# Exploring evolutionary protein fitness landscapes

**Ayush Karnawat**
Department of Electrical Engineering and Computer Science
Case Western Reserve University, Cleveland, OH 44106
axk840@case.edu

## Abstract

The ability to efficiently design protein for a specific purpose – namely for therapeutic and other industrial applications – has long been the dream of chemical and bioengineers. Conventional experimental techniques such as directed evolution, while highly percise, are unable to find rare, enhanced variants of the protein of interest. Here, we investigate how a semi-supervised adaptive sampling approach can be used to effectively explore the search space, in order to improve protein design. Unlike other black-box predictive models, which are known to suffer from bias introduced in the training distribution, our model does not directly optimize the oracle to improve its predictive power. Rather, it estimates a distribution over the design space conditioned on the desired properties. Using the streptococcal protein G's $\beta 1$ (GB1) domain as an illustrative example, we demonstrate how machine-guided protein design can efficiently navigate the landscape to locate non-trivial high-fitness protein designs. Overall, our approach is a first step toward enabling efficient use of expensive laboratory experiments without sacrificing throughput.

## 1 Introduction

Proteins are the inherent building blocks of all life, and play a highly critical role as molecular machines in cells by sensing, signaling, and catalyzing chemical reactions. Due to their extensive design capabilities [1, 2] – such as to maximize binding affinity to a target – they have been subject to a lot of trail and error experimentation, which is not only a time-consuming, but also costly procedure. In fact, according to [3], it takes an average of 12 years and $\sim$ \$2.6 billion USD to design a effective drug, with costs rising each year [4].

Recent advances in computational protein design have showed great promise in this regard by creating novel proteins *de novo* [5]. However, a closer inspection reveals its design range is (currently) limited to simple folds and static structures, which is only a small subset of proteins in the vast design space [6]. As such, improving design through directed evolution, a technique introduced in [7] and further popularized by [8], is still used today. Here, in a iterative manner, random mutations (chosen uniformly from the search space) from a parent population of proteins are induced, properties of interest are measured, and the top $k$ variants are chosen for the next round of mutations. While this approach has high-fidelity, it is (a) dependent on resource intensive assays, and (b) unlikely to be efficient in finding the best design from the search space.

One way to improve this is to apply computational methods to certain aspects of the design cycle [9, 10, 11, 12, 13]. For example, a trivial place to employ a purely computational approach is to replace the laboratory assays with an "oracle" that can accurately predict the property of interest. However, this relies on the regression oracle being well-behaved, even in regions of the search space far from the initial training dataset. Yet, recent work has shown how brittle state-of-the-art (SOTA) image [14, 15] and text [16, 17] classification models can be due to pathological biases introduced by the training distribution, especially for inputs not in the original training set [18, 19]. As such, approaches that directly optimize the predictive oracle can be led astray into regions of the search space that are poor, resulting in unreliable sequences, i.e. proteins that are intrinsically unfolded or unable to fold properly. To avoid this issue, one can use an optimization scheme that incorporates prior information using a bayesian approach [20, 21]. This can be seen as an way to encode knowledge about either (a) the regions where the oracle is expected to be accurate (i.e. by learning the distribution of the training dataset), or (b) what constitutes a "realistic" input (i.e. by learning representations of proteins known to stably fold).

Preprint. Under review.

A (potentially) more well-suited modification to the directed evolution method, which involves optimizing how the search space is traversed, is likely to find better designs. Although directed evolution is adept at searching and climbing nearby peaks within a local neighborhood [8], it is not conducive to finding functional neighborhoods far away from the original parent sequence [22], due to the fact that many randomly sampled sequences are non-functional [23, 24]. This is further exacerbated by the fact that, as the length $L$ of the protein increases, the design space becomes exponentially vast (scaling $\Theta(20^L)$), making an exhaustive random search intractable. There is, hence, an opportunity to use a combinatorial optimization technique to explore the design space more efficiently [25, 26], while also taking into account the exploration-exploitation (EvE) tradeoff [27, 28].

A class of methods, based on variants of the Monte Carlo (MC) approach, that account for the EvE tradeoff have been used extensively to explore large search spaces (e.g. Scrabble words [29] or Go moves [30, 31]) to a high degree of effectiveness. However, performing the tree search requires that the evalutation be deterministic for a specific state of the positions [32]. Contrarily, in our design task, – and more generally for most real-world design problems – the evaluation function is typically a stochastic approximation of the ground truth. Therefore, we require that the uncertainty be leveraged within our method to achieve proper optimization [33].

One way to ensure this is to formulate the approach using evolution strategy (ES) algorithms [34], which are known to handle a wide range of uncertainties [35], such as changes to the environment (i.e. design space) or noisy "oracle" function evaluations. In particular, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [36] allows for black-box optimization of non-linear non-convex functions, such as a protein's rugged evolutionary landscape [37]. Here, new candidate sets are sampled according to a multivariate gaussian distribution in $\mathbb{R}^n$. After the points are evaluated, the mean ($\mu$) and the variance ($\sigma^2$) of the mutivariate gaussian are updated to provide (potentially better) samples for the next iteration. This is repeated until either (1) no feasible set of samples is found or (2) a tolerance criteria is met between the changing $\mu, \sigma^2$. Akin to CMA-ES, we instead use a SOTA optimization algorithm based on [33], which extends beyond the multivariate gaussian to a more general set of underlying generative models. This allows us to use, for example, Variational Autoencoders (VAEs) [38] or Generative Adverserial Networks (GANs) [39] to more effectively generate plausible sequences from the search space.

As alluded to previously, the biggest benefit this approach brings over a conventional optimization technique is that it saves resources (in both time and money) since we don't waste time querying[1] poor regions of the search space [40]. Additionally, since the fitness evaluation function is derivative-free, i.e. a black box that provides a simple input to ouput mapping, a wet-lab protein assay can easily be used inplace of the oracle[2], to more accuratey optimize the design. This allows our method to be scalable in a real-world protein design context.

Overall, in this work, we explore the benefits and drawbacks of using an adaptive sampling technique based on a bayesian inference framework to navigate the protein's fitness landscape. We think this approach, owning to the use of a SOTA optimization technique, can considerable accelerate the evolutionary design process, thus reducing resource intensive tasks, such as assay preparation and quantification. We demonstrate our ideas empirically using protein variants of the $\beta 1$ domain of the G protein as a test case [42]. Here, the fitness is a proxy for the stability (i.e. the fraction of folded proteins) and function (i.e. binding affinity to IgG-Fc) of the domain. We hope to inspire a hybrid approach where the adaptive sampling portion of the optimization strategy is used in conjunction with actual wetlab results to gain an accurate picture of the protein fitness landscape.

## 2 Methods

### 2.1 Problem definition

We consider the problem of finding instances of an $L$-dimensional random vector (i.e. a protein sequence) that satisfies a certain design criteria. The criteria is highly problem dependent; most commonly, we are concerned with the maximization (i.e. finding a protein that is maximally fit) or the specification property (i.e. designing a protein that has a certain binding affinity). In our formulation, since proteins are just sequences defined by a vocab, we assume that our dataset $X$ is discrete with samples $x \in \mathbb{N}^L$. However, as will be evident from the mathematical description, we can also work with the structural representation of proteins (represented as $n$-order tensors) with realizations $x \in \mathbb{R}^{I_1 \times \cdots \times I_n}$.

To build our predictive models, we assume access to realizations $x \in X$ drawn from an unknown underlying data distribution, $p_D(x)$, denoted as the training set. Our hope is that this will help us build a certain class of generative

[1] We do not want to keep exploring the search space if there is no benefit from doing so.

[2] Careful consideration should be placed on the inherent variance induced by different assay types [41]. In fact, by design, the optimization procedure uses this variability to address the concern that the oracle is a stochastic approximation of the "true" fitness value. For further detail see §2.

models, $p(x|\theta)$, that can approximate $p_D(x)$ well[3]. We define the parameters of this model by $\theta^{(0)}$ after it has been fit to the data, which yields our prior density $p(x|\theta^{(0)})$.

In addition to a generative model, we assume that, to properly optimize how the search space is travered, we are given a predictor $p(y|x)$ (hereby named an "oracle"). This provides a distribution over the property of interest $y$ given a particular realization $x \in X$. From this oracle, we should be able to calculate the probability of events contained in $S$ occuring. Note that $S$ changes depending on the problem criteria. In particular, for maximization problems, $S$ contains the set of values $y$ such that $y \geq y_{max} \equiv \max_x \mathbb{E}_{p(y|x)}[y]$. Less stringently, following [43], we consider the set $S = \{y|y \geq y^* \leq y_{max}\}$, where $y^*$ is some predefined threshold or taken to be the quantile of the data. Regardless of the how the design criteria is specified, our ultimate aim is to condition the prior density on our desired property values $p(S|x, \theta^{(0)})$ and sample from the resulting distribution. This will ensure that the generated samples are realistic samples (i.e. variants found in nature) assuming that the resulting conditional can be well-approximated using an accurate generative model $q(x|\phi)$. As the optimizaton procedure evolves, the parameters $\phi$ will move towards values that approximate the desired conditional density well.

### 2.2 Generative model optimization

Since we are interested in the maximization property for our experiments, we focus here on maximizing our expected probability, $P(S|\theta)$, over the property values. Note that a similar formulation can be derived to optimize the probabilty for the specification problem. More specifically, we solve the following optimization problem,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log P(S|\theta) \tag{1}$$

$$= \underset{\theta}{\operatorname{argmax}} \log \int p(S|x)p(x|\theta)dx \tag{2}$$

$$= \underset{\theta}{\operatorname{argmax}} \log \mathbb{E}_{p(x|\theta)}[P(S|x)] \tag{3}$$

which, if all the underlying inputs are well-behaved, would allow us to easily find our "desired" input $\hat{x}$ from the posterior distribution, i.e. $\hat{x} \sim p(x|\hat{\theta})$. However, the biggest problem with this approach lies in the fact that our desired property $S$, by nature, is a rare condition, thereby making the probability $P(S|\theta)$ extremely small for most parameters $\theta$ of the generative model. This makes it hard to optimize and can result in high-variance predictions from the "oracle", especially when there are not enough samples in the training set.

To remedy this, we begin by iteratively optimizing (3) using $\theta^{(t)}$, where, at each iteration $t$, the generative model $p(x|\theta^{(t)})$ becomes closer and closer to the desired conditional distribution. In fact, to estimate our conditional, we make use of our prior model conditioned onto the set of values $S$ using Bayes' theorem,

$$p(x|S, \theta^{(0)}) = \frac{P(S|x)p(x|\theta^{(0)})}{P(S|\theta^{(0)})} \tag{4}$$

where $P(S|\theta^{(0)}) = \int P(S|x)p(x|\theta^{(0)})dx$. Then, to find the ideal set of parameters $\phi^*$ for the generative model (which acts as the "search" model), we minimize the KL divergence between the target conditional $p(x|S, \theta^{(0)})$ and the search model, $q(x|\phi)$,

$$\phi^* = \underset{\phi}{\operatorname{argmin}} D_{KL}\left[p(x|S, \theta^{(0)})||q(x|\phi)\right] \tag{5}$$

$$= \underset{\phi}{\operatorname{argmin}} \int p(x|S, \theta^{(0)}) \log \left[\frac{p(x|S, \theta^{(0)})}{q(x|\phi)}\right] dx \tag{6}$$

$$= \underset{\phi}{\operatorname{argmin}} \underbrace{\int p(x|S, \theta^{(0)}) \log[p(x|S, \theta^{(0)})]dx}_{H_0} - \int p(x|S, \theta^{(0)}) \log[q(x|\phi)]dx \tag{7}$$

$$= \underset{\phi}{\operatorname{argmax}} \int p(x|S, \theta^{(0)}) \log[q(x|\phi)]dx \tag{8}$$

$$= \underset{\phi}{\operatorname{argmax}} \int \frac{P(S|x)p(x|\theta^{(0)})}{P(S|\theta^{(0)})} \log[q(x|\phi)]dx \tag{9}$$

[3]Here, "well" is a particulary fuzzy term; by it we mean that the samples generated are similar to the training set. A more mathematically accurate definition can be found in §2.2.

$$= \underset{\phi}{\operatorname{argmax}} \frac{1}{P(S|\theta^{(0)})} \int p(S|x)p(x|\theta^{(0)}) \log[q(x|\phi)]dx \tag{10}$$

$$= \underset{\phi}{\operatorname{argmax}} \mathbb{E}_{p(x|\theta^{(0)})}[p(S|x) \log q(x|\phi)] \tag{11}$$

Note that $P(S|\theta^{(0)})$ and $H_0$ are removed from the overall objective since they do not rely on the optimization parameter $\phi$. In most cases, (11) is the function we want to optimize, but, as mentioned previously, it doesn't take into account if $S$ is a rare property. To address this, we use an importance sampling distribution, where rare samples are given higher probability versus more common samples, and rewrite our objective function as,

$$\underset{\phi}{\operatorname{argmax}} \mathbb{E}_{r(x)} \left[ \frac{p(x|\theta^{(0)})}{r(x)} p(S|x) \log q(x|\phi) \right] \tag{12}$$

$$\underset{\phi}{\operatorname{argmax}} \mathbb{E}_{r^{(t)}(x)} \left[ \frac{p(x|\theta^{(0)})}{r^{(t)}(x)} p(S^{(t)}|x) \log q(x|\phi) \right] \tag{13}$$

by adding an iteration dependence $t$ on both $S^{(t)}$ and $r^{(t)}(x)$. This ensures that $\mathbb{E}_{r^{(t)}(x)}[P(S^{(t)}|x)]$ has non-vanishing probability, meaning that we can draw samples $x \sim r^{(t)}(x)$ that have reasonably high values of $P(S^{(t)}|x)$. Additionally, this allows for $S^{(t)}$ to approach $S$ as $t$ grows large ($S^{(t)} \supset S^{(t+1)} \supset \cdots \supset S$).

For the first iteration ($t = 0$) of the algorithm, we set (a) $S^{(0)}$ to be the property values that are greater than the $Q^{\text{th}}$ percentile of the oracle values predicted for the training sample, i.e. $S^{(0)} = p(y \geq \gamma^{(0)}|x)$ and (b) the importance sampling distribution to the prior, i.e. $r^{(0)}(x) = p(x|\theta^{(0)})$. This helps us build the approximate conditional $q(x|\phi^{(0)})$. Similarly, for the remaining iterations, we set (a) $\gamma^{(t)}$ to the $Q^{\text{th}}$ percentile of the property values of the samples $x^{(t)}$ generated from the previous iterations search distribution $q(x|\phi^{(t-1)})$, and (b) $r^{(t)} = q(x|\phi^{(t-1)})$. Substituting these in, (13) becomes

$$\phi^{(t+1)} = \underset{\phi}{\operatorname{argmax}} \mathbb{E}_{q(x|\phi^{(t)})} \left[ \frac{p(x|\theta^{(0)})}{q(x|\phi^{(t)})} P(S^{(t)}|x) \log q(x|\phi) \right] \tag{14}$$

which can be approximated by drawing $M$ random samples from the search model, $x_i^{(t)} \sim q(x|\phi^{(t)})$ for $i = 1, \ldots, M$,

$$\phi^{(t+1)} = \underset{\phi}{\operatorname{argmax}} \sum_{i=1}^{M} \frac{p(x_i^{(t)}|\theta^{(0)})}{q(x_i^{(t)}|\phi^{(t)})} P(S^{(t)}|x_i^{(t)}) \log q(x_i^{(t)}|\phi) \tag{15}$$

It is necessary in this approximation that the property values $P(S^{(t)}|x)$ be non-zero and the ratio $\frac{p(x|\theta^{(0)})}{q(x|\phi^{(t)})}$ well-behaved. Luckily, this is satisfied by the weighting scheme itself, which ensures that the density of the search model, $q(x|\phi)$, is close to the prior distribution, $p(x|\theta^{(0)})$, through minimization of the KL divergence as shown in (5).

### 2.3 Connection to latent variable models

The objective presented in (15) requires us to accurately calculate the density of the prior, $p(x|\theta^{(0)})$, and the search model, $q(x|\phi)$, for any input $x$. For certain classes of latent variable models this is not possible, since marginalization over the latent space is intractable. However, if both the densities are defined on the same latent space, $z$, then one can evaluate the joint densities of the observed and latent variables by computing $p(x, z|\theta^{(0)})$ and $q(x, z|\phi^{(t)})$. This is possible due to the well-known fact that an expectation over the marginal is equal to the expectation over the joint, i.e. $\mathbb{E}_{p(x)}[f(x)] = \mathbb{E}_{p(x,y)}[f(x)]$. Using this result, we can reformulate our objective as,

$$\phi^{(t+1)} = \underset{\phi}{\operatorname{argmax}} \mathbb{E}_{q(x,z|\phi^{(t)})} \left[ \frac{p(x, z|\theta^{(0)})}{q(x, z|\phi^{(t)})} P(S^{(t)}|x) \log q(x|\phi) \right] \tag{16}$$

$$= \underset{\phi}{\operatorname{argmax}} \sum_{i=1}^{M} \frac{p(x_i^{(t)}, z_i^{(t)}|\theta^{(0)})}{q(x_i^{(t)}, z_i^{(t)}|\phi^{(t)})} P(S^{(t)}|x_i^{(t)}) \log q(x_i^{(t)}|\phi) \tag{17}$$

which is optimized in a similar fashion to (15); the only difference being that $x_i^{(t)}, z_i^{(t)} \sim q(x, z|\phi^{(t)})$.

### 2.4 Performance measures

To measure how well our overall objective traverses the search space, we first compute the fitness $w$ for a given variant $x$ as,

$$w(x) = \frac{\frac{\text{count}_{x,\text{selected}}}{\text{count}_{x,\text{input}}}}{\frac{\text{count}_{\text{WT, selected}}}{\text{count}_{\text{WT, input}}}} \tag{18}$$

where $\text{count}_{x,\text{selected}}$ and $\text{count}_{x,\text{input}}$ represent the count of variant $x$ in the selected library and input library, respectively. Similarly, $\text{count}_{\text{WT, selected}}$ and $\text{count}_{\text{WT, input}}$ represent the count of the WT in the selected and input libraries. Here, the fitness of each variant is normalized relative the WT sequence fitness score. Note that, since the regression-based oracle gives us a equivalent calculation via a mapping $f : \mathbb{R}^L \to \mathbb{R}$, we do not actually compute the fitness score using (18).

We want to use this score to visualize the fitness landscape for different inputs $x$. However, with a one-dimensional visualization, it is hard to understand which collection of variants are more "fit" than others. Naively, one could view the sequences in 2D space, where each sequence (inserted iteratively) represents a entry in 2D space. But this representaton is poor at preserving groups; instead, we want a projection that maintains the locality of the data points (i.e. sequences close to each other in 1D space should remain close to each other in the 2D representation). As it turns out, using a Hilbert curve [44], with order $\eta$ determined by the total number of datapoints $n$,[4]

$$\eta = \left\lceil \frac{\log \sqrt{n}}{\log 2} \right\rceil \tag{19}$$

allows for a locality-preserving mapping $\varphi : \mathbb{R}^1 \to \mathbb{R}^2$ [46]. Through this projection, we are able to spatio-temporally understand which collection of sequences are being queried by the search model $q(x|\phi^{(t)})$ for each iteration $t$.

## 3 Experiments

To demostrate the benefits of this approach, we focus on the problem of maximixation of a protein's fitness, although, as mentioned above, specification of a particular frequency can also be optimized. We systematically compare our approach to the widely-used random sampling technique found in directed evolution on a set of protein variants taken from the $\beta 1$ domain of the G protein [42].

In order to compare the methods, it was necessary to evaluate the data with a known ground truth since evaluating on a held-out test set was not feasible for our problem. To obtain a (somewhat) accurate ground truth model, we trained a GP regressor (GPR) on the same training data as the oracle using a kernel designed for protein-related tasks [47], whose features space was augmented by adding a bias feature and exponentiating to obtain a second-order kernel. We make this modification because a protein's fitness scores (for its variants) typically follow a nonlinear function [48]. The ground truth fitness score was taken to be the mean of this GPR model and logged for comparison.

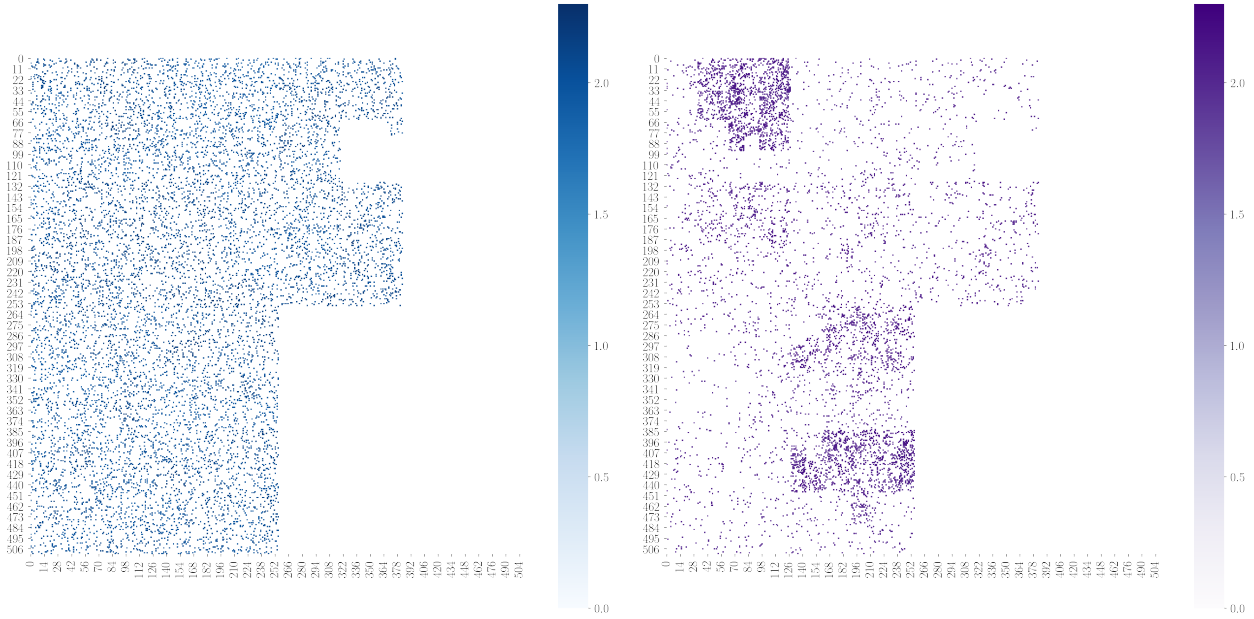
To optimize our method properly, we selected an oracle, GPR combination pair that was trained on the same subset of data. We stress, here, that we do not use different subsets of the the data for each model because we want to ensure that we do not cherry-pick the parameters of our predictive models. For the VAE model, however, we used the full dataset because it was trained in an unsupervised fashion only for the context of generating new sequences that are similar to original dataset. Table 1 shows the top 10 sequences (based off the oracle values), obtained by both the directed evolution (randomly-sampled) and the CbAS (adaptive sampling) method. While both methods reached approximately similar oracle and GT value scores, the latter reached the "optimal" variants more efficienty, requiring less oracle queries, as evidence by the more localized search in Fig. 1. This, as mentioned previously, will be especially useful in a real-world setting where querying resource-intensive assays to assess a variant's fitness might be unfeasible for most proteins.

Beyond the top $k$ sequences, when comparing the variants explored by the CbAS sample strategy versus the random strategy, we notice in Fig. 2 that the designed variants are likely to be "better" (in terms of fitness score) than the wildtype (WT) sequence. In fact, for variants with a relativey high amount of mutations, the adaptive sampling scheme seemingly outperforms the random search. This, along with the results above, suggests that the CbAS algorithm was (a) not led into untrusthworthy regions of the search space, and, more importantly, (b) more effectively able to design sequences for a specified design criteria.

[4] We kindly refer to §2.4 in [45] for a detailed derivation of how the 2D coordinates for the curve are generated.

| Method | | Top $k$ fitness scores (oracle) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CbAS | Sequence | QWAA | SWRA | QWGM | IWGV | QHGV | QWGV | SHCM | SWCV | QWMC | SWMC |
| | Oracle | 2.2608 | 2.2612 | 2.2613 | 2.2619 | 2.2634 | 2.2662 | 2.2716 | 2.2788 | 2.2820 | 2.3006 |
| | GPR (GT) | 2.5723 | 1.5646 | 1.3517 | 0.9700 | 1.8256 | 1.4020 | 1.1935 | 1.3307 | 1.8353 | 1.4270 |
| Random | Sequence | WWGW | SWGG | SWKC | WHGC | SWMV | IWGW | QWGW | IHGC | SWGA | SWMC |
| | Oracle | 2.2702 | 2.2730 | 2.2755 | 2.2759 | 2.2764 | 2.2780 | 2.2824 | 2.2832 | 2.3000 | 2.3006 |
| | GPR (GT) | 1.5839 | 1.2342 | 1.1606 | 2.0981 | 1.4024 | 0.9101 | 1.2201 | 2.8968 | 1.4159 | 1.4270 |

Table 1: Comparision between the top $k = 10$ sequences outputted by the CbAS and random optimization strategy, respectively. The values are sorted based on their oracle predictions since ground truth would be unknown.



(a) (b)

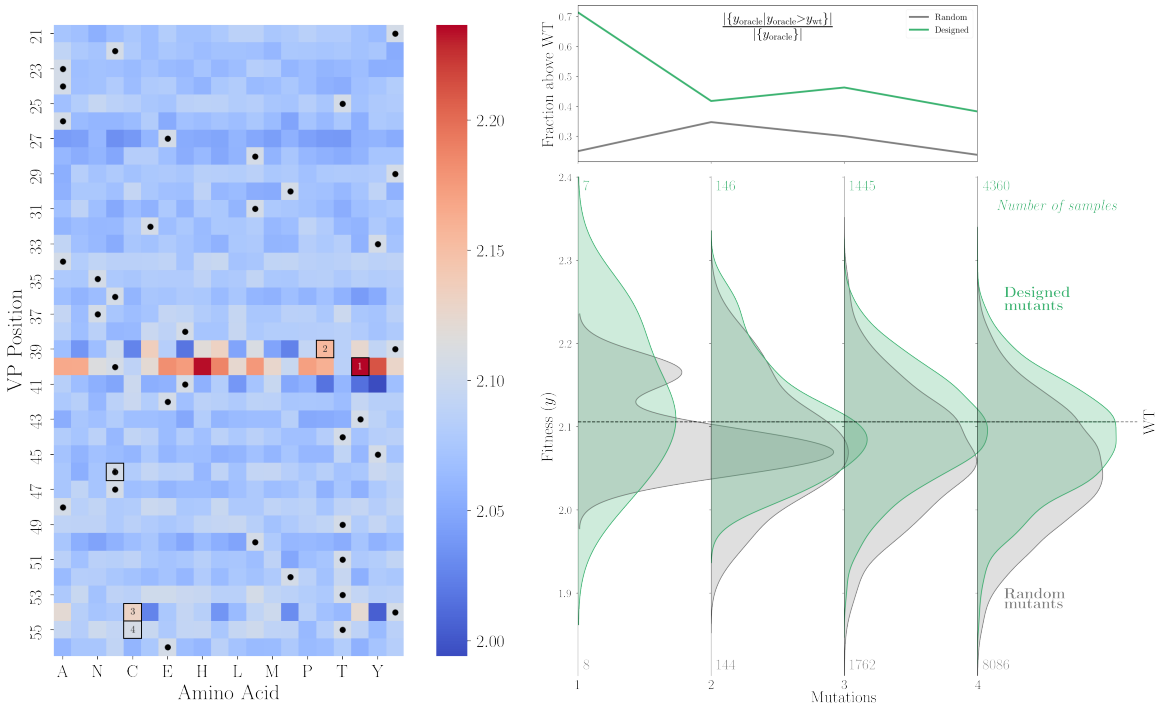
Figure 1: **Optimized search.** Protein sequences sampled uniformly **(a)** and using the CbAS optimization scheme **(b)** from the search space projected onto a order 9 Hilbert curve. Compared to the random search, the CbAS method quickly localizes its search space to regions where it thinks may contain feasible variants, leading to more efficient querying. This is especially beneficial for expensive-to-run querying tasks. Darker colors indicate higher (oracle) fitness scores.

## 4 Discussion

In this study, we used a gradient-free optimization strategy[5] to more efficiently explore a protein's fitness landscape. The statistical framework gives a grounding to approximate the conditional density even in the presense of rare events by leveraging the latent variable models structure to compute importance sampling for models whose densities cannot be computed exactly. Our experiments reveal that this approach works well under noisy "oracle" evaluations, querying regions of discrete search space with high GT scores, meaning that we can effectively design "better" proteins than other commonly used techniques, i.e. directed evolution. A big benefit with using a non-differentiable approach is that is allows us to optimize the design space with a black-box "oracle" – one that provides a simple input to output mapping, i.e. a wet-lab assay – meaning that this optimization scheme can be readily employed as part of a bigger protein design pipeline.

Additionally, due to how the generative model's parameters are updated at each step, this method can also be employed in a completely unsupervised setting, where no training set is provided to build the (initial) generative model. Instead, the prior would be a uniform distribution over the whole search space. We did not, however, test this setting since it

[5]We do use gradients to train both the oracle and generative models for performance reasons, but the optimization function itself does not require gradients. As such, it can easily be used with non-gradient based predicitve models.




(a) (b)

Figure 2: **Machine-guided design outperforms random mutagenesis. (a)** Multimutant sequence generation from individual amino-acid substitutions (black dots: WT). Fitness scores for each position are computed from the single amino-acid mutants. Sampled mutations are combined to generate a multimutant variant, which is evaluated on the oracle to determine the predicted fitness. The top 5 mutations (at different positions) are highlighted to show one possible variant. **(b)** Top: Fraction of designed mutants with fitness values greater than WT for random (grey) and machine-guided design mutants (green). Bottom: Distribution of fitness values for random and designed mutants separated by number of mutations. Even at the higher-end of mutation distances ($n = 4$), there is a seperation between the fraction of proteins above WT from designed and random mutants.

requires that the oracle predicts the property being measured well, which we cannot guarantee. As such, it might lead to less-than-optimal results when compared to other (semi-)supervised methods that use a training dataset.

While this approach does show some promising results navigating the search space, it does have its drawbacks. In particular, knowing if the algorithm has converged to the optimal sample (based off its design criteria) is hard to determine, since we do not know the underlying distribution. This is hightened by the fact that the search regions the algorithm chooses to explore are sensitive to (a) the number of iterations and (b) the number of points sampled per iteration, which directly affects the EvE tradeoff. Additionally, since the "oracle" poorly fits the training dataset (see §A.1), the search model might not perform optimally since it weights are partially optimized using the oracle's predictions[6]. Unsurprisingly, this does not affect the regions of the search space the algorithm explores, as it was designed to avoid pathological behavior of the oracle (see §A.2).

Although not directly related to the search strategy itself, another big downside is that we only consider variants with exactly $k$ mutations at the same amino acid positions. As the algorithm's main objective in (17) is to minimize the difference between the generative model's search and the conditional distribution, the samples the search model generates are not representative of the vast sequence space ($\Theta(20^L)$), but rather only a small subset of proteins ($\Theta(20^4)$). As such, there is a high likelihood that we miss finding optimal variants which satisfy our desired property (i.e. maximization of the fitness). In other words, it can be seen as finding the optimia locally rather than globally (since we do not have data sampled from the full search space to build our initial prior model).

We are working on further extensions of the approach, by using techniques found in the reinforcement learning literature [49, 50]. In particular, to tackle the EvE-tradeoff, we plan on using upper confidence bounds (UCB) [51] to prefer

[6]This does not mean that the algorithm is being lead astray into regions of space which are "poor", but rather that the reported fitness score on the newly sampled variants might not be accurate.

actions for which we don't have confident value estimates for, favoring exploration of samples with a strong potential to have great values. In a more nuanced flavor, since we have knowledge of the prior distribution, we can make better estimates of the bounds, allowing us to set the upper bound as the $95\%$ confidence interval [52]. Finally, using the Thompson sampling scheme could also allow us to select events $e$ based off its optimality probability, choosing samples that maximize the expected reward with respect to a randomly drawn belief [53].

While reinforcement learning techniques allow for better exploration strategies, simpler modifications, such as using domain-specific biochemistry knowledge, can make the oracle more accurate. For example, a model that incorporates the 3D structural features of the protein [54] could be used, owing to the fact that structure affects function, which, in turn, affects the overall protein's fitness. More specifically, consider the chemical differences between different amino acids. When substituting certain amino acids with others, where a non-polar R-group is substituted with another non-polar R-group, the overall structure of the protein might not change by much [55, 56]. On the other hand, with a polar R-group modification, the structure might change just enough to affect the overall fitness [57]. Overall, any model that is best able to learn an encoded representation of the protein (i.e. Unirep [13]) would be useful for a lot of downstream tasks.

## 5 Acknowledgements

We thank Brookes et. al. for useful clarifications and for providing python scripts[7] that served as a base to reproduce the CbAS method presented in [43]. It is important to stress that all we have done is adapted their method to work on the dataset complied in [42]. We also like to thank the authors of [58] for the inspiring the design of Fig. 2.

## References

[1] Bassil I Dahiyat and Stephen L Mayo. De novo protein design: fully automated sequence selection. *Science*, 278(5335):82–87, 1997.

[2] Christian Jäckel, Peter Kast, and Donald Hilvert. Protein design by directed evolution. *Annu. Rev. Biophys.*, 37:153–173, 2008.

[3] Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33, 2016.

[4] Michael Dickson and Jean Paul Gagnon. The cost of new drug discovery and development. *Discovery medicine*, 4(22):172–179, 2009.

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

[6] Niles A Pierce and Erik Winfree. Protein design is np-hard. *Protein engineering*, 15(10):779–782, 2002.

[7] Keqin Chen and Frances H Arnold. Enzyme engineering for nonaqueous solvents: random mutagenesis to enhance activity of subtilisin e in polar organic media. *Bio/Technology*, 9(11):1073–1077, 1991.

[8] Philip A Romero and Frances H Arnold. Exploring protein fitness landscapes by directed evolution. *Nature reviews Molecular cell biology*, 10(12):866–876, 2009.

[9] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[10] Gisbert Schneider, Wieland Schrödl, Gerd Wallukat, Johannes Müller, Eberhard Nissen, Wolfgang Rönspeck, Paul Wrede, and Rudolf Kunze. Peptide design by artificial neural networks and computer-based evolutionary search. *Proceedings of the National Academy of Sciences*, 95(21):12179–12184, 1998.

[11] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-n protein engineering with data-efficient deep learning. *bioRxiv*, 2020.

[12] Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine learning in protein engineering. *arXiv preprint arXiv:1811.10775*, 2018.

[13] Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, 2019.

[7] https://github.com/dhbrookes/CbAS

[14] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[15] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[16] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

[17] Melika Behjati, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard. Universal adversarial attacks on text classifiers. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7345–7349. IEEE, 2019.

[18] Alexander Amini, Ava P Soleimany, Wilko Schwarting, Sangeeta N Bhatia, and Daniela Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019.

[19] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

[20] Subroto Gunawan and Panos Y Papalambros. A bayesian approach to reliability-based optimization with incomplete information. 2006.

[21] Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

[22] Victoria O Pokusaeva, Dinara R Usmanova, Ekaterina V Putintseva, Lorena Espinar, Karen S Sarkisyan, Alexander S Mishin, Natalya S Bogatyreva, Dmitry N Ivankov, Guillaume J Filion, Lucas B Carey, et al. Experimental assay of a fitness landscape on a macroevolutionary scale. *bioRxiv*, page 222778, 2017.

[23] Anthony D Keefe and Jack W Szostak. Functional proteins from a random-sequence library. *Nature*, 410(6829):715–718, 2001.

[24] Louis Du Plessis, Gabriel E Leventhal, and Sebastian Bonhoeffer. How good are statistical models at approximating complex fitness landscapes? *Molecular biology and evolution*, 33(9):2454–2468, 2016.

[25] Shumeet Baluja and Scott Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1997.

[26] Johannes Schneider, Markus Dankesreiter, Werner Fettes, Ingo Morgenstern, Martin Schmid, and Johannes Maria Singer. Search-space smoothing for combinatorial optimization problems. *Physica A: Statistical Mechanics and its Applications*, 243(1-2):77–112, 1997.

[27] Jie Chen, Bin Xin, Zhihong Peng, Lihua Dou, and Juan Zhang. Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(3):680–691, 2009.

[28] Ashish Sabharwal, Horst Samulowitz, and Chandra Reddy. Guiding combinatorial optimization with uct. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 356–361. Springer, 2012.

[29] Brian Sheppard. World-championship-caliber scrabble. *Artificial Intelligence*, 134(1-2):241–275, 2002.

[30] Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michele Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113, 2012.

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[32] Sylvain Gelly and Yizao Wang. Exploration exploitation in go: Uct for monte-carlo go. 2006.

[33] David H Brookes and Jennifer Listgarten. Design by adaptive sampling. *arXiv preprint arXiv:1810.03714*, 2018.

[34] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[35] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation*, 9(3):303–317, 2005.

[36] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

[37] Catherine A Macken and Alan S Perelson. Protein evolution on rugged landscapes. *Proceedings of the National Academy of Sciences*, 86(16):6191–6195, 1989.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[39] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[40] Bruno Sareni, L Krahenbuhl, and Alain Nicolas. Efficient genetic algorithms for solving hard constrained optimization problems. *IEEE Transactions on Magnetics*, 36(4):1027–1030, 2000.

[41] James E Noble and Marc JA Bailey. Quantitation of protein. In *Methods in enzymology*, volume 463, pages 73–95. Elsevier, 2009.

[42] Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun. Adaptation in protein fitness landscapes is facilitated by indirect paths. *Elife*, 5:e16965, 2016.

[43] David H Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. *arXiv preprint arXiv:1901.10060*, 2019.

[44] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*, pages 1–2. Springer, 1935.

[45] Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

[46] Bongki Moon, Hosagrahar V Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on knowledge and data engineering*, 13(1):124–141, 2001.

[47] Wen-Jun Shen, Hau-San Wong, Quan-Wu Xiao, Xin Guo, and Stephen Smale. Introduction to the peptide binding problem of computational immunology: New results. *Foundations of Computational Mathematics*, 14(5):951–984, 2014.

[48] Changyu Hu, Xiang Li, and Jie Liang. Developing optimal non-linear scoring function for protein design. *Bioinformatics*, 20(17):3080–3098, 2004.

[49] Dimitris E Koulouriotis and A Xanthopoulos. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, 196(2):913–922, 2008.

[50] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[51] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

[52] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On bayesian upper confidence bounds for bandit problems. In *Artificial intelligence and statistics*, pages 592–600, 2012.

[53] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1, 2012.

[54] Hyeoncheol Cho and Insung S Choi. Three-dimensionally embedded graph convolutional network (3dgcn) for molecule interpretation. *arXiv preprint arXiv:1811.09794*, 2018.

[55] Matthew J Betts and Robert B Russell. Amino acid properties and consequences of substitutions. *Bioinformatics for geneticists*, 317:289, 2003.

[56] Vakhtang V Loladze, Dmitri N Ermolenko, and George I Makhatadze. Thermodynamic consequences of burial of polar and non-polar amino acid residues in the protein interior. *Journal of molecular biology*, 320(2):343–357, 2002.

[57] Rohan DA Alvares, David V Tulumello, Peter M Macdonald, Charles M Deber, and R Scott Prosser. Effects of a polar amino acid substitution on helix formation and aggregate size along the detergent-induced peptide folding pathway. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1828(2):373–381, 2013.

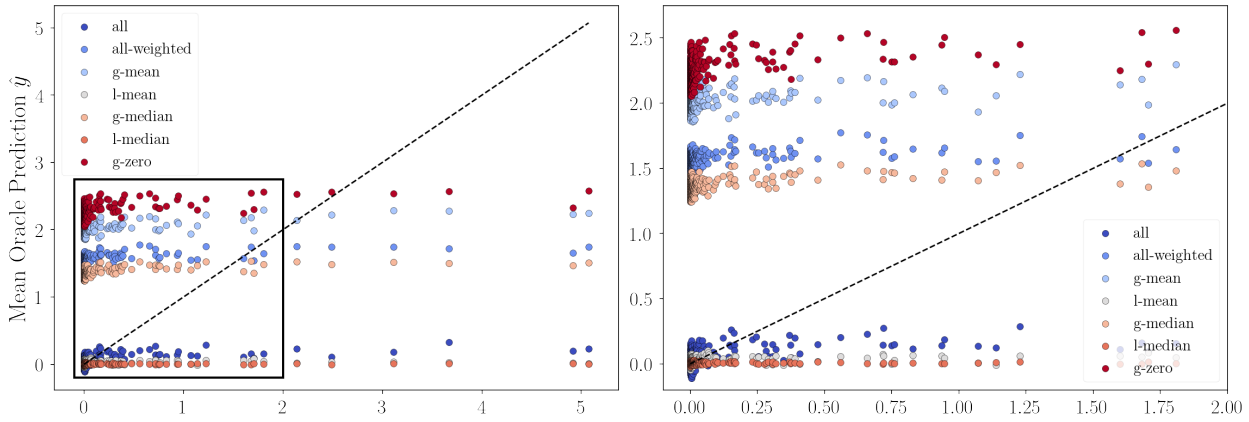
[58] Pierce J Ogden, Eric D Kelsic, Sam Sinai, and George M Church. Comprehensive aav capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science*, 366(6469):1139–1143, 2019.

# A Appendix

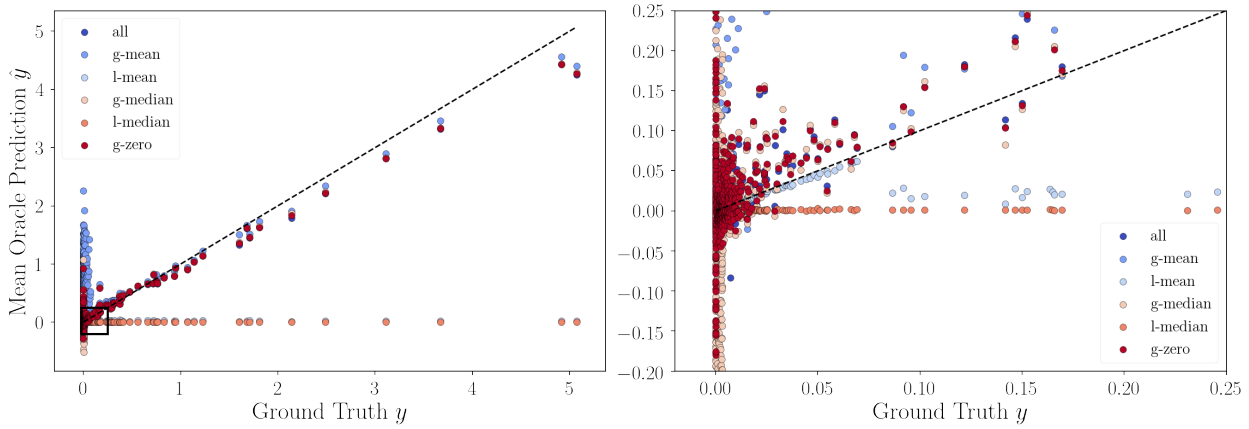
## A.1 Model accuracy

The oracle is a densely-connected neural network trained on one-hot encoded representations of the protein sequence. It is evident from Fig. 3 that the oracle was poorly trained, despite the fact that the model's validation loss was relatively low for each run. In fact, even after correcting for imbalanced samples (since most samples have a low fitness score), the oracle's predictions were still subpar when compared to the true fitness value, increasing by approximately the same amount for every sample. As such, there is very little variance in the fitness score prediction between samples.

This is remarkably different from the GPR, which shows great correspondence with the training dataset. In fact, for most subsets of the dataset with which it was trained on, the GPR is well-behaved (atleast within the domain shown). This suggests that, the oracle did not capture what makes a sequence "good" vs "bad" in terms of its fitness, while the GPR was better able to make the distinction. Overall, this suggests, for optimality, we should choose the oracle's parameters trained on values greater than the mean, and the GPR's parameters on values greater than zero.




(a)




(b)

Figure 3: **Model selection.** Predicted vs. ground truth fitness score for samples trained on different subsets of the training data. In (a), irrespective of the GT fitness score, most predictions are the same, while the GPR model in (b) shows better alignment between the predicted and ground truth values. The dashed line denotes what an ideal model will predict (aka $\hat{y} = y$ for both the predicted and true values). The figures on the right show the zoomed in view of the bounding boxes shown on the left.

## A.2 Search model optimization

To measure how well the search model is optimized, we measure both the oracle and GP predictions (a proxy for the GT) for each sequence sampled at every iteration $t$. Fig. 4 shows the $80^{\text{th}}$ percentile for both the oracle and GT values

across different runs of the algorithm, which use different models trained on various subsets of the training data. As intended per the formulation, the results indicate that the search strategy does not get led into pathological regions of the search space since the GT values remain consistently high even after a certain score is reached. If, instead, the GT values were to decrease substantially, this would indicate that the algorithm is being negatively influenced by the oracle's inconsistent behavior.

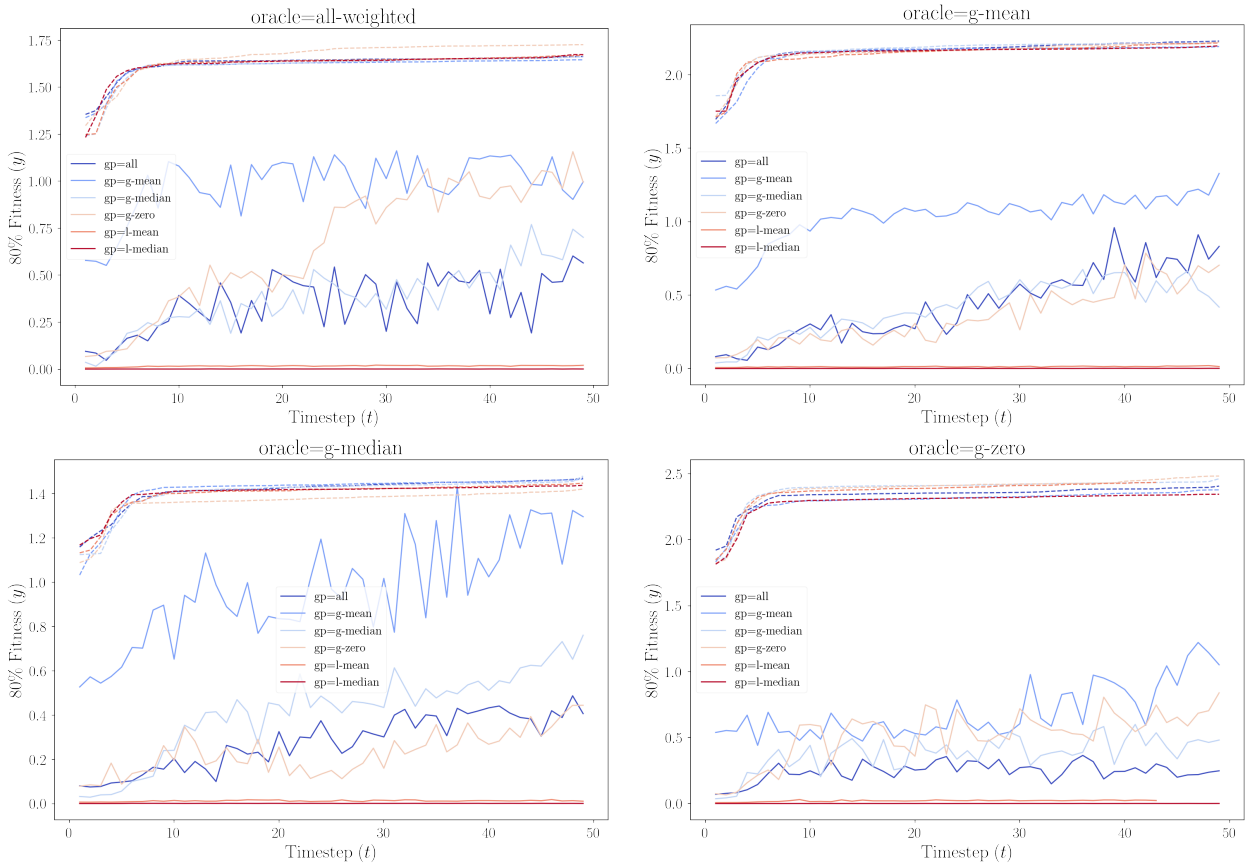



Figure 4: **Maximization of protein fitness.** Trajectory plots of runs for each oracle. Each line (within a panel) represents the algorithm's trajectory for a specific run of the GP trained on a subset of the original data. Each point in the dashed line represents the $80^{\text{th}}$ percentile of oracle evaluations at that iteration. The cooresponding point on the solid line indicates the mean ground truth value as computed by the GPR. The trajectory is sorted by increasing oracle values such that the last point on the curve represents the final designed variant that would be used (in this case the highest oracle value predicted). This is done because the ground truth values would be unknown in a real-world setting.

### A.3 Variants that could improve the model

Taking into account the algorithm's limitations, the most immediate improvement can likely be made by using a more representative training dataset (i.e. one that contains variants that have a high fitness score) to help balance the dataset. Not only would this improve the prior generative model's parameters (generating better initial samples), but also the accuracy of the oracle's predictions, further improving the search model in subsequent iteration cycles. Ideally, this will help find optimal sample(s) based off the design criteria more effectively. Alternatively, we could choose to include variants based on how uncertain we are about a given sample (this is the UCB strategy used in multi-armed bandit problems). Although not immediately useful to build a more accurate oracle, it helps sample points quickly to reduce the uncertainty of the unknowns before "locking" onto variants satisfying the property of interest.