/paper/2021/file/06fe1c234519f6812fc4c1baae25d6af-Paper.pdf.

[91] Pascal Notin et al. "Proteingym: Large-scale benchmarks for protein fitness prediction and design". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 64331–64379.

[92] Pascal Notin et al. "Proteingym: Large-scale benchmarks for protein fitness prediction and design". In: *Advances in Neural Information Processing Systems* 36 (2024).

[93] Pascal Notin et al. "Proteinnpt: Improving protein property prediction and design with non-parametric transformers". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 33529–33563.

[94] Pascal Notin et al. "TranceptEVE: Combining family-specific and family-agnostic models of protein sequences for improved fitness prediction". In: *bioRxiv* (2022), pp. 2022–12.

[95] Pascal Notin et al. "Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 16990–17017.

[96] Randal S Olson et al. "PMLB: a large benchmark suite for machine learning evaluation and comparison". In: *BioData mining* 10.1 (2017), pp. 1–13.

[97] Rose Oughtred et al. "The BioGRID database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions". In: *Protein Science* 30.1 (2021), pp. 187–200.

[98] R Kelley Pace and Ronald Barry. "Sparse spatial autoregressions". In: *Statistics & Probability Letters* 33.3 (1997), pp. 291–297.

[99] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[100] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[101] Gundula Povysil et al. "Rare-variant collapsing analyses for complex traits: guidelines and applications". In: *Nature Reviews Genetics* 20.12 (2019), pp. 747–759.

[102] Matthew T Pratola. "Efficient Metropolis–Hastings proposal mechanisms for Bayesian regression tree models". In: *Bayesian analysis* 11.3 (2016), pp. 885–911.

[103] Matthew T Pratola et al. "Heteroscedastic BART via multiplicative regression trees". In: *Journal of Computational and Graphical Statistics* 29.2 (2020), pp. 405–417.

[104] Alkes L Price et al. "Principal components analysis corrects for stratification in genome-wide association studies". In: *Nature genetics* 38.8 (2006), pp. 904–909.

[105] Roshan M Rao et al. "MSA transformer". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8844–8856.

[106] Zachary T Rewolinski and Bin Yu. "PCS Workflow for Veridical Data Science in the Age of AI". In: *arXiv preprint arXiv:2508.00835* (2025).

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

[108] Veronika Ročková and Stéphanie van der Pas. "Posterior concentration for Bayesian regression trees and forests". In: *The Annals of Statistics* 48.4 (2020), pp. 2108–2131.

[109] Veronika Ročková and Enakshi Saha. "On theory for BART". In: *The 22nd international conference on artificial intelligence and statistics*. PMLR. 2019, pp. 2839–2848.

[110] Joseph D Romano et al. "PMLB v1. 0: an open source dataset collection for benchmarking machine learning methods". In: *arXiv preprint arXiv:2012.00058* (2020).

[111] Joseph D Romano et al. "PMLB v1.0: an open source dataset collection for benchmarking machine learning methods". In: *arXiv preprint arXiv:2012.00058v2* (2021).

[112] Rajen Dinesh Shah and Nicolai Meinshausen. "Random intersection trees". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 629–654.

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

[114] Rodney A Sparapani et al. "Nonparametric survival analysis using Bayesian additive regression trees (BART)". In: *Statistics in medicine* 35.16 (2016), pp. 2741–2753.

[115] Jennifer E Starling et al. "BART with targeted smoothing: An analysis of patient-specific stillbirth risk". In: *The Annals of Applied Statistics* 14.1 (2020), pp. 28–50.

[116] Cathie Sudlow et al. "UK biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age". In: *PLoS medicine* 12.3 (2015), e1001779.

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

[118] Damian Szklarczyk et al. "The STRING database in 2023: protein–protein association networks and functional enrichment analyses for any sequenced genome of interest". In: *Nucleic acids research* 51.D1 (2023), pp. D638–D646.

[119] Nataša Tagasovska et al. "Antibody domainbed: Out-of-distribution generalization in therapeutic protein design". In: *arXiv preprint arXiv:2407.21028* (2024).

[120] 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* 33 (2020), pp. 11259–11272.

[121] Mihaly Varadi et al. "AlphaFold Protein Structure Database in 2024: providing structure coverage for over 214 million protein sequences". In: *Nucleic acids research* 52.D1 (2024), pp. D368–D375.

[122] Mihaly Varadi et al. "AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models". In: *Nucleic acids research* 50.D1 (2022), pp. D439–D444.

[123] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press, 2018.

[124] Christian Von Mering et al. "STRING: known and predicted protein–protein associations, integrated and transferred across organisms". In: *Nucleic acids research* 33.suppl_1 (2005), pp. D433–D437.

[125] Patrik Waldmann. "Genome-wide prediction using Bayesian additive regression trees". In: *Genetics Selection Evolution* 48.1 (2016), pp. 1–12.

[126] P. Walters. *rd filters*. Accessed: January 14, 2019. 2019. URL: https://github.com/PatWalters/rd_filters.

[127] Qianru Wang et al. "Epistasis regulates genetic control of cardiac hypertrophy". In: *Nature Cardiovascular Research* (2025), pp. 1–21.

[128] Wen-Hua Wei, Gibran Hemani, and Chris S Haley. "Detecting epistasis in human complex traits". In: *Nature Reviews Genetics* 15.11 (2014), pp. 722–733.

[129] Thierry Wendling et al. "Comparing methods for estimation of heterogeneous treatment effects using observational data from health care databases". In: *Statistics in medicine* 37.23 (2018), pp. 3309–3324.

[130] Scott A Wildman and Gordon M Crippen. "Prediction of physicochemical parameters by atomic contributions". In: *Journal of chemical information and computer sciences* 39.5 (1999), pp. 868–873.

[131] Yuhong Wu, Håkon Tjelmeland, and Mike West. "Bayesian CART: Prior specification and posterior simulation". In: *Journal of Computational and Graphical Statistics* 16.1 (2007), pp. 44–66.

[132] Jason Yang, Francesca-Zhoufan Li, and Frances H Arnold. "Opportunities and challenges for machine learning-assisted enzyme engineering". In: *ACS Central Science* 10.2 (2024), pp. 226–241.

[133] Jason Yang et al. "Active learning-assisted directed evolution". In: *Nature Communications* 16.1 (2025), p. 714.

[134] Kevin K Yang, Nicolo Fusi, and Alex X Lu. "Convolutions are competitive with transformers for protein sequence pretraining". In: *Cell Systems* 15.3 (2024), pp. 286–294.

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

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

[137] David S Yeager et al. "A national experiment reveals where a growth mindset improves achievement". In: *Nature* 573.7774 (2019), pp. 364–369.

[138] Bin Yu and Rebecca L Barter. *Veridical data science: The practice of responsible data analysis and decision making*. MIT Press, 2024.

[139] Bin Yu and Karl Kumbier. "Veridical data science". In: *Proceedings of the National Academy of Sciences* 117.8 (2020), pp. 3920–3929. DOI: 10.1073/pnas.1901326117. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1901326117. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1901326117.

[140] Bin Yu and Per Mykland. "Looking at Markov samplers through cusum path plots: a simple diagnostic idea". In: *Statistics and Computing* 8.3 (1998), pp. 275–286.

[141] Tianyu Zhang et al. "Application of Bayesian additive regression trees for estimating daily concentrations of pm2. 5 components". In: *Atmosphere* 11.11 (2020), p. 1233.

[142] Shijia Zhu and Gang Fang. "MatrixEpistasis: ultrafast, exhaustive epistasis scan for quantitative traits with covariate adjustment". In: *Bioinformatics* 34.14 (2018), pp. 2341–2348.

[143] Or Zuk et al. "Searching for missing heritability: designing rare variant association studies". In: *Proceedings of the National Academy of Sciences* 111.4 (2014), E455–E464.

[144] Or Zuk et al. “The mystery of missing heritability: Genetic interactions create phantom heritability”. In: *Proceedings of the National Academy of Sciences* 109.4 (2012), pp. 1193–1198.

Appendix A

Supplementary Material for Chapter 2

A.1 Proof Details for section 2.3

A.1.1 Proof of Lemma 3 (Conductance and likelihood ratio)

Proof. Let (v, τ) be the feature and threshold on which the root node splits in tree $\mathcal{T}$. Let $S = S(v, \tau)$ denote the set of tree structures whose root node splits on (v, τ). Then by assumption, we have $\mathcal{T} \in S$ and $\mathcal{T}', \mathcal{T}_0 \in S^c$, where $\mathcal{T}_0$ is the empty tree with only a root node. Assume for now that $\pi(S) \leq \frac{1}{2}$.

Because all possible moves for the Markov chain either add or remove a single terminal split, it is easy to see that S and S^c are connected by a single edge between $\mathcal{T}_0$ and $\mathcal{T}''$, the tree of depth 1 whose root node splits on (v, τ).

The conductance of S can thus be written as

$$\Phi(S) = \frac{\pi(\mathcal{T}_0)Q(\mathcal{T}''|\mathcal{T}_0)}{\pi(S)}. \tag{A.1}$$

Let $\psi(-|-)$ denote the transition kernel for the proposal step of the Metropolis-Hastings chain. We then have $\psi(\mathcal{T}''|\mathcal{T}_0) = \frac{1}{2md}$ (this comes from the probability of choosing a grow move multiplied by the probability of choosing the split (v, d).) The Metropolis-Hastings filter $\alpha(\mathcal{T}''|\mathcal{T}_0)$ is chosen precisely so that

$$\begin{aligned}\pi(\mathcal{T}_0)Q(\mathcal{T}''|\mathcal{T}_0) &= \pi(\mathcal{T}_0)\psi(\mathcal{T}''|\mathcal{T}_0)\alpha(\mathcal{T}''|\mathcal{T}_0) \\ &= \min\{\pi(\mathcal{T}_0)\psi(\mathcal{T}''|\mathcal{T}_0), \pi(\mathcal{T}'')\psi(\mathcal{T}_0|\mathcal{T}'')\}.\end{aligned}$$

Plugging in the formula for $\psi(\mathcal{T}''|\mathcal{T}_0)$ and continuing the equation gives

$$\pi(\mathcal{T}_0)Q(\mathcal{T}''|\mathcal{T}_0) \leq \frac{\pi(\mathcal{T}_0)}{2md}.$$

Next, notice that

$$\pi(S) = \sum_{\tilde{\mathcal{T}} \in S} \pi(\tilde{\mathcal{T}}) \geq \pi(\mathcal{T}).$$

As such, the conductance is bounded by a constant factor of the ratio of posterior probabilities as follows:

$$\Phi(S) \leq \frac{\pi(\mathcal{T}_0)}{2md\pi(\mathcal{T})}. \tag{A.2}$$

Since the posterior probability for any tree $\tilde{\mathcal{T}}$ can be written as

$$p(\tilde{\mathcal{T}}|\mathbf{X}, \boldsymbol{y}) = \frac{p(\tilde{\mathcal{T}})p(\boldsymbol{y}|\tilde{\mathcal{T}}, \mathbf{X})}{\sum_{\check{\mathcal{T}}\in\Omega} p(\check{\mathcal{T}})p(\boldsymbol{y}|\check{\mathcal{T}}, \mathbf{X})},$$

the right hand side of (A.2) is equal to

$$\frac{p(\mathcal{T}_0)}{2mdp(\mathcal{T})} \frac{p(\boldsymbol{y}|\mathcal{T}_0, \mathbf{X})}{p(\boldsymbol{y}|\mathcal{T}, \mathbf{X})}.$$

Since prior probabilities depend only on α, β, m, and d, the desired conclusion follows under the assumption that $\pi(S) \leq \frac{1}{2}$. If $\pi(S) > \frac{1}{2}$, we repeat the same string of calculations but with S replaced by S^c in (A.1) and with $\mathcal{T}$ replaced by $\mathcal{T}'$ thereafter.

□

A.1.2 Proof of Lemma 4 (Likelihood ratio and impurity decrease)

Proof. We first introduce some notation. Give the leaves of $\mathcal{T}$ an ordering, and denote them using $L_1, L_2, \ldots, L_b$. For $i = 1, \ldots, b$, let I_i denote the set of indices of the data points with $\boldsymbol{x}_j \in L_i$. Let n_i denote the count of data points in leaf i, i.e. $n_i := |I_i|$. Let $\mathcal{T}(x)$ denote the leaf node that contains an observation x, i.e. $\mathcal{T}(x_i) := L_j$ if $i \in I_j$. Finally, we will use C to denote absolute constants (independent of both the regression function f_0 and the tree $\mathcal{T}$) that may vary from line to line.

Step 1: Likelihood ratio to impurity decrease. By equation (42) in [87], we have

$$\begin{aligned}\frac{p(\boldsymbol{y}|\mathcal{T}_0, \mathbf{X})}{p(\boldsymbol{y}|\mathcal{T}, \mathbf{X})} &= \sqrt{\frac{\prod_{i=1}^b (n_i\sigma^2 + a\sigma^2)}{(n\sigma^2 + a\sigma^2)(a\sigma^2)^{b-1}}} \exp\left\{ \frac{\sigma^2}{2a\sigma^2(n\sigma^2 + a\sigma^2)} \cdot Z^2 - \sum_{i=1}^b \frac{\sigma^2}{2a\sigma^2(n_i\sigma^2 + a\sigma^2)} Z_i^2 \right\} \\ &= \sqrt{\frac{\prod_{i=1}^b (n_i + a)}{(n+a)a^{b-1}}} \exp\left\{ \frac{1}{2a\sigma^2} \left(\frac{1}{n+a} Z^2 - \sum_{i=1}^b \frac{1}{n_i + a} Z_i^2 \right) \right\}\end{aligned}$$

where $Z = \sum_{j=1}^n y_j$ $Z_i = \sum_{j \in I_j} y_j$ for $i = 1, \ldots, b$. We would like to convert the expression in the exponential into an impurity decrease by removing the appearance of a in the denominators. To do so, we compute the bound:

$$\begin{aligned}\left| \left(\frac{1}{n+a} Z^2 - \sum_{i=1}^b \frac{1}{n_i + a} Z_i^2 \right) - \left(\frac{1}{n} Z^2 - \sum_{i=1}^b \frac{1}{n_i} Z_i^2 \right) \right| &= \left| \frac{a}{n(n+a)} Z^2 - \sum_{i=1}^b \frac{a}{n_i(n_i + a)} Z_i^2 \right| \\ &\leq \frac{an^2K^2}{n(n+a)} + \sum_{i=1}^b \frac{an_i^2K^2}{n_i(n_i + a)} \\ &\leq a(b+1)K^2. \qquad \text{(A.3)}\end{aligned}$$

Note that this is indeed an impurity decrease as

$$\begin{aligned}\frac{1}{n} Z^2 - \sum_{i=1}^b \frac{1}{n_i} Z_i^2 &= \frac{1}{n} Z^2 - \sum_{j=1}^n y_j^2 - \sum_{i=1}^b \left(\frac{1}{n_i} Z_i^2 - \sum_{\boldsymbol{x}_j \in L_i} y_j^2 \right) \\ &= -\sum_{j=1}^n (y_j - \bar{y})^2 + \sum_{i=1}^b \sum_{\boldsymbol{x}_j \in L_i} (y_j - \bar{y}_{L_i})^2,\end{aligned}$$

where $\bar{y}_{L_i} = n_i^{-1} Z_i$.

Step 2: Concentration of impurity decrease conditioned on counts. Condition on $n_1, \ldots, n_b$ and denote $\tilde{q}_i = \frac{n_i}{n}$ for $i = 1, \ldots, b$. The impurity decrease may be written as a variance:

$$-\frac{1}{n} Z^2 + \sum_{i=1}^b \frac{1}{n_i} Z_i^2 = -n \left(\sum_{i=1}^b \tilde{q}_i \bar{y}_{L_i}^2 - \left(\sum_{i=1}^b \tilde{q}_i \bar{y}_{L_i} \right)^2 \right).$$

We may rewrite the expression in parentheses on the right in matrix form:

$$\sum_{i=1}^{b} \tilde{q}_i \bar{y}_{L_i}^2 - \left(\sum_{i=1}^{b} \tilde{q}_i \bar{y}_{L_i} \right)^2 = \boldsymbol{w}^T \left(\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right) \boldsymbol{w}$$

where $w_i = \sqrt{\tilde{q}_i} \bar{y}_{L_i}$ for $i = 1, \ldots, b$.

Denote

$$\tilde{\Delta} = \sum_{i=1}^{b} \tilde{q}_i \mathbb{E}\{f_0(\boldsymbol{x}) | \boldsymbol{x} \in L_i\}^2 - \left(\sum_{i=1}^{b} \tilde{q}_i \mathbb{E}\{f_0(\boldsymbol{x}) | \boldsymbol{x} \in L_i\} \right)^2,$$

and notice that we may also write

$$\tilde{\Delta} = \mathbb{E}\{\boldsymbol{w}\}^T \left(\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right) \mathbb{E}\{\boldsymbol{w}\}.$$

We therefore have

$$\begin{aligned} &\left(\sum_{i=1}^{b} \tilde{q}_i \bar{y}_{L_i}^2 - \left(\sum_{i=1}^{b} \tilde{q}_i \bar{y}_{L_i} \right)^2 \right) - \tilde{\Delta} \\ &= (\boldsymbol{w} - \mathbb{E}\{\boldsymbol{w}\})^T \left(\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right) (\boldsymbol{w} - \mathbb{E}\{\boldsymbol{w}\}) + 2(\boldsymbol{w} - \mathbb{E}\{\boldsymbol{w}\})^T \left(\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right) \mathbb{E}\{\boldsymbol{w}\}. \end{aligned} \tag{A.4}$$

We shall bound these two terms separately. To this end, we compute

$$\begin{aligned} \left\| \mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right\|_F^2 &= \|\mathbf{I}_b\|_F^2 - 2\mathrm{Tr}\left(\sqrt{\tilde{\boldsymbol{q}}}^T \mathbf{I}_b \sqrt{\tilde{\boldsymbol{q}}} \right) + \left\| \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right\|_F^2 \\ &= b - 1. \end{aligned}$$

Similarly,

$$\|\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T\|_{op} = 1.$$

Note also that, by Hoeffding's lemma [123], $\boldsymbol{w}$ has independent sub-Gaussian entries, each with squared sub-Gaussian norm at most

$$\left\| \sqrt{\tilde{q}_i} \bar{y}_{L_i} \right\|_{\psi_2}^2 \leq \frac{\tilde{q}_i K^2}{n_i} = \frac{K^2}{n}.$$

Applying the Hanson-Wright inequality [123], we therefore get

$$(\boldsymbol{w} - \mathbb{E}\{\boldsymbol{w}\})^T \left(\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}} \sqrt{\tilde{\boldsymbol{q}}}^T \right) (\boldsymbol{w} - \mathbb{E}\{\boldsymbol{w}\}) \leq \frac{CbK^2 \sqrt{\log n}}{n} \tag{A.5}$$

with probability at least $1 - 1/4n$.

Meanwhile, we have

$$\begin{aligned}\|\mathbb{E}\{\boldsymbol{w}\}\|_2^2 &= \sum_{i=1}^{b} \tilde{q}_i (\mathbb{E}\{f_0(\boldsymbol{x}) | \boldsymbol{x} \in L_i\})^2 \\ &\leq K^2.\end{aligned}$$

Hence, using Hoeffding's inequality, we have

$$(\boldsymbol{w} - \mathbb{E}\{\boldsymbol{w}\})^T \left(\mathbf{I}_b - \sqrt{\tilde{\boldsymbol{q}}}\sqrt{\tilde{\boldsymbol{q}}}^T\right)\mathbb{E}\{\boldsymbol{w}\} \leq CK^2\sqrt{\frac{\log n}{n}} \tag{A.6}$$

with probability at least $1 - 4/n$. Plugging (A.5) and (A.6) into (A.4) gives us

$$\left(\sum_{i=1}^{b} \tilde{q}_i \bar{y}_{L_i}^2 - \left(\sum_{i=1}^{b} \tilde{q}_i \bar{y}_{L_i}\right)^2\right) - \tilde{\Delta} \leq CK^2\sqrt{\log n}\left(\frac{b}{n} + \frac{1}{\sqrt{n}}\right) \tag{A.7}$$

with probability at least $1 - 2/n$.

Step 3: Decondition on counts. Using the law of total variance, write

$$\begin{aligned}\Delta &= \mathrm{Var}\{f_0(\boldsymbol{x})\} - \mathbb{E}\{\mathrm{Var}\{f_0(\boldsymbol{x})|\mathcal{T}(\boldsymbol{x})\}\} \\ &= \mathrm{Var}\{\mathbb{E}\{f_0(\boldsymbol{x}) \mid \mathcal{T}(\boldsymbol{x})\}\} \\ &= \sum_{i=1}^{b} q_i \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\}^2 - \left(\sum_{i=1}^{b} q_i \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\}\right)^2\end{aligned}$$

where $q_i = \mathbb{P}\{y \in L_i\}$ for $i = 1, \ldots, b$. Since $n\tilde{\boldsymbol{q}}$ is a multinomial distribution with parameters n and $\boldsymbol{q}$, we may apply Hoeffding's inequality two more times to get

$$\left|\sum_{i=1}(\tilde{q}_i - q_i)\mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\}\right| \leq CK\sqrt{\frac{\log n}{n}}$$

and

$$\left|\sum_{i=1}(\tilde{q}_i - q_i)\mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\}^2\right| \leq CK^2\sqrt{\frac{\log n}{n}} \tag{A.8}$$

jointly with probability at least $1 - 1/2n$. On this event, we also have

$$\begin{aligned}
&\left| \left(\sum_{i=1}^{b} \tilde{q}_i \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\} \right)^2 - \left(\sum_{i=1}^{b} q_i \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\} \right)^2 \right| \\
&= \left| \sum_{i=1}^{b} (\tilde{q}_i - q_i) \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\} \right| \cdot \left| \sum_{i=1}^{b} (\tilde{q}_i + q_i) \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\} \right| \\
&\leq 2K \left| \sum_{i=1}^{b} (\tilde{q}_i - q_i) \mathbb{E}\{f_0(\boldsymbol{x}) \mid \boldsymbol{x} \in L_i\} \right| \\
&\leq CK^2 \sqrt{\frac{\log n}{n}}. \qquad \text{(A.9)}
\end{aligned}$$

Using (A.8) and (A.9), we get

$$\left| \tilde{\Delta} - \Delta \right| \leq CK^2 \sqrt{\frac{\log n}{n}}.$$

Conditioning on this event and further conditioning on the $1 - 1/2n$ event guaranteeing (A.4) then gives us:

$$\begin{aligned}
\left| \left(\frac{1}{n+a} Z^2 - \sum_{i=1}^{b} \frac{1}{n_i + a} Z_i^2 \right) + n\Delta \right| &\leq a(b+1)K^2 + CK^2 \sqrt{\log n} \left(b + \sqrt{n} \right) + CK^2 \sqrt{n \log n} \\
&\leq CK^2 \sqrt{\log n} \left(\sqrt{n} + b \right)
\end{aligned}$$

since it only makes sense to choose $a \leq 1$.

Finally, we have

$$\begin{aligned}
\log \sqrt{\frac{\prod_{i=1}^{b} (n_i + a)}{(n+a) a^{b-1}}} &= \frac{1}{2} \sum_{i=1}^{b} \log(n_i + a) - \frac{1}{2} \log(n+a) - (b-1) \log a \\
&\leq b \log(n/a).
\end{aligned}$$

□

A.2 Additional Simulation Results

A.2.1 Simplified BART RMSE Kernel Density

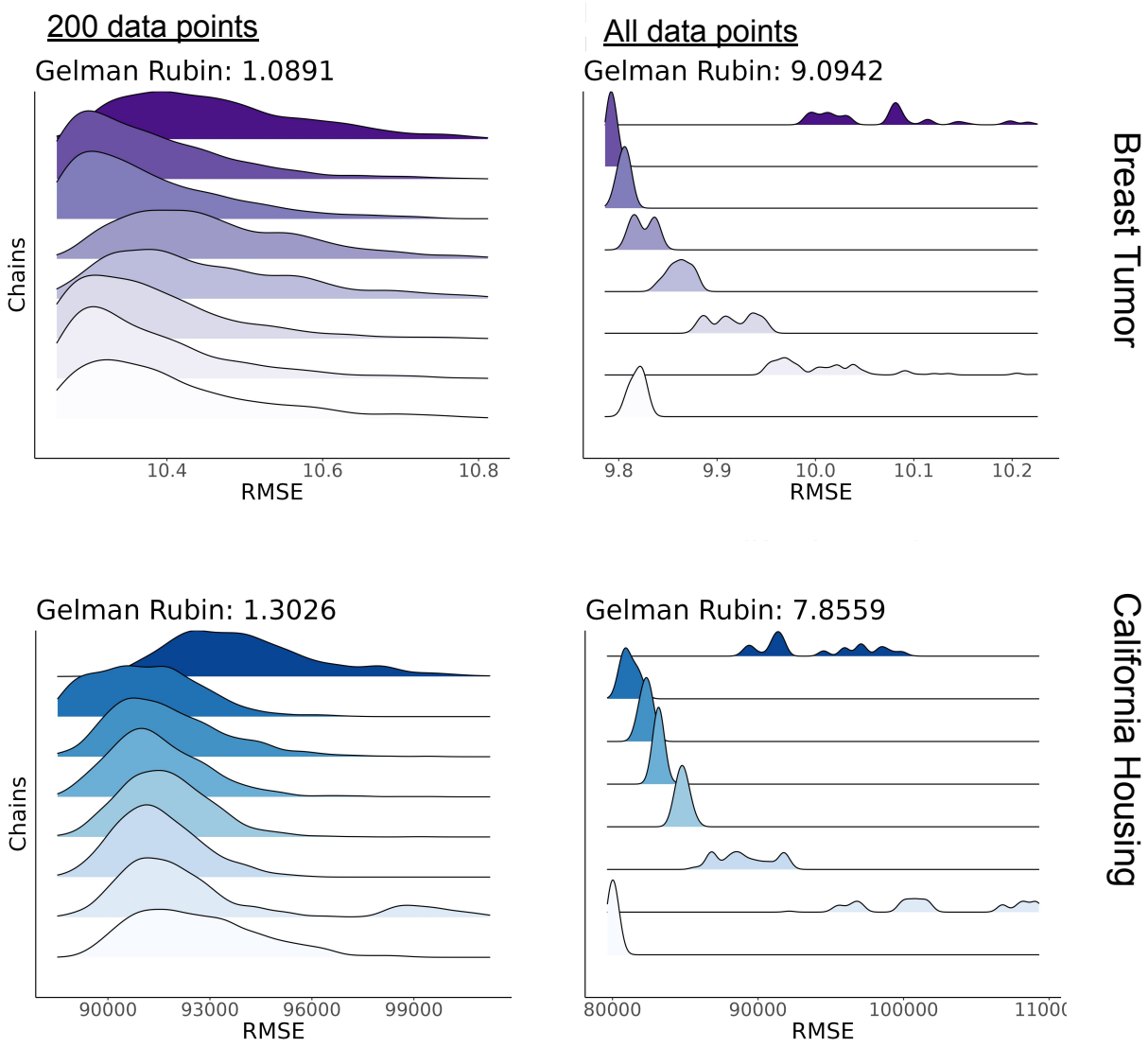

Figure A.1: Kernel density plots of simplified BART RMSE values across different chains and data sets and sample sizes. The number of data points used increases from the left to right column. Different rows correspond to different datasets.

A.2.2 Cusum Diagnostics for BART and simplified BART

In this section we visualize the BART and simplified BART mixing using the cusum path plot proposed by [140]. We now describe the cusum plot formally: Suppose we have L samples from J different MCMC chains obtained after a burn-in period. Let x_{ij} denote the RMSE value of the ith sample from the jth chain, $\bar{x}_j$ denote the mean RMSE sample from the jth chain, and let $S_t^{(j)} = \sum_{i=1}^{t}(x_{ij} - \bar{x}_j)$ be the cumulative sum of deviations from the mean. In a cusum plot, we plot $S_t^{(j)}$ against t. A "hairy" cusum plot, in which the cumulative sum varies randomly around a mean of zero, represents fast mixing. A smooth cusum plot, represents shifts in the sampling process, and is an indicating of slow mixing. In our case, a smooth cusum path would translate into a sub-sequence of tree functions within a given chain, whose RMSE value is higher or lower than the chain average.

figures A.2 and A.3 show cusum plot for BART and simplified BART respectively, across 8 different chains.

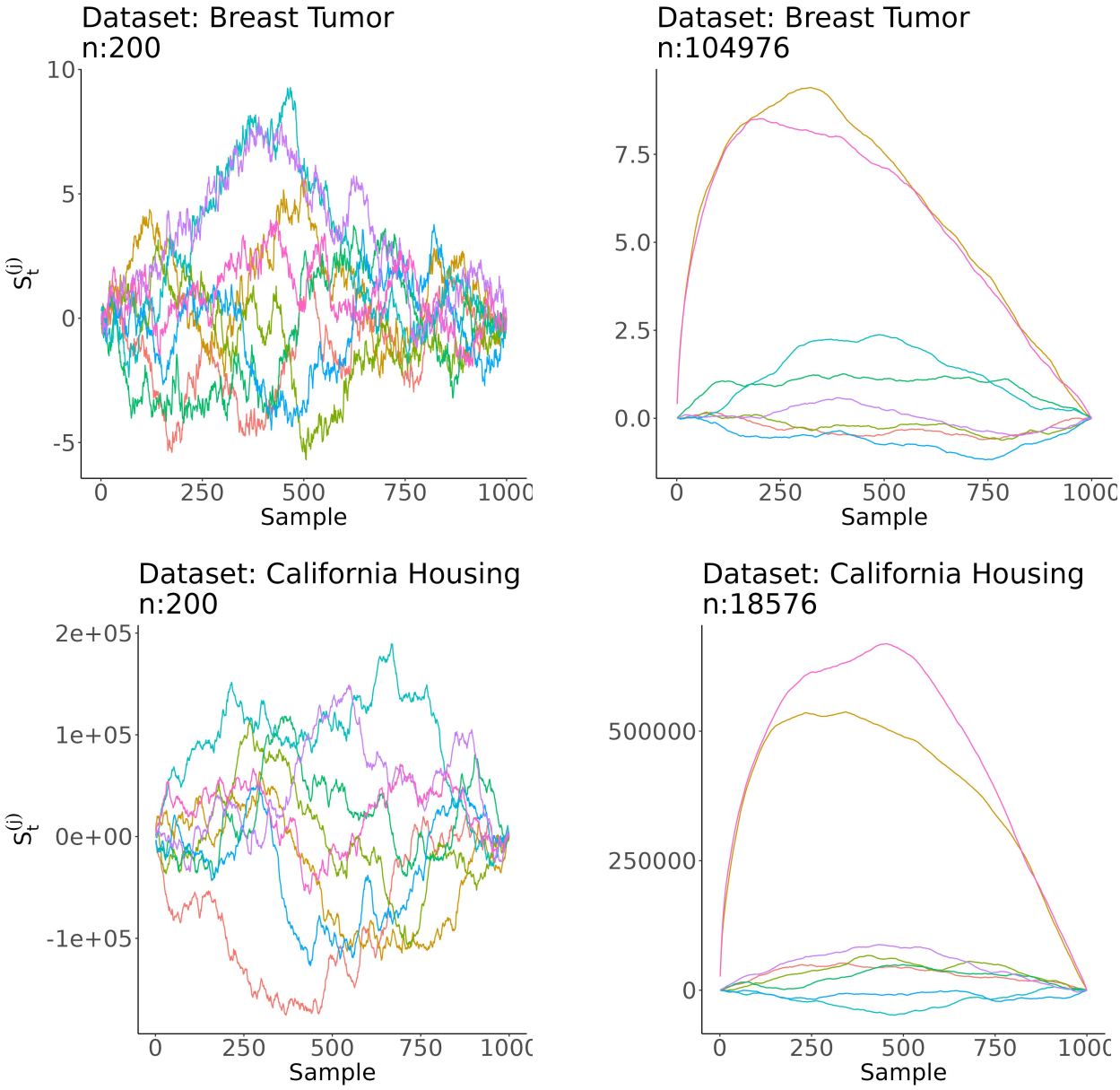

Figure A.2: BART cusum path plot become smoother as more data points are used. $S_t^{(j)}$ is plotted against t for 8 different chains. The number of data points used increases from the left to right column. Different rows correspond to different datasets, and different colors correspond to different chains.

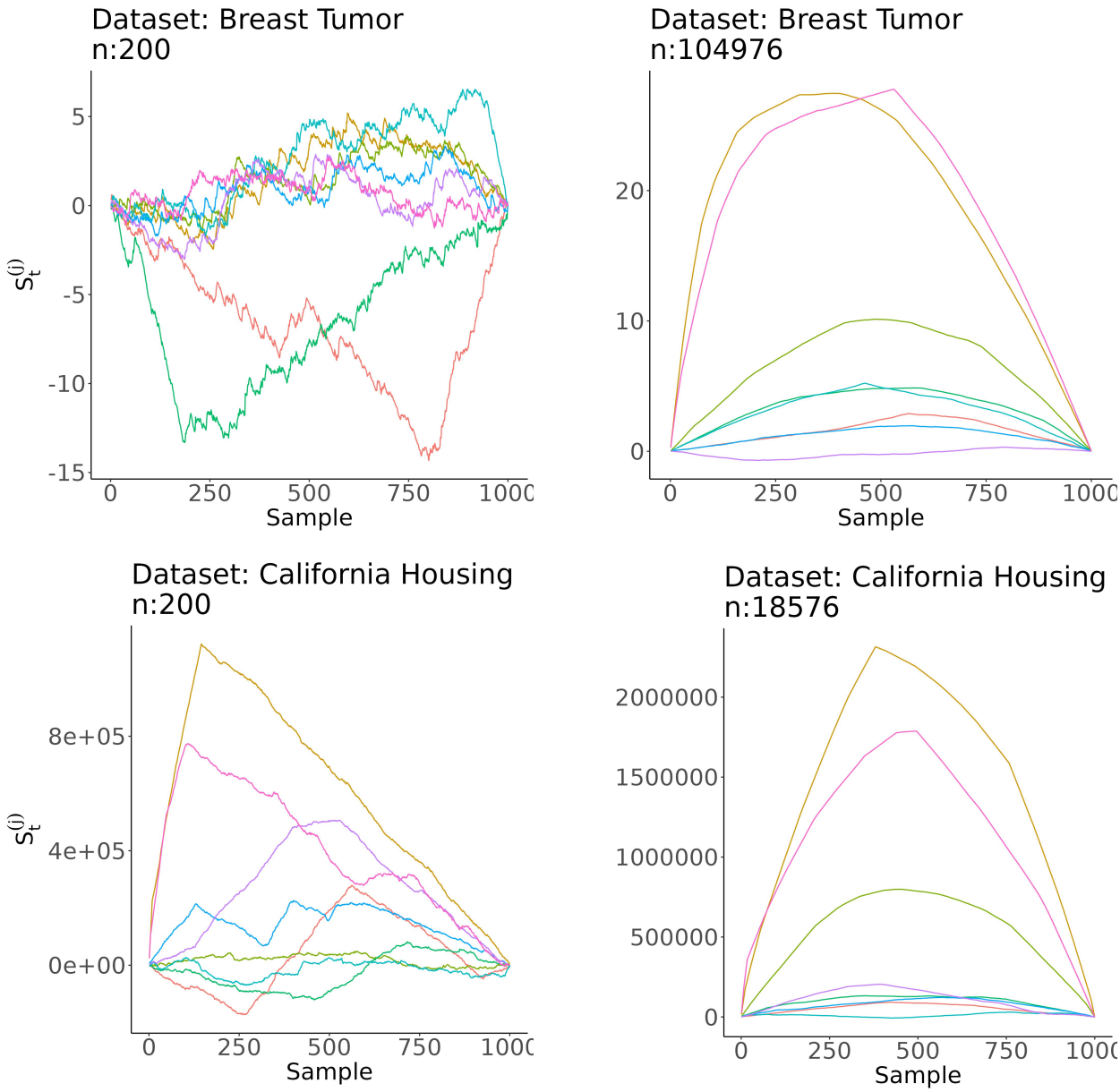

Figure A.3: Simplified BART cusum path plot become smoother as more data points are used. $S_t^{(j)}$ is plotted against t for 8 different chains. The number of data points used increases from the left to right column. Different rows correspond to different datasets, and different colors correspond to different chains.

A.2.3 Reversing the Root Split

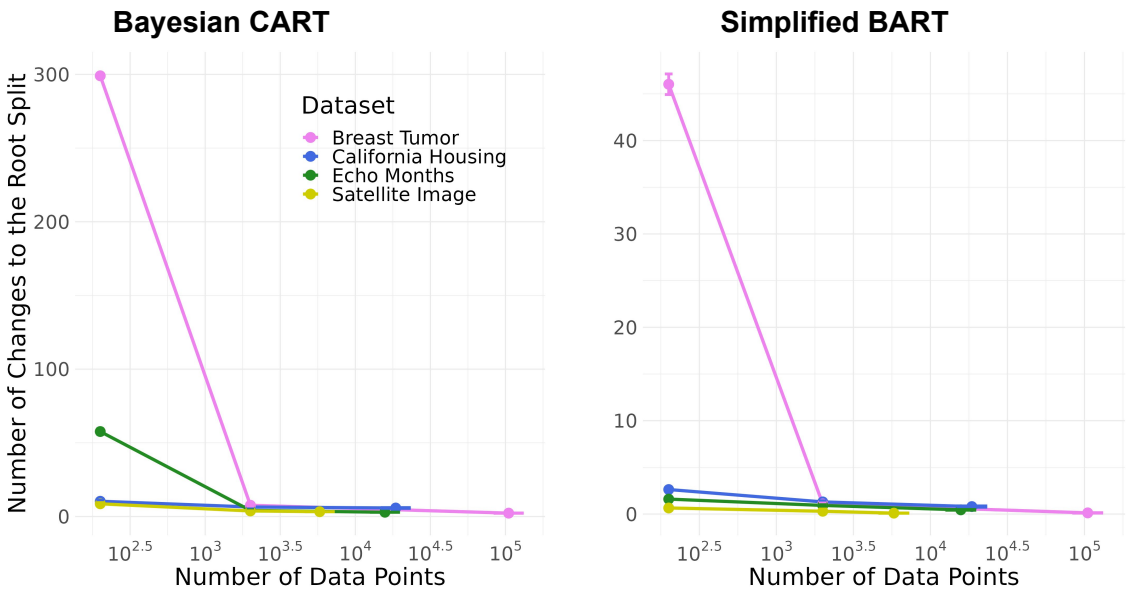

Figure A.4: The number of MCMC iterations during which the feature on which the root node splits changes is small, and decreases as the number of training data points increases. Right panel shows the results for B-CART and left for simplified BART . Error bars are calculated over 160 different runs using the same training data.

A.2.4 Simplified BART Root Node Split Indices

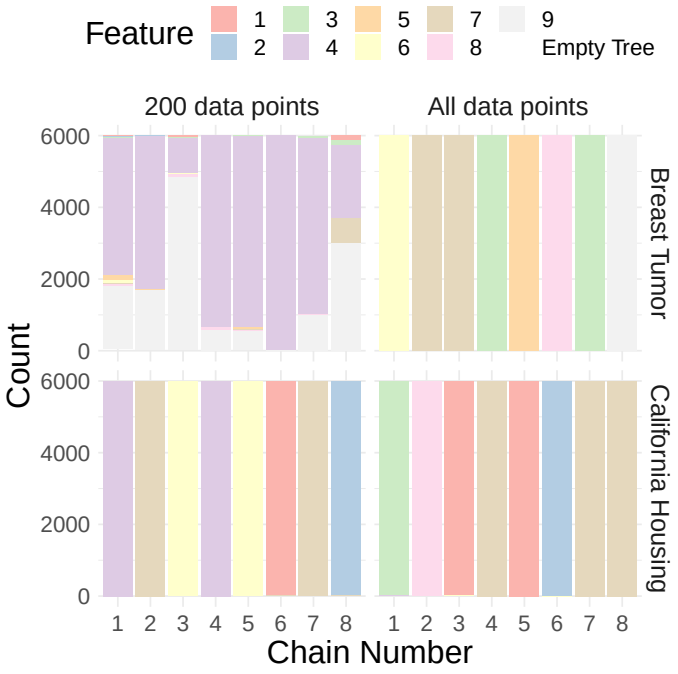

Figure A.5: Number of iterations on which the root node splits on each feature for different chains on several datasets. When n is large, almost all MCMC samples for a single chain of simplified BART have a root split on the same feature, whose index varies across chains. The different colors correspond to the different features, and empty tree refers to a single leaf (no splits).

Appendix B

Supplementary Material for Chapter 3

B.1 Proofs

Lemma 10. *Let f_θ be a DGN as defined in eq. (3.8) and assume that f_θ can be expressed as a CPA (eq. (3.5)) and is inevitable, then*

$$Jf_\theta^{-1}(\boldsymbol{x}) = \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger \right)^T, \tag{B.1}$$

where $\boldsymbol{A}_\omega^\dagger$ is the Moore–Penrose inverse of the slope matrix, at the knot whose image constrains $\boldsymbol{x}$, and

$$\boldsymbol{B}_i = \left(diag\left(\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_1}, \ldots, \frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_{\mathrm{D}}} \right), -\mathbf{1}\frac{1}{c_{\boldsymbol{z}}^{(i)}} \right)^T. \tag{B.2}$$

Proof. First we write

$$f_\theta(\boldsymbol{z}) = \mathrm{Softmax}_+(\ell_\theta(\boldsymbol{z})), \tag{B.3}$$

Where $\mathrm{Softmax}_+$ is the extension of the column wise Softmax function to include the normalizing constants. Specifically, for L by D $\ell_\theta(\boldsymbol{z})$ matrix, we have

$$\mathrm{Softmax}_+(\ell_\theta(\boldsymbol{z})) = \left(\boldsymbol{p}_{\boldsymbol{z}}^{(1)}, (c_{\boldsymbol{z}}^{(1)})^{-1}, \ldots, \boldsymbol{p}_{\boldsymbol{z}}^{(L)}, (c_{\boldsymbol{z}}^{(L)})^{-1}\right) = \boldsymbol{x}_{\boldsymbol{z}}, \tag{B.4}$$

with $\boldsymbol{p}_{\boldsymbol{z}}^{(i)} = (\frac{e^{\ell_\theta(\boldsymbol{z})_{1i}}}{c_{\boldsymbol{z}}^{(i)}})$, and $c_{\boldsymbol{z}}^{(i)} = \sum_{j=1}^{\mathrm{D}} \exp(\ell_\theta(\boldsymbol{z})_{ji})$.
Next,

$$f_\theta^{(-1)}(\boldsymbol{x}) = \ell_\theta^{-1}(\mathrm{Softmax}_+^{-1}(\boldsymbol{x})) \tag{B.5}$$

A direct calculation yields,

$$\text{Softmax}_+^{-1}(\boldsymbol{x}) = \left(\log(\boldsymbol{p}_{\boldsymbol{z}}^{(1)}) + \log(c_{\boldsymbol{z}}^{(1)}), \ldots, \log(\boldsymbol{p}_{\boldsymbol{z}}^{(L)}) + \log(c_{\boldsymbol{z}}^{(L)})\right). \tag{B.6}$$

As we assume ℓ_θ is bijective and can be written as

$$\ell_\theta(\boldsymbol{z}) = \sum_{\omega\in\Omega} (\boldsymbol{A}_\omega \boldsymbol{z} + \boldsymbol{b}_\omega)\, 1_{\boldsymbol{z}\in\omega}, \tag{B.7}$$

we have that

$$\ell_\theta^{-1}(\text{Softmax}_+^{-1}(\boldsymbol{x})) = (\text{Softmax}_+^{-1}(\boldsymbol{x}) - \boldsymbol{b}_\omega)\boldsymbol{A}_\omega^\dagger. \tag{B.8}$$

Lastly, as

$$\frac{\partial \text{Softmax}_+^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \boldsymbol{B}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{B}_{\text{L}} \end{bmatrix}, \tag{B.9}$$

for

$$\boldsymbol{B}_i = \left(\text{diag}\left(\frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_1}, \ldots, \frac{1}{(\boldsymbol{p}_{\boldsymbol{z}}^{(i)})_{\text{D}}}\right), -\boldsymbol{1}\frac{1}{c_{\boldsymbol{z}}^{(i)}}\right)^T. \tag{B.10}$$

we obtain the final result. □

Proof of theorem 8. First, we note that by our invertibility assumption we have that $\mathbb{P}(\boldsymbol{x} \in W) = \mathbb{P}(\boldsymbol{z} \in f_\theta^{(-1)}(W))$. We then proceed with a direct calculation

$$\begin{aligned}
\mathbb{P}(\boldsymbol{x} \in W) &= \mathbb{P}(\boldsymbol{z} \in f_\theta^{(-1)}(W)) && \text{(B.11)}\\
&= \sum_{\omega\in\Omega} \mathbb{P}(\boldsymbol{z} \in (f_\theta^{(-1)}(W) \cap \omega)) && \text{(B.12)}\\
&= \sum_{\omega\in\Omega} \int_{f_\theta^{(-1)}(W)\cap\omega} f_{\boldsymbol{z}}(\boldsymbol{z}) d\boldsymbol{z} && \text{(B.13)}\\
&= \sum_{\omega\in\Omega} \int_{W\cap f_\theta(\omega)} f_{\boldsymbol{z}}(f_\theta^{(-1)}(\boldsymbol{x})) \sqrt{\det\left(J f_\theta^{(-1)}(\boldsymbol{x}) J f_\theta^{(-1)}(\boldsymbol{x})^T\right)} d\boldsymbol{x} && \text{(B.14)}\\
&= \int_W \sum_{\omega\in\Omega} f_{\boldsymbol{z}}(f_\theta^{(-1)}(\boldsymbol{x})) \sqrt{\det\left(J f_\theta^{(-1)}(\boldsymbol{x}) J f_\theta^{(-1)}(\boldsymbol{x})^T\right)} 1_{\{\boldsymbol{x}\in f_\theta(\omega)\}} d\boldsymbol{x}. && \text{(B.15)}
\end{aligned}$$

Using lemma 10, we get that the volume element is

$$Jf_\theta^{(-1)}(\boldsymbol{x})Jf_\theta^{(-1)}(\boldsymbol{x})^T = \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger\right)^T \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger\right) \tag{B.16}$$

$$\left((\boldsymbol{A}_\omega^\dagger)^T \begin{bmatrix} \boldsymbol{B}_1^T & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}}^T, \end{bmatrix}\right) \left(\begin{bmatrix} \boldsymbol{B}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{B}_{\mathrm{L}} \end{bmatrix} \boldsymbol{A}_\omega^\dagger\right) \tag{B.17}$$

$$= \sum_{i=1}^{\mathrm{L}} (\boldsymbol{A}_i^\dagger)^T (\boldsymbol{B}_i)^T \boldsymbol{B}_i \boldsymbol{A}_i^\dagger, \tag{B.18}$$

where $\boldsymbol{A}_i^\dagger = \left(\boldsymbol{A}_\omega^{(1)}, \ldots, \boldsymbol{A}_\omega^{(L)}\right)^\dagger_{(i\cdot \mathrm{D}):(i+1\cdot \mathrm{D}).}$. □

Lemma 11. *Let $f : \mathcal{Z} \to \mathcal{X}$ be a function and define $f^\dagger : \mathcal{X} \to \mathcal{Z}$ as $f^\dagger(x) \in \{z : f(z) = x\}$. Let μ be the Lebesgue measure and assume that $\mu(\{z; \exists z' \text{ s.t. } f(z) = f(z')\}) = 0$, then for every $B \subseteq \mathcal{Z}$ we have*

$$\mu(\{z; f(z) \in B\}) = \mu(f^\dagger(B)) \tag{B.19}$$

Proof. We proceed with direct calculation

$$\mu(\{z; f(z) \in B\}) \leq \mu(f^\dagger(B)) + \mu(\{z; \exists z' \text{ s.t. } f(z) = f(z')\}) \tag{B.20}$$

Now assume that $\mu(\{z; \exists z' \text{ s.t. } f(z) = f(z')\}) = 0$, we have that $\mu(\{z; f(z) \in B\}) \leq \mu(f^\dagger(B))$. The other direction follows immediately from the definition of $f^\dagger$. □

lemma 11 implies that eq. (B.14) can still hold under the assumption that $\mu(\{z; \exists z' \text{ s.t. } f_\theta(z) = f_\theta(z')\}) = 0$.

B.2 Ablation studies

B.2.1 TuRBO hyper-parameters

We provide an ablation study of the initial length and success and failure tolerances. For each setup, the values reported in the main paper are highlighted in bold.

Table B.1: Ablation study for the initial length, success tolerance and failure tolerance for the TuRBO method. Average across 10 independent runs, of the top 20 best values across datasets, architectures. Column names indicate the length/success/fail values. Results from the main paper are in bold.

	Architecture	β	0.8/10/10	0.8/10/2	0.8/2/10	0.8/2/2	1.6/10/10	1.6/10/2	1.6/2/10	1.6/2/2
Expressions	GRU	0.05	-2.03 (0.12)	**-2.1 (0.11)**	-2.0 (0.09)	-2.0 (0.11)	-1.93 (0.09)	-1.92 (0.08)	-1.91 (0.06)	-1.93 (0.07)
		0.1	-2.18 (0.15)	**-2.14 (0.18)**	-2.11 (0.2)	-2.14 (0.2)	-1.98 (0.13)	-2.05 (0.12)	-2.0 (0.17)	-2.09 (0.22)
		1	-1.39 (0.12)	**-1.34 (0.05)**	-1.56 (0.11)	-1.61 (0.11)	-1.36 (0.08)	-1.42 (0.11)	-1.45 (0.12)	-1.38 (0.12)
	LSTM	0.05	-1.45 (0.09)	**-1.59 (0.13)**	-1.63 (0.12)	-1.49 (0.1)	-1.46 (0.08)	-1.52 (0.08)	-1.51 (0.12)	-1.31 (0.07)
		0.1	-1.31 (0.12)	**-1.32 (0.13)**	-1.37 (0.11)	-1.31 (0.11)	-1.31 (0.1)	-1.36 (0.13)	-1.36 (0.11)	-1.35 (0.09)
		1	-2.16 (0.06)	**-2.22 (0.08)**	-2.23 (0.08)	-2.23 (0.08)	-2.13 (0.07)	-2.16 (0.06)	-2.15 (0.05)	-2.08 (0.06)
	Transformer	0.05	-2.07 (0.12)	**-2.16 (0.09)**	-2.03 (0.17)	-2.15 (0.14)	-1.96 (0.13)	-1.74 (0.13)	-2.09 (0.1)	-1.79 (0.11)
		0.1	-1.45 (0.13)	**-1.39 (0.12)**	-1.56 (0.13)	-1.38 (0.13)	-1.32 (0.11)	-1.43 (0.17)	-1.4 (0.1)	-1.33 (0.12)
		1	-1.96 (0.11)	**-1.77 (0.12)**	-1.81 (0.11)	-1.85 (0.09)	-1.85 (0.11)	-1.76 (0.08)	-1.78 (0.08)	-1.75 (0.09)
SELFIES	Transformer (pdop)	0.05	0.02 (0.01)	**0.04 (0.01)**	0.04 (0.01)	0.04 (0.01)	0.09 (0.01)	0.09 (0.03)	0.08 (0.01)	0.09 (0.02)
		0.1	0.01 (0.0)	**0.01 (0.0)**	0.02 (0.01)	0.02 (0.01)	0.03 (0.01)	0.08 (0.03)	0.04 (0.02)	0.05 (0.02)
		1	0.24 (0.01)	**0.23 (0.02)**	0.21 (0.02)	0.24 (0.01)	0.27 (0.01)	0.26 (0.01)	0.28 (0.01)	0.26 (0.01)
	Transformer (rano)	0.05	0.06 (0.01)	**0.06 (0.0)**	0.06 (0.01)	0.06 (0.01)	0.04 (0.01)	0.02 (0.0)	0.04 (0.0)	0.03 (0.01)
		0.1	0.03 (0.01)	**0.03 (0.01)**	0.03 (0.0)	0.02 (0.0)	–	–	–	–
		1	–	–	0.08 (0.0)	–	0.05 (0.01)	0.07 (0.01)	0.07 (0.0)	–
	Transformer (zale)	0.05	0.02 (0.01)	**0.04 (0.01)**	0.02 (0.01)	0.04 (0.01)	0.07 (0.01)	0.07 (0.02)	0.06 (0.01)	0.06 (0.01)
		0.1	0.04 (0.01)	**0.04 (0.01)**	0.04 (0.01)	0.03 (0.01)	0.01 (0.0)	0.02 (0.0)	0.09 (0.0)	0.04 (0.0)
		1	0.16 (0.02)	**0.17 (0.02)**	0.16 (0.02)	0.17 (0.02)	0.18 (0.02)	0.16 (0.02)	0.18 (0.02)	0.19 (0.02)
SELFIES ([84])	Transformer (pdop)	1	0.44 (0.01)	**0.44 (0.01)**	0.44 (0.01)	0.44 (0.01)	0.45 (0.0)	0.45 (0.01)	0.44 (0.01)	0.44 (0.01)
	Transformer (rano)	1	0.20 (0.02)	**0.22 (0.02)**	0.21 (0.01)	0.20 (0.01)	0.23 (0.01)	0.22 (0.01)	0.22 (0.01)	0.23 (0.01)
	Transformer (zale)	1	0.37 (0.01)	**0.37 (0.01)**	0.36 (0.01)	0.37 (0.01)	0.39 (0.01)	0.38 (0.01)	0.39 (0.01)	0.39 (0.01)
SMILES	GRU	0.05	0.89 (0.19)	**0.82 (0.15)**	0.64 (0.2)	0.89 (0.14)	1.0 (0.14)	0.98 (0.19)	0.7 (0.24)	1.07 (0.19)
		0.1	0.4 (0.22)	**0.41 (0.19)**	0.63 (0.26)	0.55 (0.14)	0.83 (0.13)	0.96 (0.33)	0.69 (0.18)	0.75 (0.18)
		1	1.44 (0.0)	**-0.16 (0.0)**	–	–	0.14 (0.65)	–	–	1.14 (0.0)
	LSTM	0.05	1.18 (0.25)	**1.54 (0.41)**	1.1 (0.15)	1.1 (0.36)	0.86 (0.27)	1.05 (0.21)	1.07 (0.29)	1.22 (0.27)
		0.1	–	–	–	–	0.21 (0.0)	–	–	–
		1	0.61 (0.16)	**0.52 (0.26)**	0.48 (0.13)	0.7 (0.23)	0.67 (0.23)	1.13 (0.16)	0.97 (0.28)	0.88 (0.23)
	Transformer	0.05	0.88 (0.18)	**0.97 (0.19)**	0.92 (0.14)	0.67 (0.15)	0.93 (0.17)	1.16 (0.1)	1.09 (0.11)	0.97 (0.24)
		0.1	0.42 (0.12)	**0.62 (0.14)**	0.44 (0.16)	0.73 (0.14)	0.91 (0.1)	0.6 (0.14)	0.81 (0.14)	0.7 (0.14)
		1	0.72 (0.18)	**0.48 (0.18)**	0.61 (0.15)	0.47 (0.16)	0.99 (0.12)	0.74 (0.15)	0.83 (0.16)	0.68 (0.15)

Table B.2: Ablation study for the initial length, success tolerance and failure tolerance for the TuRBO method. Average across 10 independent runs, of the best value across datasets, architectures. Column names indicate the length/success/fail values. Results from the main paper are in bold.

	Architecture	β	0.8/10/10	0.8/10/2	0.8/2/10	0.8/2/2	1.6/10/10	1.6/10/2	1.6/2/10	1.6/2/2
Expressions	GRU	0.05	-0.65 (0.11)	**-0.73 (0.12)**	-0.71 (0.09)	-0.61 (0.1)	-0.65 (0.12)	-0.59 (0.06)	-0.62 (0.11)	-0.72 (0.13)
		0.1	-0.56 (0.05)	**-0.61 (0.07)**	-0.59 (0.07)	-0.71 (0.1)	-0.54 (0.04)	-0.58 (0.05)	-0.62 (0.04)	-0.63 (0.04)
		1	-0.56 (0.05)	**-0.54 (0.05)**	-0.54 (0.06)	-0.6 (0.08)	-0.56 (0.06)	-0.61 (0.05)	-0.6 (0.04)	-0.59 (0.05)
	LSTM	0.05	-0.46 (0.03)	**-0.43 (0.02)**	-0.43 (0.02)	-0.38 (0.02)	-0.43 (0.02)	-0.43 (0.02)	-0.47 (0.04)	-0.4 (0.01)
		0.1	-0.38 (0.04)	**-0.39 (0.0)**	-0.41 (0.01)	-0.42 (0.02)	-0.42 (0.02)	-0.42 (0.02)	-0.42 (0.02)	-0.42 (0.02)
		1	-0.96 (0.09)	**-0.86 (0.0)**	-0.86 (0.0)	-1.01 (0.11)	-0.88 (0.01)	-0.98 (0.06)	-0.92 (0.04)	-0.88 (0.01)
	Transformer	0.05	-0.39 (0.04)	**-0.44 (0.02)**	-0.39 (0.04)	-0.4 (0.05)	-0.44 (0.02)	-0.39 (0.04)	-0.38 (0.04)	-0.42 (0.02)
		0.1	-0.38 (0.04)	**-0.41 (0.02)**	-0.42 (0.02)	-0.42 (0.02)	-0.39 (0.04)	-0.41 (0.01)	-0.41 (0.02)	-0.37 (0.04)
		1	-0.67 (0.1)	**-0.58 (0.1)**	-0.52 (0.08)	-0.62 (0.06)	-0.66 (0.11)	-0.62 (0.07)	-0.56 (0.05)	-0.54 (0.05)
SELFIES	Transformer (pdop)	0.05	0.13 (0.02)	**0.15 (0.03)**	0.17 (0.03)	0.18 (0.03)	0.23 (0.03)	0.17 (0.04)	0.22 (0.04)	0.2 (0.04)
		0.1	0.08 (0.02)	**0.09 (0.03)**	0.1 (0.03)	0.09 (0.03)	0.1 (0.03)	0.12 (0.04)	0.1 (0.03)	0.14 (0.04)
		1	0.36 (0.02)	**0.31 (0.03)**	0.36 (0.02)	0.34 (0.02)	0.38 (0.01)	0.38 (0.0)	0.4 (0.01)	0.36 (0.02)
	Transformer (rano)	0.05	0.17 (0.02)	**0.2 (0.02)**	0.2 (0.02)	0.18 (0.02)	0.11 (0.02)	0.08 (0.01)	0.12 (0.02)	0.13 (0.02)
		0.1	0.09 (0.02)	**0.1 (0.01)**	0.11 (0.02)	0.1 (0.02)	0.06 (0.01)	0.05 (0.01)	0.04 (0.01)	0.06 (0.01)
		1	0.07 (0.02)	**0.05 (0.02)**	0.11 (0.03)	0.07 (0.02)	0.16 (0.02)	0.16 (0.03)	0.11 (0.02)	0.12 (0.02)
	Transformer (zale)	0.05	0.11 (0.02)	**0.15 (0.03)**	0.12 (0.02)	0.16 (0.03)	0.22 (0.03)	0.23 (0.03)	0.21 (0.04)	0.19 (0.03)
		0.1	0.14 (0.01)	**0.16 (0.01)**	0.15 (0.02)	0.14 (0.01)	0.1 (0.02)	0.12 (0.02)	0.13 (0.02)	0.14 (0.03)
		1	0.36 (0.02)	**0.31 (0.03)**	0.38 (0.02)	0.36 (0.02)	0.38 (0.01)	0.34 (0.02)	0.31 (0.03)	0.38 (0.02)
SELFIES ([84])	Transformer (pdop)	1	0.48 (0.02)	**0.48 (0.02)**	0.53 (0.03)	0.49 (0.02)	0.51 (0.02)	0.51 (0.02)	0.49 (0.02)	0.50 (0.01)
	Transformer (rano)	1	0.38 (0.01)	**0.35 (0.01)**	0.35 (0.01)	0.32 (0.01)	0.36 (0.01)	0.36 (0.01)	0.37 (0.01)	0.34 (0.01)
	Transformer (zale)	1	0.44 (0.02)	**0.44 (0.01)**	0.44 (0.03)	0.49 (0.02)	0.47 (0.01)	0.46 (0.01)	0.49 (0.01)	0.47 (0.01)
SMILES	GRU	0.05	2.47 (0.22)	**2.47 (0.22)**	2.42 (0.21)	2.24 (0.17)	2.35 (0.23)	2.42 (0.28)	1.97 (0.18)	2.25 (0.31)
		0.1	1.76 (0.3)	**2.57 (0.31)**	2.45 (0.26)	2.07 (0.34)	2.24 (0.28)	2.26 (0.33)	2.24 (0.31)	1.92 (0.24)
		1	2.43 (0.32)	**2.48 (0.29)**	2.76 (0.24)	2.17 (0.49)	2.43 (0.32)	2.35 (0.37)	1.4 (0.52)	2.04 (0.33)
	LSTM	0.05	2.65 (0.32)	**2.73 (0.33)**	2.89 (0.26)	3.02 (0.34)	2.64 (0.3)	2.91 (0.29)	2.77 (0.23)	2.68 (0.37)
		0.1	1.97 (0.37)	**1.78 (0.43)**	1.98 (0.4)	2.16 (0.29)	2.36 (0.41)	1.87 (0.39)	2.34 (0.33)	1.73 (0.41)
		1	3.06 (0.2)	**2.71 (0.34)**	2.65 (0.16)	2.83 (0.25)	2.75 (0.28)	3.49 (0.26)	2.68 (0.31)	3.16 (0.35)
	Transformer	0.05	2.47 (0.23)	**2.88 (0.24)**	2.42 (0.17)	2.67 (0.27)	2.91 (0.3)	2.25 (0.2)	2.43 (0.24)	2.48 (0.29)
		0.1	3.03 (0.3)	**2.15 (0.18)**	2.51 (0.25)	2.41 (0.24)	2.28 (0.12)	2.28 (0.22)	2.53 (0.26)	2.46 (0.16)
		1	2.37 (0.21)	**2.25 (0.18)**	2.16 (0.17)	2.21 (0.25)	2.46 (0.2)	2.11 (0.12)	2.31 (0.16)	2.71 (0.32)

B.2.2 L-BFGS hyper-parameters

We provide an ablation study for the facet-length parameter. For each setup, the values reported in the main paper are highlighted in bold.

Table B.3: Ablation study for the facet length parameter of L-BFGS method. Average across 10 independent runs of the average top 20 values across datasets, architectures, and bound methods are displayed for facet lengths of size 1, 5 and 10. Results from the main paper are bold.

	Architecture	β	1	5	10
Expressions	GRU	0.05	-1.79 (0.08)	**-1.72 (0.07)**	-1.72 (0.07)
		0.1	-1.73 (0.11)	**-1.93 (0.09)**	-1.89 (0.11)
		1	-1.94 (0.07)	**-1.97 (0.07)**	-2.03 (0.09)
	LSTM	0.05	-1.89 (0.09)	**-1.78 (0.09)**	-1.93 (0.06)
		0.1	-1.29 (0.06)	**-1.39 (0.08)**	-1.37 (0.06)
		1	-2.04 (0.05)	**-2.04 (0.04)**	-2.04 (0.05)
	Transformer	0.05	-3.11 (0.14)	**-2.93 (0.13)**	-3.02 (0.09)
		0.1	-3.19 (0.28)	**-2.69 (0.25)**	-2.93 (0.22)
		1	-2.44 (0.11)	**-2.41 (0.09)**	-2.28 (0.11)
SELFIES	Transformer (pdop)	0.05	0.22 (0.01)	**0.21 (0.01)**	0.25 (0.01)
		0.1	0.25 (0.01)	**0.19 (0.01)**	0.22 (0.03)
		1	0.26 (0.01)	**0.25 (0.01)**	0.27 (0.01)
	Transformer (rano)	0.05	0.04 (0.0)	**0.03 (0.0)**	–
		0.1	–	–	–
		1	0.09 (0.02)	**0.07 (0.0)**	0.08 (0.01)
	Transformer (zale)	0.05	0.12 (0.02)	**0.13 (0.01)**	0.11 (0.01)
		0.1	0.08 (0.01)	**0.06 (0.0)**	0.07 (0.0)
		1	0.18 (0.02)	**0.18 (0.02)**	0.19 (0.02)
SELFIES ([84])	Transformer (pdop)	1	0.45 (0.00)	**0.42 (0.00)**	0.43 (0.01)
	Transformer (rano)	1	0.22 (0.01)	**0.17 (0.01)**	0.19 (0.01)
	Transformer (zale)	1	0.4 (0.01)	**0.37 (0.01)**	0.38 (0.01)
SMILES	GRU	0.05	0.65 (0.12)	**0.52 (0.12)**	0.52 (0.13)
		0.1	0.52 (0.14)	**0.12 (0.14)**	0.36 (0.15)
		1	–	–	–
	LSTM	0.05	0.67 (0.15)	**0.66 (0.12)**	0.56 (0.2)
		0.1	–	–	–
		1	0.49 (0.16)	**0.7 (0.2)**	0.55 (0.15)
	Transformer	0.05	0.52 (0.18)	**0.42 (0.15)**	0.55 (0.12)
		0.1	0.63 (0.18)	**0.61 (0.15)**	0.56 (0.17)
		1	0.16 (0.15)	**0.23 (0.12)**	0.39 (0.13)

Table B.4: Ablation study for the facet length parameter of L-BFGS method. Average across 10 independent runs, of the best value across datasets, architectures, and bound methods are displayed for facet lengths of size 1, 5 and 10. Results from the main paper are bold.

	Architecture	β	1	5	10
Expressions	GRU	0.05	-0.57 (0.07)	**-0.56 (0.04)**	-0.56 (0.07)
		0.1	-0.53 (0.04)	**-0.46 (0.02)**	-0.46 (0.02)
		1	-0.68 (0.05)	**-0.77 (0.02)**	-0.73 (0.04)
	LSTM	0.05	-0.57 (0.11)	**-0.56 (0.04)**	-0.67 (0.09)
		0.1	-0.4 (0.05)	**-0.44 (0.04)**	-0.47 (0.04)
		1	-1.01 (0.1)	**-1.02 (0.07)**	-0.86 (0.0)
	Transformer	0.05	-1.06 (0.13)	**-0.8 (0.1)**	-1.11 (0.18)
		0.1	-0.8 (0.14)	**-0.65 (0.1)**	-0.69 (0.09)
		1	-0.85 (0.1)	**-0.82 (0.1)**	-0.78 (0.12)
SELFIES	Transformer (pdop)	0.05	0.36 (0.01)	**0.36 (0.02)**	0.34 (0.03)
		0.1	0.29 (0.03)	**0.34 (0.02)**	0.27 (0.03)
		1	0.39 (0.01)	**0.36 (0.02)**	0.35 (0.03)
	Transformer (rano)	0.05	0.06 (0.01)	**0.07 (0.01)**	0.08 (0.01)
		0.1	0.08 (0.02)	**0.08 (0.02)**	0.07 (0.01)
		1	0.16 (0.02)	**0.16 (0.02)**	0.16 (0.02)
	Transformer (zale)	0.05	0.27 (0.03)	**0.27 (0.03)**	0.26 (0.03)
		0.1	0.15 (0.04)	**0.19 (0.03)**	0.18 (0.04)
		1	0.38 (0.01)	**0.38 (0.02)**	0.39 (0.01)
SELFIES ([84])	Transformer (pdop)	1	0.51 (0.02)	**0.47 (0.01)**	0.48 (0.01)
	Transformer (rano)	1	0.34 (0.02)	**0.37 (0.01)**	0.34 (0.02)
	Transformer (zale)	1	0.47 (0.0)	**0.44 (0.01)**	0.45 (0.02)
SMILES	GRU	0.05	2.06 (0.3)	**1.71 (0.24)**	2.02 (0.22)
		0.1	2.11 (0.28)	**1.74 (0.2)**	1.94 (0.22)
		1	2.4 (0.31)	**2.1 (0.32)**	2.34 (0.31)
	LSTM	0.05	2.31 (0.21)	**2.32 (0.19)**	2.15 (0.18)
		0.1	1.74 (0.5)	**1.85 (0.34)**	1.47 (0.49)
		1	2.8 (0.25)	**3.09 (0.16)**	2.42 (0.3)
	Transformer	0.05	1.82 (0.2)	**1.79 (0.13)**	1.74 (0.17)
		0.1	2.19 (0.16)	**2.24 (0.1)**	2.09 (0.17)
		1	1.72 (0.18)	**2.0 (0.14)**	1.96 (0.14)

B.2.3 Likelihood hyper-parameters

We provide an ablation study on the effect of changing the regularization strength parameter λ. For each setup, the values reported in the main paper are highlighted in bold.

B.2.3.1 Expressions

Table B.5: Ablation study for the regularization parameter λ for Likelihood. We report the average of the top 20 solutions found for the Expressions VAEs.

Architecture	β	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$
GRU	0.05	**-1.15**	-1.10	-1.11
	0.1	**-1.31**	-1.16	-1.14
	1	**-0.88**	-1.04	-1.15
LSTM	0.05	**-1.00**	-1.05	-1.17
	0.1	**-0.75**	-0.81	-0.72
	1	**-1.81**	-1.79	-1.78
Transformer	0.05	**-0.93**	-0.95	-0.98
	0.1	**-0.81**	-0.74	-0.70
	1	**-1.45**	-1.42	-1.19

Table B.6: Ablation study for the regularization parameter λ for Likelihood. We report the average of the top solution found for the Expressions VAEs.

Architecture	β	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$
GRU	0.05	**-0.4**	-0.43	-0.48
	0.1	**-0.43**	-0.42	-0.47
	1	**-0.43**	-0.58	-0.59
LSTM	0.05	**-0.4**	-0.45	-0.44
	0.1	**-0.32**	-0.39	-0.36
	1	**-0.86**	-0.86	-0.86
Transformer	0.05	**-0.38**	-0.39	-0.43
	0.1	**-0.41**	-0.39	-0.39
	1	**-0.65**	-0.50	-0.42

B.2.3.2 SMILES

Table B.7: Ablation study for the regularization parameter λ for Likelihood. We report the average of the top 20 solutions found for the SMILES VAEs.

Architecture	β	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.8$
GRU	0.05	2.15	**2.31**	2.25
	0.1	2.10	**2.12**	2.22
	1	0.60	**1.49**	1.40
LSTM	0.05	2.17	**2.28**	2.30
	0.1	0.81	**1.43**	1.58
	1	0.53	**1.67**	1.77
Transformer	0.05	2.43	**2.25**	2.32
	0.1	2.26	**2.23**	2.34
	1	2.01	**2.16**	1.98

Table B.8: Ablation study for the regularization parameter λ for Likelihood. We report the average of the top solution found for the SMILES VAEs.

Architecture	β	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.8$
GRU	0.05	2.96	**3.26**	3.27
	0.1	3.02	**3.33**	3.07
	1	3.12	**3.89**	3.47
LSTM	0.05	3.30	**3.37**	3.19
	0.1	2.44	**3.54**	2.92
	1	3.00	**3.48**	2.94
Transformer	0.05	3.24	**3.19**	3.07
	0.1	3.10	**3.16**	3.25
	1	3.18	**3.2**	2.92

B.2.3.3 SELFIES

Table B.9: Ablation study for the regularization parameter λ for Likelihood. We report the average of the top 20 solutions found for the SELFIES VAEs.

Objective	β	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$
Pdop	0.05	**0.37**	0.37	0.37
	0.1	**0.36**	0.36	0.34
	1	**0.34**	0.29	0.28
	Maus et al. [84]	0.42	0.40	**0.35**
Rano	0.05	**0.21**	0.22	0.21
	0.1	**0.22**	0.21	0.20
	1	**0.21**	0.23	0.19
	Maus et al. [84]	**0.22**	0.29	0.26
Zale	0.05	**0.33**	0.33	0.32
	0.1	**0.34**	0.33	0.32
	1	**0.30**	0.29	0.26
	Maus et al. [84]	0.41	0.39	**0.31**

Table B.10: Ablation study for the regularization parameter λ for Likelihood. We report the average of the top solution found for the SELFIES VAEs.

Objective	β	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$
Pdop	0.05	**0.43**	0.43	0.42
	0.1	**0.43**	0.41	0.39
	1	**0.40**	0.40	0.37
	Maus et al. [84]	0.47	0.45	**0.42**
Rano	0.05	**0.31**	0.31	0.29
	0.1	**0.33**	0.30	0.30
	1	**0.33**	0.30	0.33
	Maus et al. [84]	**0.34**	0.36	0.37
Zale	0.05	**0.44**	0.45	0.42
	0.1	**0.44**	0.42	0.42
	1	**0.42**	0.40	0.37
	Maus et al. [84]	0.49	0.49	**0.42**

B.2.4 LES hyper-parameters

We provide an ablation study on the effect of changing the regularization strength parameter λ. For each setup, the values reported in the main paper are highlighted in bold.

B.2.4.1 Expressions

Table B.11: Ablation study for the regularization parameter λ for LES. We report the average of the top 20 solutions found for the Expressions VAEs.

Architecture	β	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$
GRU	0.05	**-1.43**	-1.28	-1.32
	0.1	**-1.34**	-1.06	-1.29
	1	**-0.84**	-0.98	-1.08
LSTM	0.05	**-1.06**	-1.09	-1.00
	0.1	**-0.80**	-0.72	-0.73
	1	**-1.79**	-1.79	-1.84
Transformer	0.05	**-1.00**	-0.86	-0.92
	0.1	**-0.77**	-0.75	-0.67
	1	**-1.36**	-1.28	-1.05

Table B.12: Ablation study for the regularization parameter λ for LES. We report the average of the top solution found for the Expressions VAEs.

Architecture	β	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$
GRU	0.05	**-0.55**	-0.56	-0.44
	0.1	**-0.45**	-0.44	-0.41
	1	**-0.47**	-0.47	-0.52
LSTM	0.05	**-0.43**	-0.42	-0.41
	0.1	**-0.32**	-0.39	-0.39
	1	**-0.86**	-0.86	-0.86
Transformer	0.05	**-0.43**	-0.39	-0.41
	0.1	**-0.36**	-0.39	-0.32
	1	**-0.52**	-0.52	-0.45

B.2.4.2 SMILES

Table B.13: Ablation study for the regularization parameter λ for LES. We report the average of the top 20 solutions found for the SMILES VAEs.

Architecture	β	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.8$
GRU	0.05	2.20	**2.31**	2.32
	0.1	2.12	**2.30**	2.35
	1	0.38	**1.64**	1.77
LSTM	0.05	2.34	**2.33**	2.37
	0.1	0.94	**1.57**	1.80
	1	0.59	**1.94**	2.17
Transformer	0.05	2.30	**2.26**	2.35
	0.1	2.31	**2.26**	2.29
	1	2.09	**2.17**	2.30

Table B.14: Ablation study for the regularization parameter λ for LES. We report the average of the top solution found for the SMILES VAEs.

Architecture	β	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.8$
GRU	0.05	3.16	**3.29**	3.18
	0.1	3.19	**3.55**	3.42
	1	3.13	**3.85**	3.53
LSTM	0.05	3.51	**3.29**	3.35
	0.1	3.10	**3.54**	2.93
	1	2.6	**3.6**	3.29
Transformer	0.05	3.16	**3.21**	3.35
	0.1	3.09	**3.23**	3.03
	1	2.94	**3.2**	3.17

B.2.4.3 SELFIES

Table B.15: Ablation study for the regularization parameter λ for LES. We report the average of the top 20 solutions found for the SELFIES VAEs.

Objective	β	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$
Pdop	0.05	**0.37**	0.37	0.37
	0.1	**0.36**	0.36	0.36
	1	**0.35**	0.32	0.33
	Maus et al. [84]	0.46	0.48	**0.47**
Rano	0.05	**0.21**	0.22	0.21
	0.1	**0.21**	0.21	0.19
	1	**0.21**	0.21	0.20
	Maus et al. [84]	**0.30**	0.29	0.30
Zale	0.05	**0.33**	0.34	0.34
	0.1	**0.34**	0.33	0.33
	1	**0.31**	0.31	0.31
	Maus et al. [84]	0.40	0.40	**0.41**

Table B.16: Ablation study for the regularization parameter λ for LES. We report the average of the top solution found for the SELFIES VAEs.

Objective	β	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$
Pdop	0.05	**0.43**	0.43	0.42
	0.1	**0.43**	0.41	0.40
	1	**0.41**	0.42	0.40
	Maus et al. [84]	0.50	0.50	**0.54**
Rano	0.05	**0.33**	0.32	0.29
	0.1	**0.33**	0.30	0.31
	1	**0.31**	0.28	0.30
	Maus et al. [84]	**0.37**	0.37	0.39
Zale	0.05	**0.43**	0.45	0.45
	0.1	**0.44**	0.45	0.42
	1	**0.42**	0.43	0.39
	Maus et al. [84]	0.50	0.50	**0.47**

B.3 Additional experimental results

Table B.17: Average of the top solution found during LSO (higher is better), across datasets and decoder architectures. We **bold** the best method and underline the second-best. The average ranking for each method (lower is better) is provided, along with the number of times each method is within one standard deviation of the best. LES and p achieve the highest value most frequently (14 out of 30) and outperforms other methods both in terms of the average ranking and the frequency of being within one standard deviation of the best result.

	Architecture	β	LES	L-BFGS	UC	GA	Prior	TuRBO	Likelihood
Expressions	GRU	0.05	-0.55 (0.04)	-0.56 (0.04)	-0.59 (0.04)	-0.45 (0.05)	**-0.4 (0.07)**	-0.73 (0.12)	**-0.4 (0.07)**
		0.1	-0.45 (0.03)	-0.46 (0.02)	-0.47 (0.05)	-0.43 (0.02)	**-0.37 (0.03)**	-0.61 (0.07)	-0.43 (0.02)
		1	-0.47 (0.03)	-0.77 (0.02)	-0.51 (0.04)	-0.46 (0.02)	-0.47 (0.03)	-0.54 (0.05)	**-0.43 (0.01)**
	LSTM	0.05	-0.43 (0.02)	-0.56 (0.04)	-0.52 (0.05)	-0.43 (0.01)	-0.41 (0.01)	-0.43 (0.02)	**-0.4 (0.01)**
		0.1	**-0.32 (0.05)**	-0.44 (0.04)	-0.39 (0.04)	-0.38 (0.02)	-0.4 (0.01)	-0.39 (0.0)	**-0.32 (0.04)**
		1	**-0.86 (0.0)**	-1.02 (0.07)	**-0.86 (0.0)**	**-0.86 (0.0)**	**-0.86 (0.0)**	**-0.86 (0.0)**	-0.91 (0.04)
	Transformer	0.05	-0.43 (0.03)	-0.8 (0.1)	-0.57 (0.05)	-0.44 (0.02)	**-0.37 (0.05)**	-0.44 (0.02)	-0.38 (0.04)
		0.1	-0.36 (0.03)	-0.65 (0.1)	-0.55 (0.04)	-0.39 (0.01)	**-0.35 (0.04)**	-0.41 (0.02)	-0.41 (0.02)
		1	**-0.52 (0.05)**	-0.82 (0.1)	-0.58 (0.05)	-0.58 (0.04)	-0.62 (0.09)	-0.58 (0.1)	-0.65 (0.08)
SELFIES	Transformer (pdop)	0.05	**0.43 (0.0)**	0.36 (0.02)	0.42 (0.0)	0.42 (0.0)	0.42 (0.0)	0.15 (0.03)	**0.43 (0.0)**
		0.1	**0.43 (0.0)**	0.34 (0.02)	0.42 (0.01)	0.42 (0.0)	0.41 (0.01)	0.09 (0.03)	**0.43 (0.01)**
		1	**0.41 (0.01)**	0.36 (0.02)	0.39 (0.01)	0.4 (0.0)	0.38 (0.01)	0.31 (0.03)	0.4 (0.0)
	Transformer (rano)	0.05	0.33 (0.01)	0.07 (0.01)	**0.36 (0.01)**	0.33 (0.01)	0.32 (0.01)	0.2 (0.02)	0.31 (0.01)
		0.1	0.33 (0.01)	0.08 (0.02)	**0.36 (0.02)**	0.32 (0.01)	0.32 (0.01)	0.1 (0.01)	0.33 (0.01)
		1	0.31 (0.01)	0.16 (0.02)	**0.39 (0.02)**	0.31 (0.01)	0.33 (0.02)	0.05 (0.02)	0.33 (0.01)
	Transformer (zale)	0.05	0.43 (0.01)	0.27 (0.03)	0.42 (0.01)	0.43 (0.01)	0.42 (0.0)	0.15 (0.03)	**0.44 (0.01)**
		0.1	**0.44 (0.01)**	0.19 (0.03)	0.42 (0.01)	0.42 (0.01)	0.42 (0.01)	0.16 (0.01)	**0.44 (0.01)**
		1	**0.42 (0.01)**	0.38 (0.02)	0.39 (0.01)	**0.42 (0.01)**	0.37 (0.01)	0.31 (0.03)	**0.42 (0.01)**
SELFIES ([84])	Transformer (pdop)	1	**0.54 (0.01)**	0.46 (0.01)	0.50 (0.01)	0.49 (0.01)	0.36 (0.02)	0.48 (0.02)	0.42 (0.00)
	Transformer (rano)	1	**0.37 (0.00)**	**0.37 (0.01)**	0.36 (0.01)	**0.37 (0.01)**	0.34 (0.01)	0.35 (0.02)	0.34 (0.01)
	Transformer (zale)	1	**0.47 (0.01)**	0.43 (0.01)	0.44 (0.01)	**0.47 (0.01)**	0.39 (0.01)	0.44 (0.01)	0.42 (0.01)
SMILES	GRU	0.05	**3.29 (0.1)**	1.71 (0.24)	3.13 (0.07)	3.26 (0.11)	3.18 (0.06)	2.47 (0.22)	3.26 (0.08)
		0.1	**3.55 (0.14)**	1.74 (0.2)	3.2 (0.1)	3.31 (0.16)	3.15 (0.11)	2.57 (0.31)	3.33 (0.12)
		1	3.85 (0.17)	2.1 (0.32)	2.24 (0.28)	3.66 (0.16)	**3.89 (0.28)**	2.48 (0.29)	**3.89 (0.2)**
	LSTM	0.05	3.29 (0.07)	2.32 (0.19)	3.12 (0.08)	3.3 (0.1)	3.28 (0.1)	2.73 (0.33)	**3.37 (0.09)**
		0.1	**3.66 (0.2)**	1.85 (0.34)	2.65 (0.19)	3.52 (0.22)	3.57 (0.18)	1.78 (0.43)	3.54 (0.16)
		1	**3.6 (0.14)**	3.09 (0.16)	2.6 (0.3)	3.18 (0.17)	3.28 (0.17)	2.71 (0.34)	3.48 (0.11)
	Transformer	0.05	**3.21 (0.08)**	1.79 (0.13)	3.1 (0.07)	3.14 (0.04)	3.14 (0.04)	2.88 (0.24)	3.19 (0.08)
		0.1	3.23 (0.04)	2.24 (0.1)	**3.28 (0.08)**	3.11 (0.05)	3.09 (0.06)	2.15 (0.18)	3.16 (0.06)
		1	**3.2 (0.07)**	2.0 (0.14)	2.8 (0.13)	3.13 (0.07)	3.11 (0.1)	2.25 (0.18)	**3.2 (0.06)**
	Average rank		**2.15**	6.17	4.17	3.15	3.86	5.7	2.76
	# within 1 std of best		**19**	1	5	8	10	2	16

Table B.20: Effect of increasing λ parameter for LES, for the expressions dataset. Increasing the value of the parameter λ increases the percentage of valid solution in all cases.

Architecture	β	**LES** ($\lambda = 0.05$)	**LES** ($\lambda = 0.5$)
GRU	**0.05**	0.92	0.96
	0.1	0.7	0.82
	1	0.72	0.87
LSTM	**0.05**	0.93	0.98
	0.1	0.93	0.98
	1	0.76	0.97
Transformer	**0.05**	0.87	0.94
	0.1	0.88	0.96
	1	0.83	0.85

B.4 Background on related work

Bayesian uncertainty [90] Under a Bayesian viewpoint, the trained neural network parameters ($\boldsymbol{\theta}$) follow a variational distribution, which we can sample from using MC-Dropout [41]. Based on this distribution, the uncertainty is defined as

$$\mathcal{M}(\boldsymbol{z}) = \mathcal{H}_p(p(\boldsymbol{X}|\boldsymbol{Z} = \boldsymbol{z})) - \mathbb{E}_{\boldsymbol{\theta}}\mathcal{H}_{p_{\boldsymbol{\theta}}}(p_{\boldsymbol{\theta}}(\boldsymbol{X}|\boldsymbol{Z} = \boldsymbol{z})), \tag{B.21}$$

where $\mathcal{H}$ is the entropy and $p(\boldsymbol{X}|\boldsymbol{Z} = \boldsymbol{z})$ is the posterior predictive distribution. The uncertainty is estimated using MC-Dropout with important sampling (using the posteior predictive as the importance distribution) designed to approximate the expectations over the random variable $\boldsymbol{X}$, as it is typically a very large space (i.e., the sample of molecules that can implemented using a SMILE string with 120 characters)

Table B.18: Average of the top 20 solutions found during LSO (higher is better) across datasets and decoder architectures. We **bold** the best method and underline the second-best. The average ranking for each method (lower is better) is provided, along with the number of times each method is within one standard deviation of the best. LES achieves the highest value in most experiments (20 out of 30) and outperforms other methods in terms of both the average ranking and the frequency of being within one standard deviation of the best result.

	Architecture	β	LES	L-BFGS	UC	GA	Prior	TuRBO	Likelihood
Expressions	GRU	0.05	-1.43 (0.05)	-1.72 (0.07)	-1.5 (0.05)	-1.25 (0.06)	-1.28 (0.05)	-2.1 (0.11)	**-1.15 (0.09)**
		0.1	-1.34 (0.08)	-1.93 (0.09)	-2.01 (0.12)	-1.21 (0.09)	**-1.18 (0.11)**	-2.14 (0.18)	-1.31 (0.08)
		1	**-0.84 (0.02)**	-1.97 (0.07)	-0.91 (0.03)	**-0.84 (0.02)**	-0.89 (0.05)	-1.34 (0.05)	-0.88 (0.03)
	LSTM	0.05	-1.06 (0.06)	-1.78 (0.09)	-1.53 (0.06)	**-0.98 (0.05)**	-1.02 (0.03)	-1.59 (0.13)	-1.0 (0.07)
		0.1	-0.8 (0.03)	-1.39 (0.08)	-1.09 (0.04)	-0.79 (0.03)	-0.79 (0.03)	-1.32 (0.13)	**-0.75 (0.03)**
		1	**-1.79 (0.02)**	-2.04 (0.04)	-2.02 (0.03)	-1.81 (0.02)	-1.83 (0.02)	-2.22 (0.08)	-1.81 (0.03)
	Transformer	0.05	-1.0 (0.04)	-2.93 (0.13)	-1.64 (0.08)	-1.03 (0.04)	-0.99 (0.05)	-2.16 (0.09)	**-0.93 (0.05)**
		0.1	**-0.77 (0.02)**	-2.69 (0.25)	-1.79 (0.08)	**-0.77 (0.04)**	-0.78 (0.03)	-1.39 (0.12)	-0.81 (0.03)
		1	**-1.36 (0.09)**	-2.41 (0.09)	-1.52 (0.09)	-1.5 (0.07)	-1.41 (0.08)	-1.77 (0.12)	-1.45 (0.12)
SELFIES	Transformer (pdop)	0.05	**0.37 (0.0)**	0.21 (0.01)	0.36 (0.0)	**0.37 (0.0)**	**0.37 (0.0)**	0.04 (0.01)	**0.37 (0.0)**
		0.1	**0.36 (0.0)**	0.19 (0.01)	0.35 (0.0)	**0.36 (0.0)**	0.34 (0.0)	0.01 (0.0)	**0.36 (0.0)**
		1	**0.35 (0.0)**	0.25 (0.01)	0.31 (0.01)	0.33 (0.0)	0.28 (0.01)	0.23 (0.02)	0.34 (0.0)
	Transformer (rano)	0.05	0.21 (0.0)	0.03 (0.0)	**0.22 (0.0)**	0.21 (0.0)	0.21 (0.0)	0.06 (0.0)	0.21 (0.0)
		0.1	0.21 (0.0)	–	**0.23 (0.01)**	0.21 (0.0)	0.22 (0.0)	0.03 (0.01)	0.22 (0.0)
		1	0.21 (0.0)	0.07 (0.0)	**0.22 (0.01)**	0.19 (0.0)	0.2 (0.0)	–	0.21 (0.0)
	Transformer (zale)	0.05	**0.33 (0.0)**	0.13 (0.01)	0.32 (0.0)	**0.33 (0.0)**	0.32 (0.0)	0.04 (0.01)	**0.33 (0.0)**
		0.1	**0.34 (0.0)**	0.06 (0.0)	0.31 (0.01)	0.32 (0.01)	0.31 (0.0)	0.04 (0.01)	**0.34 (0.0)**
		1	**0.31 (0.0)**	0.18 (0.02)	0.26 (0.01)	0.29 (0.01)	0.23 (0.01)	0.17 (0.02)	0.3 (0.01)
SELFIES ([84])	Transformer (pdop)	1	**0.48 (0.01)**	0.42 (0.01)	0.44 (0.00)	0.44 (0.00)	0.3 (0.01)	0.44 (0.01)	0.35 (0.01)
	Transformer (rano)	1	**0.29 (0.00)**	0.17 (0.03)	0.27 (0.01)	0.27 (0.01)	**0.25 (0.01)**	0.22 (0.03)	0.22 (0.01)
	Transformer (zale)	1	**0.4 (0.00)**	0.37 (0.01)	0.38 (0.01)	0.39 (0.0)	0.27 (0.01)	0.37 (0.01)	0.31 (0.01)
SMILES	GRU	0.05	**2.31 (0.04)**	0.52 (0.12)	2.18 (0.05)	2.21 (0.05)	2.2 (0.04)	0.82 (0.15)	**2.31 (0.03)**
		0.1	**2.3 (0.05)**	0.12 (0.14)	1.68 (0.04)	2.09 (0.07)	1.93 (0.06)	0.41 (0.19)	2.12 (0.07)
		1	**1.64 (0.14)**	–	–	1.19 (0.2)	0.58 (0.28)	-0.16 (0.0)	1.49 (0.14)
	LSTM	0.05	**2.33 (0.03)**	0.66 (0.12)	2.02 (0.03)	2.22 (0.05)	2.13 (0.05)	1.54 (0.41)	2.28 (0.03)
		0.1	**1.57 (0.1)**	–	–	0.8 (0.12)	0.9 (0.09)	–	1.43 (0.12)
		1	**1.94 (0.1)**	0.7 (0.2)	0.95 (0.24)	1.14 (0.21)	0.81 (0.26)	0.52 (0.26)	1.67 (0.13)
	Transformer	0.05	**2.26 (0.04)**	0.42 (0.15)	2.04 (0.05)	2.22 (0.03)	2.24 (0.03)	0.97 (0.19)	2.25 (0.03)
		0.1	**2.26 (0.03)**	0.61 (0.15)	2.08 (0.04)	2.21 (0.03)	2.17 (0.03)	0.62 (0.14)	2.23 (0.02)
		1	**2.17 (0.05)**	0.23 (0.12)	1.32 (0.09)	1.98 (0.06)	1.82 (0.06)	0.48 (0.18)	2.16 (0.05)
	Average rank		**1.9**	6.4	4.3	2.87	3.8	6.1	2.53
	# within 1 std of best		**22**	0	3	7	5	0	13

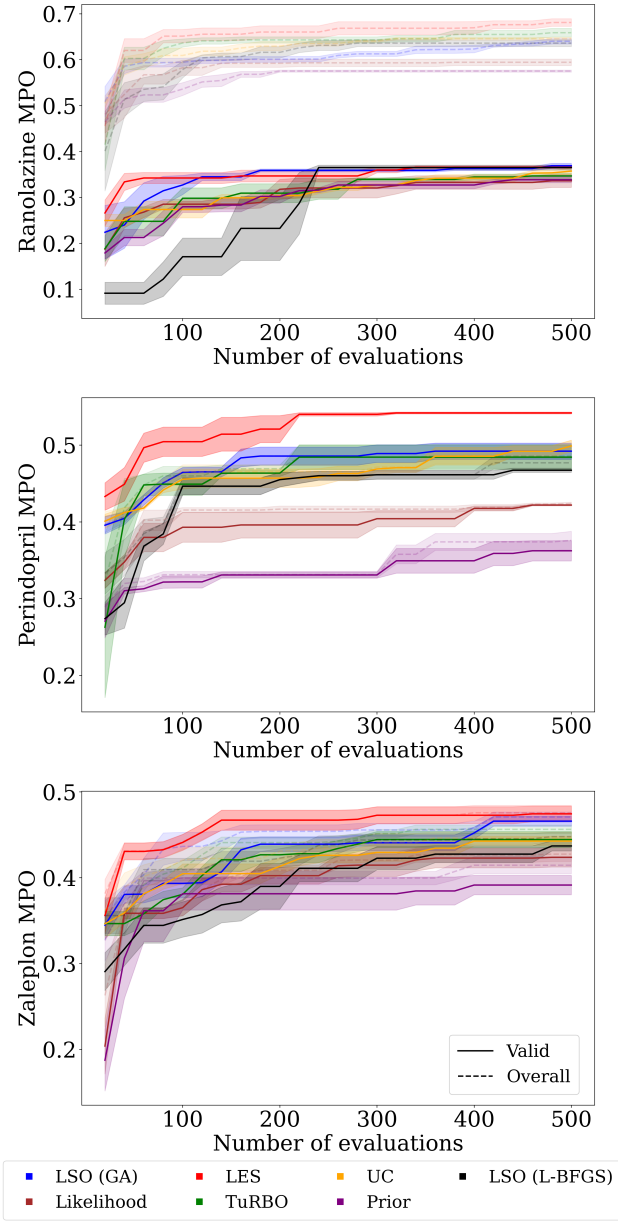

Figure B.1: Cumulative true objective for the best solution found during Bayesian optimization with the pre-trained SELFIES-VAE ([84]). Each method is shown in a distinct color. Solid lines represent solutions passing quality filters, while dashed lines include all evaluations. LES achieves the best performance on Perindopril MPO and is competitive on Zaleplon MPO and Ranolazine MPO.

Table B.19: AUROC values (higher is better) for identifying valid data points within the latent space, across datasets and decoder architectures. Data points are sampled from the training data, the VAE prior (standard Gaussian) and out of distribution (Gaussian with std of 5). LES achieves the best performance in most cases (18 out of 22) and on average. In addition, LES achieves AUROC values of at least 0.75 in all cases, indicating it can effectively differentiate valid from invalid data points.

Dataset	Arch.	β	LES	Polarity	Prior	Uncertainty	Train distances	Likelihood
SMILES	GRU	0.05	0.93	0.42	0.09	0.85	0.85	**0.94**
		0.1	**0.94**	0.72	0.12	0.84	0.86	**0.94**
		1.0	0.91	0.35	0.16	0.87	0.80	**0.92**
	LSTM	0.05	**0.99**	0.93	0.07	**0.99**	0.91	0.98
		0.1	0.89	0.67	0.21	0.89	0.81	**0.9**
		1.0	**0.97**	0.76	0.12	0.95	0.85	0.96
	Transformer	0.05	**0.93**	0.89	0.14	0.84	0.77	0.92
		0.1	**0.94**	0.91	0.14	0.87	0.86	0.93
		1.0	**0.97**	0.93	0.10	0.89	0.85	0.95
Expressions	GRU	0.05	**0.96**	0.89	0.38	**0.96**	0.67	0.88
		0.1	**0.94**	0.86	0.42	**0.94**	0.71	0.8
		1.0	**0.94**	0.80	0.57	**0.94**	0.75	0.89
	LSTM	0.05	**0.96**	0.86	0.38	0.90	0.67	**0.96**
		0.1	**0.95**	0.83	0.37	0.91	0.66	**0.95**
		1.0	**0.95**	0.79	0.56	0.91	0.72	0.91
	Transformer	0.05	**0.91**	0.79	0.43	0.86	0.71	0.90
		0.1	**0.91**	0.79	0.53	0.87	0.78	0.89
		1.0	0.86	0.61	0.70	**0.92**	0.89	**0.92**
SELFIES	Transformer	0.05	**1.0**	0.99	0.02	0.99	0.62	0.97
		0.1	**0.99**	0.98	0.03	0.96	0.81	0.98
		1.0	**0.95**	0.94	0.06	0.85	0.72	0.93
SELFIES ([84])	Transformer	–	**0.75**	0.39	0.69	0.33	0.69	0.70
	Average		**0.93**	0.78	0.29	0.88	0.77	0.91

Table B.21: Proportion of valid solutions found during LSO (higher is better) across datasets and decoder architectures.We bold the best method (higher is better) and underline the second best. LES improves the validity of the solutions compared with GA (which is LES with $\lambda = 0$) across all datasets.

	Architecture	β	LES	L-BFGS	UC	GA	p	TuRBO	Likelihood
Expressions	GRU	0.05	0.92	0.59	**1.0**	0.91	0.91	0.94	0.89
		0.1	0.7	0.57	**1.0**	0.64	0.66	0.9	0.67
		1	0.72	0.45	**0.99**	0.69	0.69	0.83	0.69
	LSTM	0.05	0.93	0.62	**1.0**	0.88	0.89	0.94	0.92
		0.1	0.93	0.67	**1.0**	0.9	0.89	0.94	0.92
		1	0.76	0.58	**0.99**	0.65	0.66	0.89	0.73
	Transformer	0.05	0.87	0.37	**1.0**	0.83	0.85	0.9	0.84
		0.1	0.88	0.28	**1.0**	0.8	0.81	0.89	0.85
		1	0.83	0.36	**1.0**	0.74	0.76	0.87	0.79
SELFIES	Transformer (pdop)	0.05	0.8	0.05	**0.81**	0.76	0.73	0.14	0.77
		0.1	0.68	0.04	**0.71**	0.62	0.56	0.14	0.64
		1	**0.59**	0.08	0.55	0.49	0.44	0.08	0.53
	Transformer (rano)	0.05	0.66	0.02	**0.73**	0.61	0.57	0.09	0.62
		0.1	0.57	0.01	**0.67**	0.52	0.44	0.04	0.53
		1	0.43	0.02	**0.49**	0.36	0.28	0.01	0.39
	Transformer (zale)	0.05	0.71	0.08	**0.74**	0.66	0.62	0.12	0.69
		0.1	0.66	0.01	**0.67**	0.59	0.51	0.08	0.63
		1	**0.54**	0.18	0.47	0.43	0.35	0.17	0.46
SELFIES ([84])	Transformer (pdop)	1	0.69	0.45	0.56	0.58	**0.75**	0.43	0.68
	Transformer (rano)	1	**0.26**	0.07	0.14	0.19	0.20	0.11	0.15
	Transformer (zale)	1	0.65	0.52	0.66	0.66	0.75	0.48	**0.76**
SMILES	GRU	0.05	**0.61**	0.14	0.48	0.47	0.44	0.12	0.59
		0.1	**0.37**	0.07	0.25	0.23	0.22	0.06	0.35
		1	**0.09**	0.02	0.05	0.05	0.04	0.02	0.08
	LSTM	0.05	**0.6**	0.11	0.44	0.42	0.39	0.11	0.57
		0.1	**0.08**	0.01	0.06	0.05	0.04	0.01	0.07
		1	**0.16**	0.09	0.07	0.08	0.07	0.09	0.12
	Transformer	0.05	0.7	0.42	0.7	0.68	0.68	0.31	**0.72**
		0.1	0.65	**0.67**	0.63	0.6	0.55	0.41	0.64
		1	**0.48**	0.35	0.46	0.34	0.26	0.3	0.45
	Average		0.62	0.26	**0.64**	0.55	0.53	0.38	0.59

Appendix C

Supplementary Material for Chapter 4

C.1 List of zero shot scores and embeddings

C.1.1 Zero-shot scores

We describe below the different methods included in our study for zero-shot score prediction. Our focus was to include a set of diverse methods, which include PLMs, MSA based models and structure based models.

ESM2 ESM2 [77] is a transformer-based protein language model. We use the 650 million parameters version consisting of 33 layers. It follows a BERT style encoder only transformer architecture and is trained on the UniRef50 [117] protein sequence database using a masked language modeling objective.

MSA Transformer MSA Transformer [105] uses a specialized transformer architecture with interleaved row and column (axial) self-attention layers to process multiple sequence alignments (MSAs). It has approximately 100 million parameters and 12 layers, and is trained on 26 million MSAs. An MSA is generated for each UniRef50 sequence.

TranceptEVE TranceptEVE [94] is a hybrid model that integrates an autoregressive transformer (Tranception [95]) with a variational autoencoder (EVE) trained on family-specific MSAs. Tranception offers three model variants; the best-performing Tranception-EVE configuration on the ProteinGym indel benchmark uses the medium-sized Tranception model (16 attention heads, 24 layers, 1024-dimensional embeddings) trained on UniRef100. EVE is trained on MSAs built from UniRef100 for 3,219 clinically relevant proteins.

ProSST ProSST [76] consists of a transformer model integrated with a geometric vector perceptron (GVP) encoder that encodes 3D structural information. The model has around 250 million parameters, with the transformer comprising 12 layers and the GVP encoder

using approximately six layers. This model is pretrained on data collected from AlphaFoldDB [62, 122, 121], which contained more than 214 million structures. The dataset used for training the structure encoder is extracted from CATH43-S40 [113], a dataset of manually annotated protein crystal structural domains.

C.1.2 Embedding models

We describe below the embeddings models we used, our focus was to include at least two models with different architecture. ESM2 and ESM1v are similar and were included for ease of implementation.

ESM2 See Section C.1.1

CARP CARP (Convolutional Autoencoding Representations of Proteins [134]) is a convolutional neural network model based on a ByteNet encoder architecture with dilated convolutions. We used the CAPR model with approximately 640 million parameters and 56 layers. The model is trained on UniRef50 using a masked language modeling task. Unlike transformers, CARP captures long-range dependencies via convolution, scaling linearly with sequence length. Embeddings are taken from the final hidden representations, with a hidden dimension of 1280.

ESM1v ESM1v [86] is a transformer model based on the ESM1b architecture, optimized for zero-shot variant effect prediction. It has 650 million parameters and 33 layers, with a hidden dimension of 1,280. The final model is an ensemble of five independently trained networks, each trained on UniRef90. We extract embeddings from the last hidden layer (layer 33) of the first model in the ensemble.

C.2 Pred-check procedure

Following the PCS framework, we apply a **prediction check (Pred-Check)** step to filter and weight representations that are less predictive for a particular fitness prediction problem.

The pred-check procedure has two main goals:

- **Filtering:** Remove the worst-performing zero-shot scores.
- **Weighting:** Assign weights to the remaining scores based on their predictive quality.

Procedure Let $\{y_i\}_{i=1}^n$ denote the training fitness labels and $\{z_i^{(j)}\}_{i=1}^n$ for $j = 1, \ldots, h$ denote the values of the zero-shot scores. Our procedure involves the following steps:

1. Assign a prediction score ($s^{(i)}$) for every zero-shot scores vector ($\{z_i^{(j)}\}_{i=1}^n$) using the training labels ($\{y_i\}_{i=1}^n$).

2. Keep the top $k \leq h$ scores (assuming higher is better). We set $k = 3$, but find that similar performance is obtained with $k = 4$ or $k = 2$.

3. Obtain the weights by normalizing the prediction score via soft-max transformation.

We consider the following prediction scores:

- **Correlation (Corr)** — For each zero-shot score vector, we compute the Spearman correlation with the training fitness labels.
- **LASSO** — All zero-shot score vectors are concatenated into a feature matrix. We then fit a LASSO regression model (using 5-fold cross-validation to select the regularization strength [100]). The prediction score for each zero-shot score is the absolute value of its corresponding LASSO coefficient, reflecting its importance in predicting fitness. This can be viewed as a form of feature selection.
- **RF-MDI** — As with LASSO, the zero-shot scores are concatenated into a feature matrix. A Random Forest model is trained to predict fitness, and the prediction score for each zero-shot score is its Mean Decrease in Impurity (MDI) importance value.

We highlight that the above pred-check procedure does not require any held-out calibration set, and is done using the training set only.

C.3 Ablation studies

We perform an ablation study to quantify the contribution of each component in our SP fitness predictors. Specifically, we ablate the following elements: (1) ensembling across multiple embeddings, (2) ensembling across dimensionality reduction methods, and (3) the pred-check procedure (i.e., use of RF-MDI and its advantage over using a single score or just an average).

Each ablation is evaluated by measuring the change in Spearman correlation across 217 ProteinGym assays. For each component, we report box plots and p-values from two-sample t-tests assessing whether the component improves average correlation across assays. The specific ablations are described below:

Embedding To assess the benefit of ensembling multiple embeddings (ESM1v [86], ESM2 [77], and CARP [134]), we compare the performance of the full ensemble (after pred-check) to that of a variant that uses only a single embedding model. Both versions use the same weighting scheme based on RF-MDI.

Pred-Check To isolate the impact of the pred-check step, we compare our RF-MDI procedure to the three alternative methods described in Section C.2. We also include a baseline that uses each zero-shot score individually in the ensemble. All settings use the same embeddings and dimensionality reduction methods; only the selection and weighting of zero-shot scores differ.

Dimensionality Reduction To evaluate the effect of ensembling across dimensionality reduction methods, we compare the full ensemble to a variant that uses only mean pooling for each embedding. Both use the same zero-shot selection and weighting via RF-MDI.

Results The results for SP Kermut, SP Bayesian Ridge, and SP CNN are shown in Figures C.1, C.2, and C.3, respectively. For all three base models, the pred-check step and dimensionality reduction ensemble lead to statistically significant improvements. Ensembling embeddings yields additional gains for SP Kermut and SP Bayesian Ridge. For SP CNN—the weakest overall model—embedding ensembling performs similarly to using ESM2 alone, but not worse.

C.4 Prediction intervals

C.4.1 Description of methods

We consider the supervised learning setup described in Section 4.2, where we are given a training and a calibration set of mutated sequences and their corresponding fitness labels (assuming these two are exchangeable, the marginal coverage guarantees will hold), denoted by $\mathcal{D}_{Train} = (m_1, y_1), \ldots, (m_n, y_n), \mathcal{D}_{Cal} = (m_1, y_1), \ldots, (m_n, y_m)$. Let $\hat{f}$ be a fitness predictor trained on $\mathcal{D}_{Train}$ that maps a mutation m_i to a predicted fitness value $\hat{y}_i$ along with an associated uncertainty estimate $\hat{\sigma}_i$, i.e., $\hat{f}(m_i) = (\hat{y}_i, \hat{\sigma}_i)$. We define an α-level prediction set $\hat{C}^\alpha(m) \subset \mathbb{R}$ intended to contain the true fitness value of a new test point (m_{new}, y_{new}) with probability at least $1 - \alpha$:

$$\mathbb{P}\left(y_{new} \in \hat{C}^\alpha(m_{new})\right) \geq 1 - \alpha, \tag{C.1}$$

where the probability is taken over the randomness in the calibration set and test point, conditional on the training set.

Split Conformal [72] is the most widely used conformal inference method. It forms prediction sets of the form:

$$\hat{C}^\alpha(m) = [\hat{f}(m) - \hat{q}_\alpha, \hat{f}(m) + \hat{q}_\alpha], \tag{C.2}$$

where $\hat{q}_\alpha$ is defined as the $\lceil (1-\alpha) \times (m+1) \rceil$ smallest value of $\left(|\hat{f}(m_1) - y_1|, \ldots, |\hat{f}(m_m) - y_m|\right)$.

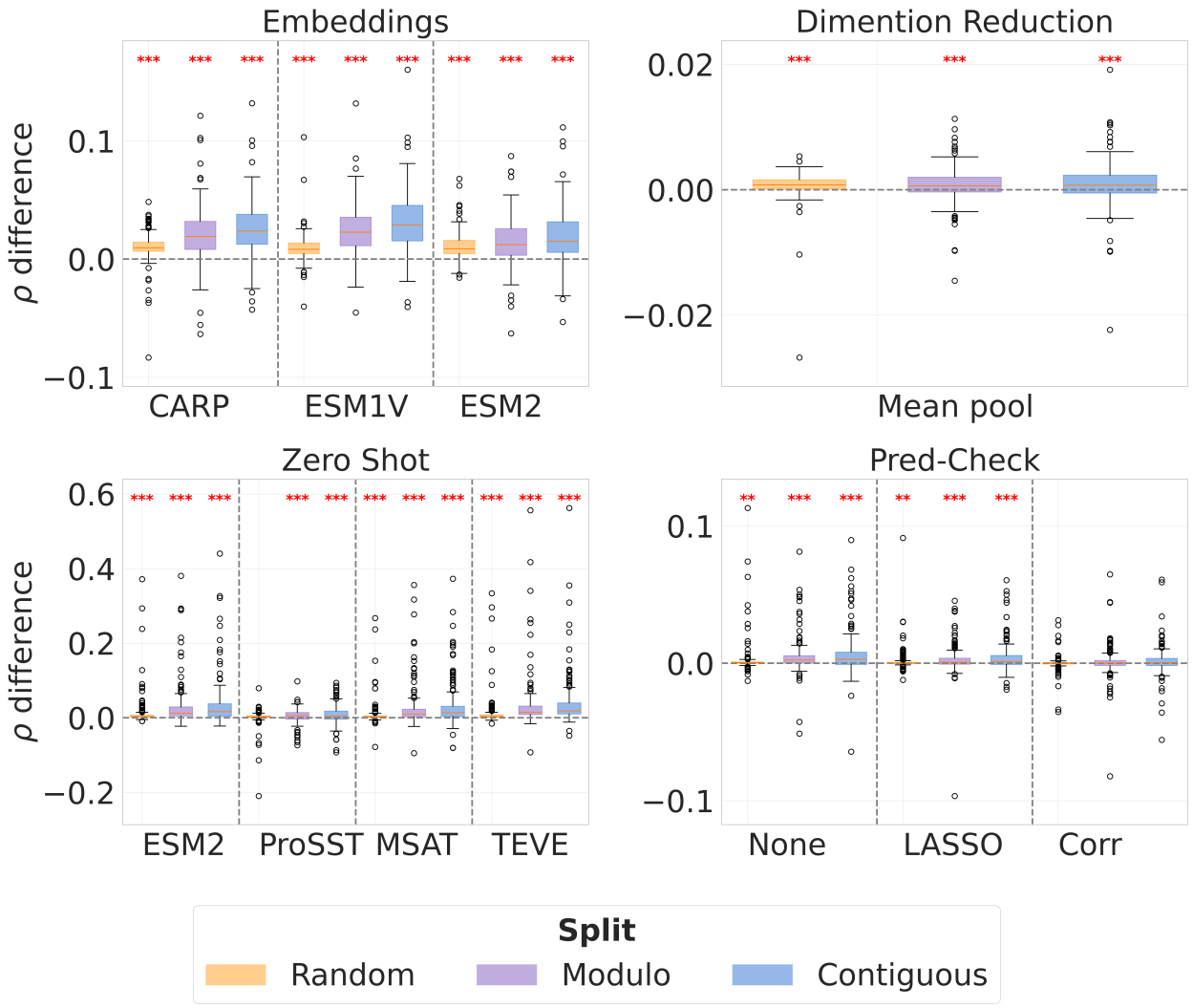

Figure C.1: Each component of SP Kermut improves prediction accuracy. Box plots show the change in test set Spearman correlation when ablations are applied to individual components. Asterisks indicate significance levels from one-sided two-sample t-tests (*<0.05, **<0.01, ***<0.001), and colors denote different data splits. The top left panel compares SP Kermut to a model using only a single embedding; the top right to a model with only mean pooling instead of multiple randomized dimensionality reduction; the bottom left to a model using only a single zero-shot score (MSAT is short for MSA Transformer and TEVE is a short for TranceptEVE) ; and the bottom right compares the RF-MDI pred-check to alternative pred-check procedures. Ensembling embeddings and random dimensionality reduction both significantly improve performance. The RF-MDI pred-check outperforms each individual zero-shot score and their ensemble, and performs comparably to the correlation-based method (Corr).

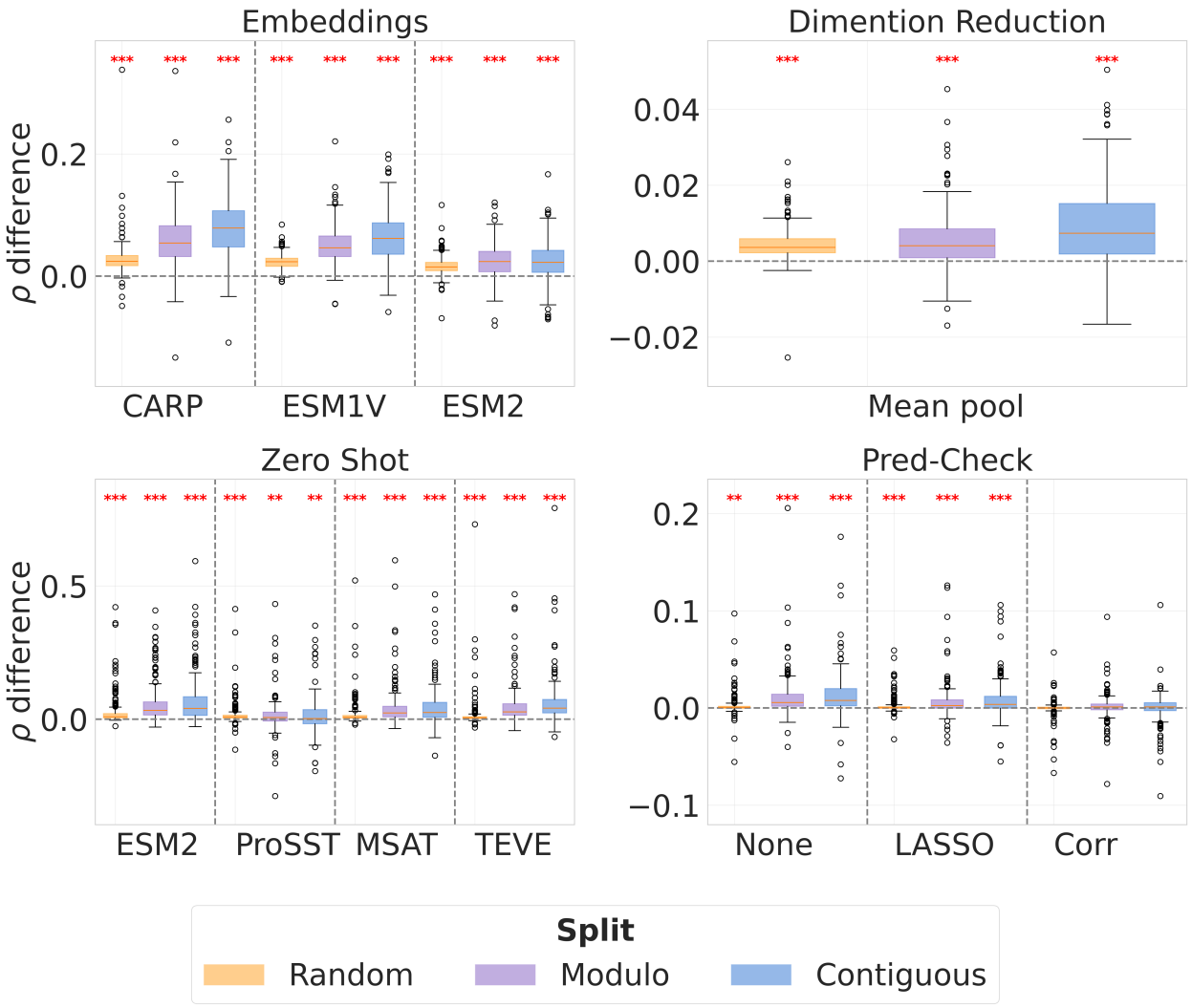

Figure C.2: Each component of SP Bayesian Ridge improves prediction accuracy. Box plots show the change in test set Spearman correlation when ablations are applied to individual components. Asterisks indicate significance levels from one sided two-sample t-tests (*< 0.05, **< 0.01, ***< 0.001), and colors denote different data splits. The top left panel compares SP Bayesian Ridge to a model using only a single embedding; the top right to a model with only mean pooling instead of multiple randomized dimensionality reduction; the bottom left to a model using only a single zero-shot score (MSAT is short for MSA Transformer and TEVE is a short for TranceptEVE); and the bottom right compares the RF-MDI pred-check to alternative pred-check procedures. Ensembling embeddings and random dimensionality reduction both significantly improve performance. The RF-MDI pred-check outperforms each individual zero-shot score and their ensemble, and performs comparably to the correlation-based method (Corr).

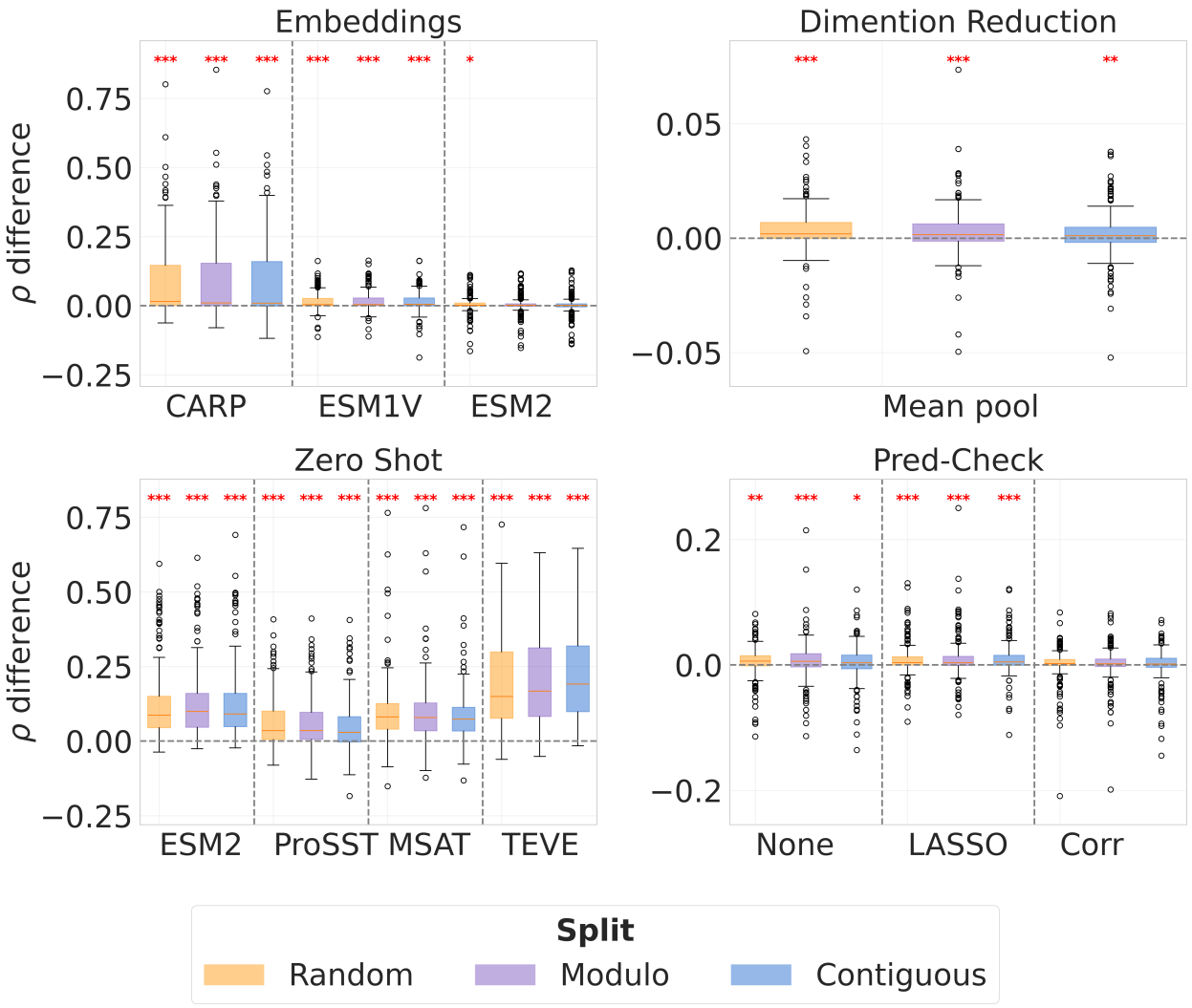

Figure C.3: Each component of SP CNN improves or does not hurt prediction accuracy. Box plots show the change in test set Spearman correlation when ablations are applied to individual components. Asterisks indicate significance levels from one-sided two-sample t-tests (*<0.05, **<0.01, ***<0.001), and colors denote different data splits. The top left panel compares SP CNN to a model using only a single embedding; the top right to a model with only mean pooling instead of multiple randomized dimensionality reduction; the bottom left to a model using only a single zero-shot score (MSAT is short for MSA Transformer and TEVE is a short for TranceptEVE); and the bottom right compares the RF-MDI pred-check to alternative pred-check procedures. Random dimensionality reduction significantly improves performance, while ensembeling of embedding is on par with using ESM2. The RF-MDI pred-check outperforms each individual zero-shot score and their ensemble, and performs comparably to the correlation-based method (Corr).

Studentized Conformal [72] is a simple variant of split conformal designed to produce sets whose width adapts to the input (i.e., m in our setting). The prediction sets are of the form

$$\hat{C}^{\alpha}(m) = [\hat{f}(m) - \hat{\sigma}(m)\hat{s}_{\alpha}, \hat{f}(m) + \hat{\sigma}(m)\hat{s}_{\alpha}], \tag{C.3}$$

where $\hat{s}_{\alpha}$ is defined as the $\lceil (1-\alpha) \times (m+1) \rceil$ smallest value of $\left(\frac{|\hat{f}(m_1) - y_1|}{\hat{\sigma}(m_1)}, \ldots, \frac{|\hat{f}(m_m) - y_m|}{\hat{\sigma}(m_m)} \right)$. Lastly our adaptation of the **PCS UQ** [1] procedure applies a multiplicative calibration creating sets of the form

$$\hat{C}^{\alpha}(m) = [\hat{f}(m) - \hat{\sigma}(m)\hat{\gamma}_{\alpha}, \hat{f}(m) + \hat{\sigma}(m)\hat{\gamma}_{\alpha}], \tag{C.4}$$

where γ is selected to be the smallest values over an equally spaced grid of 50K points between 0 and 500, that achieves $1-\alpha$ coverage on the calibration set.

We emphasize that under the assumptions that test and calibration data point are exchangeable, all the procedures above provide the same marginal coverage guarantees.

C.4.2 Additional results

We provide the results for SP CNN and SP Bayesian Ridge under the same setup described in Section 4.4.2. Figures C.4 and C.5 show that both SP CNN and SP Bayesian Ridge yield shorter average median interval lengths compared to their respective base models. Specifically, for Bayesian Ridge, both conformal methods produce shorter intervals when applied to the SP version than to the base model, whereas the PCS method exhibits high variance. Similarly, SP CNN produces shorter median interval widths than the original CNN.

C.5 Detailed description of fitness predictors

Bayesian Ridge The Bayesian Ridge model assumes that the fitness label follows a Gaussian likelihood given the d-dimensional representation of sequence ($\mathbf{x}(m)$) (which we consider as the concatenation of the mean-pooled embeddings with a zero-shot score):

$$y|\mathbf{x}(m) \sim N\left(\sum_{i=1}^{d} x(m)_i \beta_i, \sigma^2 \right) \tag{C.5}$$

where β_i is the weight of the i-th feature, and σ^2 is the variance of the noise.

The weights are assumed to follow a Gaussian prior:

$$\beta_i|\alpha_i \sim N(0, \alpha_i^{-1}), \tag{C.6}$$

where α_i is the precision parameter for the i-th weight. The model also places gamma priors on α_i and the noise precision $\tau = 1/\sigma^2$:

$$\alpha_i \sim Gamma(a_0, b_0), \quad \tau \sim Gamma(c_0, d_0) \tag{C.7}$$

The hyperparameters a_0, b_0, c_0, and d_0 are set to small values (10^{-6}) following the default scikit-learn [100] implementation.

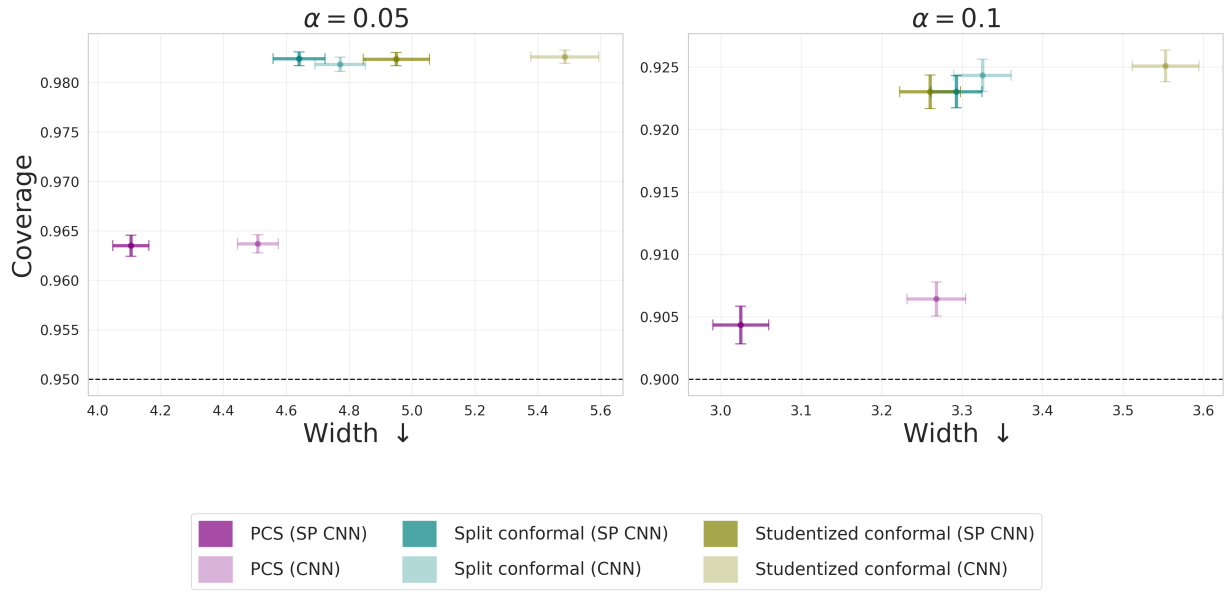

Figure C.4: SP CNN achieves sharper prediction intervals across calibration methods. We report the median interval width and empirical coverage for Kermut and SP Kermut using three prediction interval techniques. The left and right panels correspond to target coverages of 95% and 90%, respectively. In all cases, SP CNN yields narrower intervals while maintaining the desired coverage, indicating improved uncertainty quantification.

Kermut The Kermut method [49] defines a Gaussian Process for distribution of the fitness given the sequence. Its mean function is defined using a zero-shot score, while its kernel is defined as the following product:

$$K_{Kermut}(x, x') = \pi K_{seq}(x, x') + (1 - \pi) K_{struc}(x, x'), \tag{C.8}$$

where $K_{seq}(x, x')$ represents a sequence kernel defined using RBF Kernel applied to mean pooled embeddings of a PLM. and $K_{struc}(x, x')$ is a structure kernel defined as the multiplication of three parts (1) A Hellinger Kernel which is a negative exponential of the Hellinger distance between the inverse folding probabilities between the mutated sites. (2) An exponential kernel in the absolute difference between the log-probabilites of the specific amino acid mutations assinged by an inverse folding model, and (3) a distance kernel which is the negative exponential of the physical distance between the mutation sites. We set $\pi = 0.5$ in our experiments as it is the default values in the Kermut implementation.

CNN Model We use a CNN architecture with the following structure:

- Three 1D convolutional layers with [4, 4, 6] filters respectively, each with kernel size 8

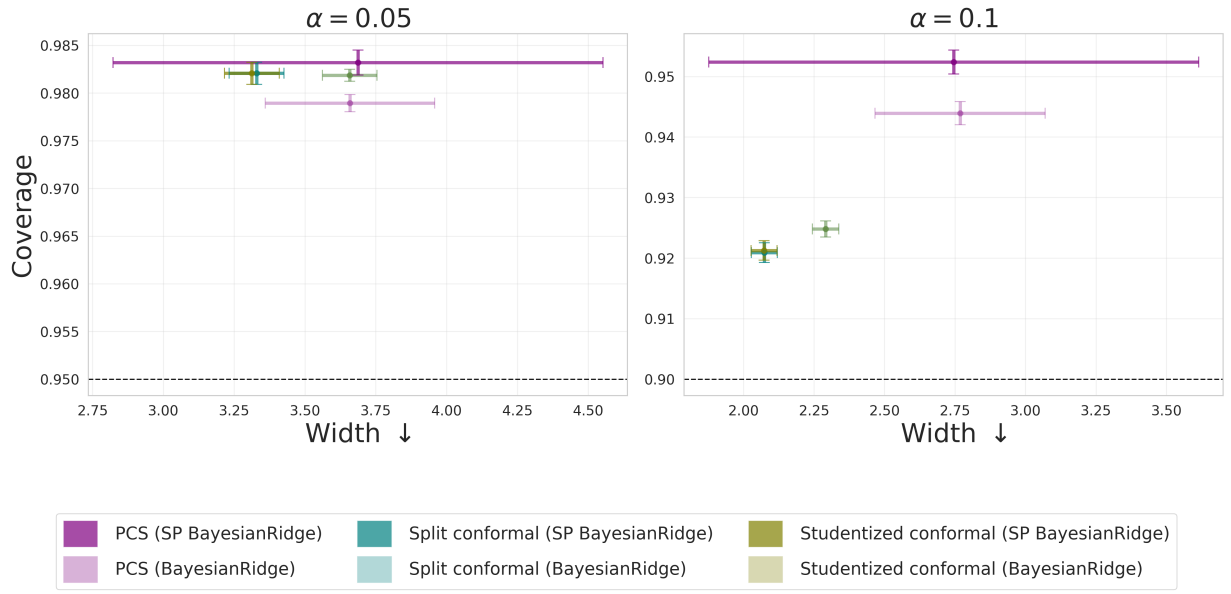

Figure C.5: SP Bayesian Ridge achieves sharper prediction intervals across calibration methods. We report the median interval width and empirical coverage for Kermut and SP Kermut using three prediction interval techniques. The left and right panels correspond to target coverages of 95% and 90%, respectively. In all cases, SP Bayesian Ridge yields narrower intervals while maintaining the desired coverage, indicating improved uncertainty quantification.

- Each conv layer is followed by ReLU activation, batch normalization, and dropout (p=0.1)
- Global average pooling after the final conv layer
- A tanh activation followed by a linear layer that outputs a single value

The model is trained using Adam optimizer with learning rate 1e-3 and batch size 128 for 100 epochs, using the concatenation of the mean-pooled embeddings with a zero-shot score.

C.6 Additional BO results

We report additional BO results. For $\lambda = 0.1$, Figure C.6 shows, at each BO iteration, the cumulative fraction of evaluated sequences whose fitness exceeds the 90th and 70th percentiles, averaged over all datasets, with ± 1,s.d. computed from three independent runs (each initialized with a different random subset). Figure C.8 presents the same analysis for

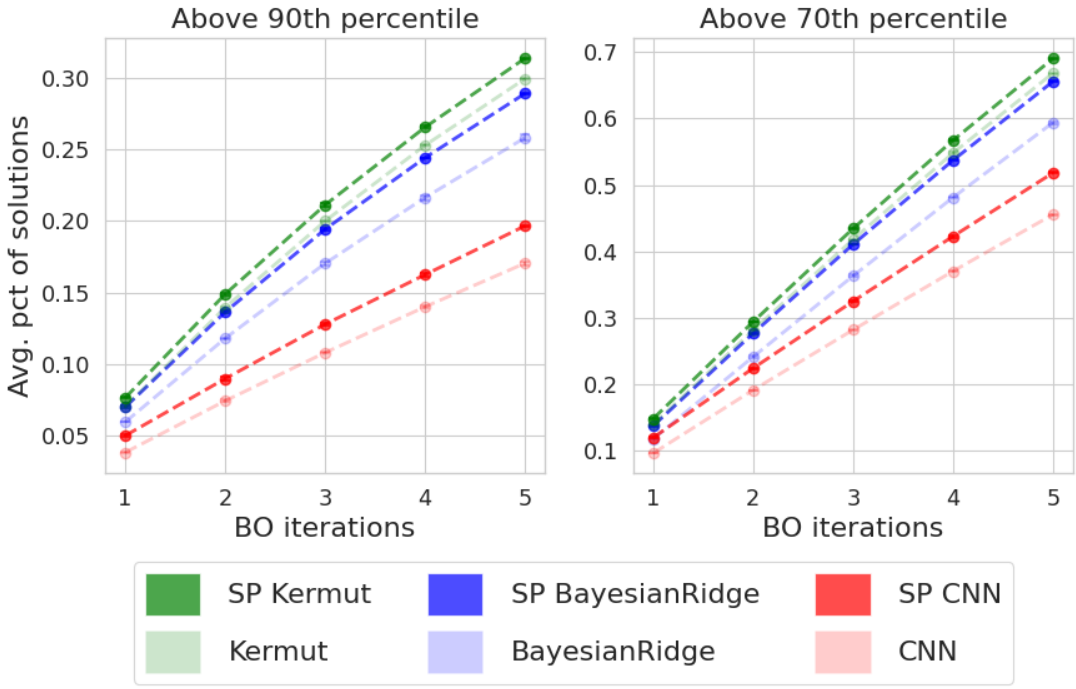

Figure C.6: SP predictors recover high-fitness sequences more often than their base models when using BO with $\lambda = 0.1$. The y-axis shows the cumulative percentage of solutions that are above the 90th (left) and 70th (right) percentiles of the assay's fitness distribution. The x-axis shows the BO step. SP Kermut provides the strongest results under this setting.

$\lambda = 2$. Finally, Figure C.7 plots the running percentage of assays in which at least one highest-fitness sequence has been recovered for $\lambda = 0.1$.

SP variants outperform Kermut (and Bayesian Ridge) on the high-percentile metrics at both $\lambda = 0.1$ and $\lambda = 2$, with the larger gains at $\lambda = 0.1$. For recovery of the highest-fitness sequence, SP Kermut and SP Bayesian Ridge match Kermut at $\lambda = 0.1$, but both show clear improvements at $\lambda = 2$, surpassing their own and Kermut's $\lambda = 0.1$ performance.

C.7 Additional ProteinGym benchmark results

C.7.1 Results on single mutants

We report the numerical results for the random split in Table C.3, modulo split Table C.2 and contiguous split in Table C.1

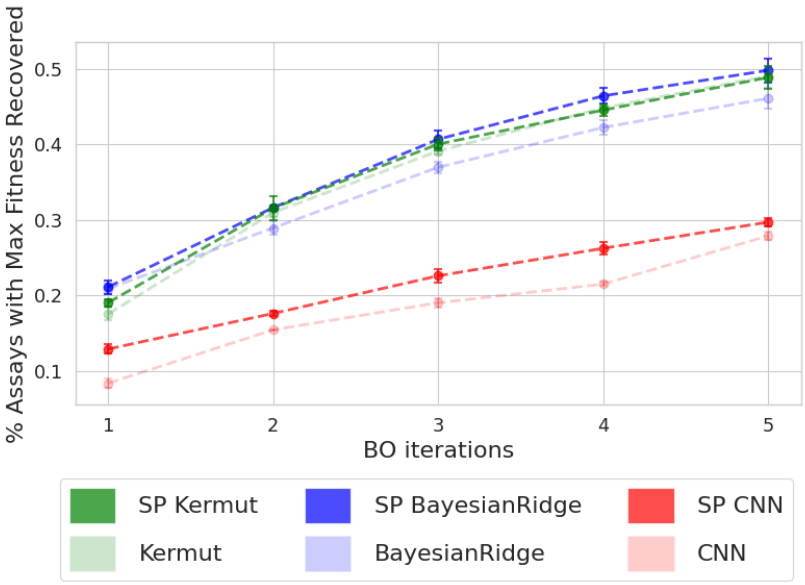

Figure C.7: SP predictors recover the highest-fitness sequence more often—or at least as often—as their base models when using BO with $\lambda = 0.1$. The y-axis shows the percentage of experiments in which the top-fitness sequence is recovered; the x-axis shows the number of BO steps. Results are averaged over three random training set initializations, with error bars showing standard deviation (often too small to be visible). Both SP Kermut and SP Bayesian Ridge perform worse under $\lambda = 0.1$ compared to $\lambda = 2$.

C.7.2 Results on multiple mutants

We consider the same setup as in [49], where we analyze the 51 datasets with less than 7500 sequences (due to Kermut's memory requirements which require storing the covariance matrix). We consider two splitting strategies (1) *random* split where 80% of the data point are assigned to the training set and 20% to the test set and (2) *one vs. two* where all single mutations are assigned to the training set and all double mutations are assigned to the test set. Similar to [49] we find that using zero-shot score hurts performance in the one vs. two setting and does provide significant improvement in the random setting. We therefore report the results for all method without using any zero-shot scores (i.e., mean function of Kermut and SP Kermut is zero).

Results The results are presented in Figure C.9 and Tables C.5 and C.4 for one vs. two and random, respectively. On the challenging one vs. two split, SP Bayesian Ridge provides the strongest spearman results with an average 0.72 compared with 0.69 for Bayesian Ridge and 0.67 for both Kermut and SP Kermut. For the random split SP Kermut achieves a

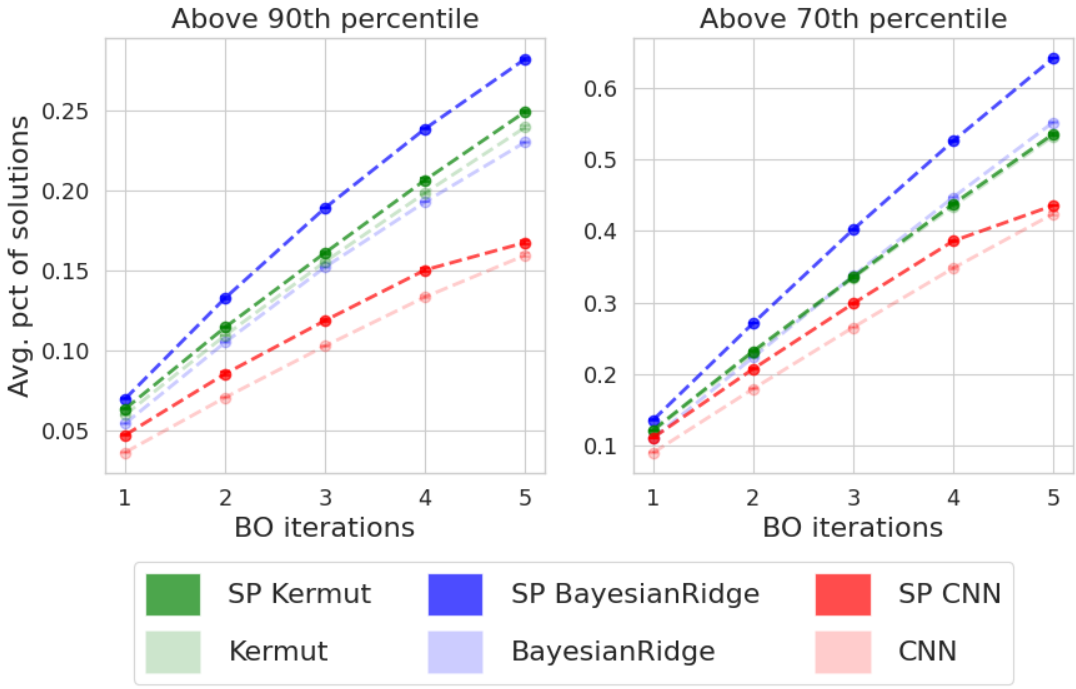

Figure C.8: SP predictors recover high-fitness sequences more often—or at least as often—as their base models when using BO with $\lambda = 2$. The y-axis shows the cumulative percentage of solutions that are above the 90th (left) and 70th (right) percentiles of the assay's fitness distribution. The x-axis shows the BO step. SP Bayesian Ridge provides the strongest results under this setting; however, it underperforms compared to both Kermut and SP Kermut with $\lambda = 0.1$.

spearman value of 0.95 surpassing Kermut with Spearman of 0.94. SP Bayesian Ridge is able to match Kermut's performance with Spearman of 0.94.

On both random and one vs. two splits, SP Kermut and SP Bayesian Ridge predictors provide uncertainty values whose correlation with the model errors are higher compared with their base models.

C.8 Computational Resources

All experiments in this work were carried out using a single A100 GPU.

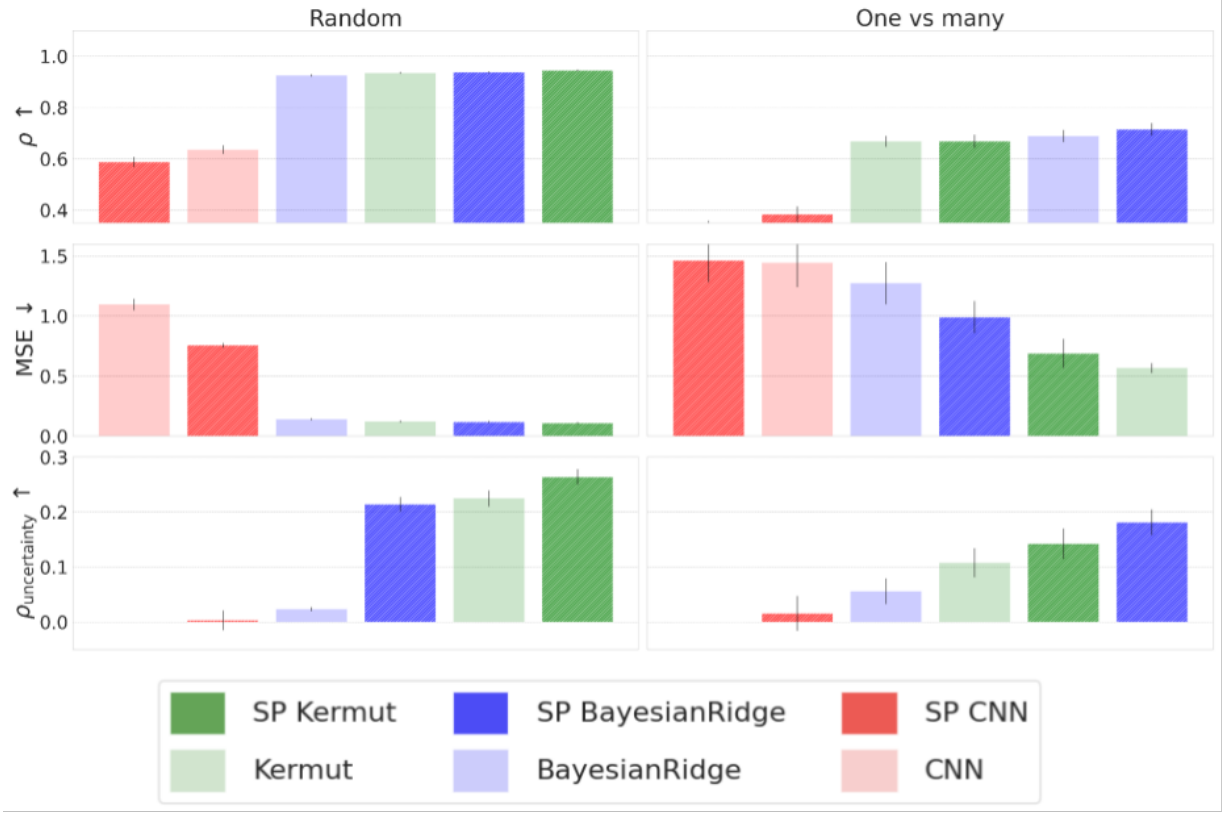

Figure C.9: ProteinGym benchmark results on 51 multiples mutation datasets studied in [49]. We report the average Spearman correlation (ρ), mean squared error (MSE), and the Spearman correlation between the uncertainty scores and the absolute errors of the predictions ($\rho_{\text{uncertainty}}$). Each column corresponds to a different ProteinGym split. SP predictors outperform their base models on Spearman correlation and on the correlation of the uncertainty with the absolute error.

Table C.1: Benchmark results on ProteinGym single-substitution assays for the contiguous train / test split. ProteinNPT results are taken from the ProteinGym website, which does not provide uncertainty estimates.

Model	$\rho \uparrow$	MSE $\downarrow$	$\rho_{uncertainty} \uparrow$
CNN	0.344 ± 0.013	1.113 ± 0.070	0.019 ± 0.006
SP CNN	0.492 ± 0.010	0.916 ± 0.026	0.129 ± 0.008
Bayesian Ridge	0.422 ± 0.014	0.973 ± 0.031	0.008 ± 0.004
SP Bayesian Ridge	0.597 ± 0.011	0.677 ± 0.022	0.153 ± 0.008
Kermut	0.606 ± 0.012	0.688 ± 0.028	0.110 ± 0.008
SP Kermut	0.667 ± 0.010	0.587 ± 0.020	0.164 ± 0.010
ProteinNPT	0.561 ± 0.013	0.784 ± 0.034	-

Table C.2: Benchmark results on ProteinGym single-substitution assays for the modulo train / test split. ProteinNPT results are taken from the ProteinGym website, which does not provide uncertainty estimates.

Model	$\rho \uparrow$	MSE $\downarrow$	$\rho_{uncertainty} \uparrow$
CNN	0.344 ± 0.013	1.113 ± 0.070	0.019 ± 0.006
SP CNN	0.492 ± 0.010	0.916 ± 0.026	0.129 ± 0.008
Bayesian Ridge	0.422 ± 0.014	0.973 ± 0.031	0.008 ± 0.004
SP Bayesian Ridge	0.597 ± 0.011	0.677 ± 0.022	0.153 ± 0.008
Kermut	0.606 ± 0.012	0.688 ± 0.028	0.110 ± 0.008
SP Kermut	0.667 ± 0.010	0.587 ± 0.020	0.164 ± 0.010
ProteinNPT	0.561 ± 0.013	0.784 ± 0.034	-

Table C.3: Benchmark results on ProteinGym single-substitution assays for the random train / test split. ProteinNPT results are taken from the ProteinGym website, which does not provide uncertainty estimates.

Model	$\rho \uparrow$	MSE $\downarrow$	$\rho_{uncertainty} \uparrow$
CNN	0.365 ± 0.012	0.975 ± 0.040	0.013 ± 0.006
SP CNN	0.509 ± 0.010	0.815 ± 0.011	0.133 ± 0.008
Bayesian Ridge	0.693 ± 0.013	0.460 ± 0.020	0.005 ± 0.004
SP Bayesian Ridge	0.755 ± 0.011	0.382 ± 0.016	0.130 ± 0.007
Kermut	0.758 ± 0.012	0.377 ± 0.019	0.106 ± 0.008
SP Kermut	0.785 ± 0.010	0.334 ± 0.016	0.184 ± 0.008
ProteinNPT	0.753 ± 0.013	0.397 ± 0.022	-

Table C.4: Benchmark results on ProteinGym multiple-substitution assays for the random train / test split.

Model	$\rho \uparrow$	MSE $\downarrow$	$\rho_{uncertainty} \uparrow$
CNN	0.423 ± 0.018	1.027 ± 0.037	0.083 ± 0.019
SP CNN	0.588 ± 0.020	0.757 ± 0.020	0.003 ± 0.018
Bayesian Ridge	0.927 ± 0.004	0.140 ± 0.008	0.023 ± 0.004
SP Bayesian Ridge	0.939 ± 0.004	0.118 ± 0.008	0.214 ± 0.013
Kermut	0.935 ± 0.005	0.129 ± 0.014	0.203 ± 0.015
SP Kermut	0.945 ± 0.004	0.110 ± 0.008	0.264 ± 0.014

Table C.5: Benchmark results on ProteinGym multiple-substitution assays for the one vs. two train / test split.

Model	$\rho \uparrow$	MSE $\downarrow$	$\rho_{uncertainty} \uparrow$
CNN	0.324 ± 0.030	1.417 ± 0.203	0.021 ± 0.031
SP CNN	0.384 ± 0.030	1.463 ± 0.181	0.015 ± 0.032
Bayesian Ridge	0.678 ± 0.028	1.189 ± 0.157	0.057 ± 0.023
SP Bayesian Ridge	0.715 ± 0.024	0.991 ± 0.133	0.182 ± 0.024
Kermut	0.666 ± 0.022	0.666 ± 0.117	0.124 ± 0.024
SP Kermut	0.669 ± 0.026	0.689 ± 0.121	0.143 ± 0.027

Appendix D

Supplementary Material for Chapter 5

D.1 Summary Statistics for Filtering and Interactions

Table D.1 provides a summary of the number of genes filter at each step and the number of interactions in each database.

Table D.1: Summary of genes and interactions for each phenotype. Associated genes are the number of genome-wide significant hits extracted from Clarke et al. [28]. Predictive genes describes the size of the subset of Associated genes which passed the prediction screen. Total interaction is the total pairwise combinations of the predictive genes. BioGRID is the total number of interactions reported in the BioGRID database, and STRING is the number of reported interactions in STRING with different confidence levels (0.9, 0.7, 0.4).

Phenotype	**Associated Genes**	**Predictive Genes**	**Total Interactions**	**BioGRID**	**STRING (.9/.7/.4)**
Urate	48	16	120	1	3 / 5 / 11
ApoB	53	10	45	2	3 / 5 / 10
MPTVS	157	28	378	8	3 / 13 / 23
LDL Direct	40	11	55	7	10 / 19 / 30
Standing Height	211	20	190	0	0 / 1 / 2
Platelet Dist. Width	84	11	55	1	1 / 3 / 8
IGF-1	56	8	28	3	3 / 3 / 5
Mean Retic. Vol.	86	10	45	2	3 / 4 / 6
Lymphocyte Pct.	51	10	45	0	0 / 2 / 6
Calcium	24	5	10	0	1 / 1 / 3
SHBG	55	10	45	0	0 / 0 / 2
HDL Cholesterol	61	14	91	3	13 / 25 / 39
Erythrocyte Count	83	11	55	0	0 / 0 / 1
Platelet Crit	70	11	55	2	5 / 10 / 13
Mean Corp. Vol.	116	12	66	0	1 / 1 / 2
Neutrophil Count	56	10	45	2	2 / 7 / 9
Cholesterol	49	11	55	6	9 / 17 / 30
Triglycerides	54	6	15	0	4 / 5 / 6
Platelet Count	91	4	6	0	0 / 0 / 1
ApoA	56	14	91	4	18 / 30 / 48
Total	**1563**	**255**	**1748**	**48**	**101 / 187 / 315**

D.2 Ablation Study of the SNP Representations and Aggregation Methods

Table D.2 provides the AUROC scores presented in section 5.3.2, across the different choice of SNP representations and the different choices of aggregation functions for combining these representations.

Table D.2: Ablation study for AUC scores across databases. For US-iRF we ablate the choice of the mean function for aggregation against the max and min, as well as every single SNP representation. For US-MatrixEpistasis we consider every choice of the representation, but only the Cauchy aggregregation which preserves the null distribution of the p-value. Note that the US-MatrixEpistasis (dose) applies PCA to the dose matrix prior to the analysis unlike MatrixEpistasis.

	STRING (0.4)	STRING (0.7)	STRING (0.9)	BioGRID
US-iRF (mean)	0.780	0.785	0.784	0.677
US-iRF (max)	0.776	0.777	0.772	0.664
US-iRF (ESM1v)	0.726	0.730	0.720	0.629
iRF (dose)	0.706	0.693	0.701	0.644
US-iRF (ESM2)	0.691	0.695	0.732	0.649
US-iRF (CARP)	0.691	0.706	0.745	0.644
US-iRF (min)	0.645	0.651	0.664	0.601
MatrixEpistasis	0.585	0.602	0.590	0.647
US-MatrixEpistasis (esm2)	0.510	0.514	0.520	0.551
US-MatrixEpistasis (gwas)	0.509	0.509	0.506	0.513
US-MatrixEpistasis (carp)	0.500	0.499	0.463	0.527
US-MatrixEpistasis (cauchy)	0.499	0.490	0.485	0.547
US-MatrixEpistasis (esm1v)	0.497	0.495	0.488	0.532

D.3 Phenotype-Level Results for Section 5.3.2

We provide AUROC results stratified at a phenotype level to complement the pooled analysis in Section 5.3.2.

D.4 Detailed Results for Section 5.3.3

For each database, Tables D.7 to D.10 reports the number of discoveries and the number of true discoveries for each method in the analysis described in Section 5.3.3.

Phenotype / Method	US-iRF	iRF (dose)	US-MatrixEpistasis	MatrixEpistasis
Urate	0.651	0.332	0.681	**0.857**
ApoB	0.977	**0.988**	0.640	0.186
MPTVS	0.554	0.530	0.501	**0.595**
LDL Direct	0.580	0.571	0.546	**0.747**
Standing Height	–	–	–	–
Platelet Dist. Width	0.463	0.519	**0.611**	0.556
IGF-1	**0.920**	0.827	0.907	0.747
Mean Retic. Vol.	**0.872**	0.814	0.802	0.442
Lymphocyte Pct.	–	–	–	–
Calcium	–	–	–	–
SHBG	–	–	–	–
HDL Cholesterol	**0.818**	0.814	0.422	0.617
Erythrocyte Count	–	–	–	–
Platelet Crit	**0.991**	0.915	0.915	0.887
Mean Corp. Vol.	–	–	–	–
Neutrophil Count	**0.733**	0.523	0.465	0.512
Cholesterol	0.466	**0.526**	0.221	0.493
Triglycerides	–	–	–	–
Platelet Count	–	–	–	–
ApoA	**0.641**	0.333	0.602	0.323

Table D.3: AUC scores for different interaction discovery methods across phenotypes using BioGRID as ground truth. Higher AUC values indicate better performance in identifying true protein-protein interactions.

Phenotype / Method	US-iRF	iRF (dose)	US-MatrixEpistasis	MatrixEpistasis
Urate	**0.749**	0.648	0.586	0.472
ApoB	**0.717**	0.660	0.554	0.326
MPTVS	**0.801**	0.773	0.510	0.556
LDL Direct	**0.681**	0.640	0.467	0.500
Standing Height	**0.862**	0.856	0.710	0.707
Platelet Dist. Width	**0.781**	0.741	0.572	0.468
IGF-1	**0.965**	0.896	0.696	0.565
Mean Retic. Vol.	**0.825**	0.782	0.697	0.573
Lymphocyte Pct.	0.821	**0.829**	0.571	0.553
Calcium	**0.905**	0.690	0.762	0.476
SHBG	0.500	0.459	0.407	**0.628**
HDL Cholesterol	**0.841**	0.655	0.428	0.597
Erythrocyte Count	0.093	0.315	**0.722**	0.667
Platelet Crit	**0.764**	0.562	0.511	0.531
Mean Corp. Vol.	0.969	**1.000**	0.508	0.543
Neutrophil Count	**0.662**	0.630	0.505	0.343
Cholesterol	**0.673**	0.636	0.332	0.537
Triglycerides	0.870	**0.907**	0.176	0.306
Platelet Count	**1.000**	1.000	0.600	0.800
ApoA	**0.923**	0.763	0.498	0.516

Table D.4: AUC scores for different interaction discovery methods across phenotypes using STRING (threshold = 0.4) as ground truth. Higher AUC values indicate better performance in identifying functional associations.

Phenotype / Method	US-iRF	iRF (dose)	US-MatrixEpistasis	MatrixEpistasis
Urate	**0.689**	0.578	0.465	0.651
ApoB	**0.800**	0.745	0.760	0.185
MPTVS	**0.739**	0.738	0.447	0.578
LDL Direct	**0.702**	0.611	0.404	0.469
Standing Height	0.836	**0.910**	0.439	0.820
Platelet Dist. Width	**0.606**	0.522	0.391	0.429
IGF-1	**0.920**	0.827	0.907	0.747
Mean Retic. Vol.	**0.768**	0.671	0.604	0.616
Lymphocyte Pct.	**0.953**	0.913	0.512	0.733
Calcium	0.778	**1.000**	0.778	0.000
SHBG	–	–	–	–
HDL Cholesterol	**0.808**	0.645	0.509	0.650
Erythrocyte Count	–	–	–	–
Platelet Crit	**0.762**	0.504	0.627	0.616
Mean Corp. Vol.	0.969	0.992	0.938	**1.000**
Neutrophil Count	0.733	**0.741**	0.502	0.335
Cholesterol	**0.643**	0.618	0.348	0.543
Triglycerides	0.900	**0.920**	0.260	0.400
Platelet Count	–	–	–	–
ApoA	**0.867**	0.697	0.479	0.496

Table D.5: AUC scores for different interaction discovery methods across phenotypes using STRING (threshold = 0.7) as ground truth. Higher AUC values indicate better performance in identifying functional associations.

Phenotype / Method	US-iRF	iRF (dose)	US-MatrixEpistasis	MatrixEpistasis
Urate	0.839	0.742	0.598	**0.849**
ApoB	**1.000**	1.000	0.746	0.246
MPTVS	**0.823**	0.759	0.198	0.564
LDL Direct	**0.633**	0.598	0.562	0.444
Standing Height	–	–	–	–
Platelet Dist. Width	0.407	**0.463**	0.093	0.148
IGF-1	**0.920**	0.827	0.907	0.747
Mean Retic. Vol.	**0.825**	0.786	0.659	0.492
Lymphocyte Pct.	–	–	–	–
Calcium	0.778	**1.000**	0.778	0.000
SHBG	–	–	–	–
HDL Cholesterol	**0.744**	0.608	0.449	0.581
Erythrocyte Count	–	–	–	–
Platelet Crit	**0.720**	0.532	0.544	0.712
Mean Corp. Vol.	0.969	0.992	0.938	**1.000**
Neutrophil Count	0.512	**0.663**	0.273	0.116
Cholesterol	0.540	**0.579**	0.353	0.512
Triglycerides	**0.932**	0.886	0.352	0.364
Platelet Count	–	–	–	–
ApoA	**0.817**	0.615	0.441	0.456

Table D.6: AUC scores for different interaction discovery methods across phenotypes using STRING (threshold = 0.9) as ground truth. Higher AUC values indicate better performance in identifying functional associations.

Method / Top Pct. Interactions	**1**	**2**	**4**	**5**
US-iRF	(4 / 21)	(5 / 41)	(8 / 62)	(11 / 82)
iRF (dose)	(5 / 57)	(5 / 57)	(5 / 68)	(6 / 83)
US-MatrixEpistasis	(0 / 21)	(0 / 41)	(0 / 62)	(1 / 82)
MatrixEpistasis	(1 / 21)	(1 / 41)	(2 / 62)	(4 / 82)
iRF (ESM1v)	(3 / 56)	(3 / 56)	(3 / 67)	(4 / 85)
iRF (CARP)	(8 / 55)	(8 / 55)	(9 / 68)	(9 / 82)
iRF (ESM2)	(8 / 68)	(8 / 68)	(8 / 68)	(9 / 82)

Table D.7: Number of true interactions and total interactions found by each method when selecting the top interactions at different percentages using BioGRID as ground truth. Format: (true interactions / total interactions).

Method / Top Pct. Interactions	**10**	**20**	**31**	**41**
US-iRF	(75 / 128)	(119 / 255)	(149 / 383)	(177 / 510)
iRF (dose)	(66 / 130)	(104 / 255)	(136 / 383)	(163 / 510)
US-MatrixEpistasis	(23 / 128)	(44 / 255)	(64 / 383)	(88 / 510)
MatrixEpistasis	(31 / 128)	(61 / 255)	(95 / 383)	(122 / 510)
iRF (ESM1v)	(70 / 128)	(115 / 255)	(146 / 383)	(167 / 510)
iRF (CARP)	(62 / 128)	(110 / 255)	(139 / 384)	(255 / 1495)
iRF (ESM2)	(70 / 129)	(104 / 255)	(128 / 383)	(255 / 1495)

Table D.8: Number of true interactions and total interactions found by each method when selecting the top interactions at different percentages using STRING (threshold = 0.4) as ground truth. Format: (true interactions / total interactions).

Method / Top Pct. Interactions	**6**	**12**	**18**	**24**
US-iRF	(35 / 76)	(53 / 151)	(67 / 227)	(79 / 302)
iRF (dose)	(28 / 76)	(44 / 151)	(57 / 227)	(70 / 302)
US-MatrixEpistasis	(6 / 76)	(10 / 151)	(19 / 227)	(28 / 302)
MatrixEpistasis	(10 / 76)	(25 / 151)	(38 / 227)	(53 / 302)
iRF (ESM1v)	(27 / 78)	(47 / 151)	(62 / 227)	(79 / 302)
iRF (CARP)	(30 / 76)	(44 / 151)	(64 / 227)	(79 / 302)
iRF (ESM2)	(30 / 78)	(50 / 152)	(61 / 228)	(74 / 302)

Table D.9: Number of true interactions and total interactions found by each method when selecting the top interactions at different percentages using STRING (threshold = 0.7) as ground truth. Format: (true interactions / total interactions).

Method / Top Pct. Interactions	**3**	**6**	**10**	**13**
US-iRF	(13 / 40)	(26 / 79)	(29 / 119)	(34 / 158)
iRF (dose)	(11 / 57)	(17 / 79)	(23 / 119)	(27 / 158)
US-MatrixEpistasis	(0 / 40)	(2 / 79)	(3 / 119)	(4 / 158)
MatrixEpistasis	(2 / 40)	(5 / 79)	(11 / 119)	(15 / 158)
iRF (ESM1v)	(12 / 56)	(19 / 80)	(24 / 119)	(30 / 158)
iRF (CARP)	(12 / 55)	(17 / 80)	(22 / 119)	(26 / 158)
iRF (ESM2)	(17 / 68)	(19 / 81)	(29 / 119)	(35 / 158)

Table D.10: Number of true interactions and total interactions found by each method when selecting the top interactions at different percentages using STRING (threshold = 0.9) as ground truth. Format: (true interactions / total interactions).