Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Science

Diploma thesis

Protein engineering with large language models

Author: Bc. Matouš Soldát
Supervisor: doc. Ing. Jiří Kléma, Ph.D.
Study programme: Medical Electronics and Bioinformatics
Specialization: Bioinformatics

Prague, May 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Soldát Matouš** Personal ID number: **491946**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Medical Electronics and Bioinformatics**

Specialisation: **Bioinformatics**

II. Master’s thesis details

Master’s thesis title in English:

Protein engineering with large language models

Master’s thesis title in Czech:

Využití velkých jazykových modelů v inženýringu proteinů

Guidelines:

Protein engineering is a process of designing proteins with desired properties related to its function such as the protein’s stability, catalytic function or binding to a specific molecule. Because the properties of proteins are determined by their structure and structure is encoded in the sequence of amino acids, the task translates to finding a sequence of amino acids with the specified function. The aim of machine learning (ML) assisted protein engineering is to improve the fitness of the final protein without increasing the amount of carried-out screening. ML accomplishes this goal by incorporating information about the already screened variants into a model which predicts a protein’s fitness based on its sequence. Pre-trained protein language models can be used to generate suitable protein variants for screening by masking positions in the protein sequence and letting the models predict amino acids with high pseudolikelihood at those positions. The goal is to increase the quality of screened variants and reduce the total amount of screening needed to reach a variant of satisfactory fitness.
1. Familiarize yourself with the theory of large language models and existing pre-trained protein models.
2. Study existing applications of large language models in in-silico active learning of proteins.
3. Perform a literature review on topics 1 and 2.
4. Suggest an effective way to use these models over the benchmark data available in the literature (protein databases with the known screening outcomes).
5. Evaluate procedure ad 4, the basic measure of quality is the number of in-vitro protein evaluations required to achieve the target protein design (i.e., the number of accesses to the database with screening outcomes).
6. Compare with basic reference benchmarks (naïve model, ablation studies).

Bibliography / sources:

Zhang, Shuang, et al. "Applications of transformer-based language models in bioinformatics: a survey." Bioinformatics Advances 3.1 (2023): vbad001.
Madani, Ali, et al. "Progen: Language modeling for protein generation." arXiv preprint arXiv:2004.03497 (2020).
Settles, Burr. "Active learning literature survey." (2009).

Name and workplace of master's thesis supervisor:

doc. Ing. Jiří Kléma, Ph.D. Intelligent Data Analysis FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **12.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Ing. Jiří Kléma, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

Author's signature

Acknowledgements

First and foremost, I am deeply grateful to my supervisor, doc. Ing. Jiří Kléma, Ph.D., for his guidance through all stages of writing my diploma thesis from the definition of the objectives and fruitless initial experimentation to presentation and interpretation of the final results. My thanks are also due to my twin brother, Bc. Šimon Soldát, for enthusiastic discussions of the proposed methods and an insight into Bayesian optimization.

I wish to extend special thanks to my partner, Maria Šťastná, and my family for their continued support of my studies, both financial and otherwise, and to my partner especially for her patience with my absent-minded babbling about abstract academic concepts.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged.

Abstract

The objective of protein engineering is to design proteins with desired properties. Directed evolution is an iterative laboratory process of designing such proteins by iteratively synthesizing new protein variants and evaluating their desired property with expensive and time-consuming biochemical screening. Machine learning methods can help select informative or promising variants for screening to increase the quality of screened variants and reduce the amount of necessary screening. The goal of this thesis is to suggest an effective way to exploit pre-trained protein language models in directed evolution. The thesis provides a review of existing pre-trained protein language models and their application in protein engineering, as well as an introduction to the application of Bayesian optimization for directed evolution. Afterward, three machine-learning-assisted methods for directed evolution are proposed and compared to classical methods of directed evolution and state-of-the-art machine-learning-assisted methods. The proposed methods exploit protein sequence representation extracted from a pre-trained protein language model. The most promising of the proposed methods, Bayesian optimization in embedding space (BOES), combines the high-dimensional representation with Bayesian optimization by limiting the effective number of dimensions to one with a custom kernel. BOES outperforms state-of-the-art model-regression methods by 17 % with the same screening effort and can save 44 % of the experimental burden in comparison to BO-based methods with a different informative protein sequence representation.

Keywords: protein engineering, directed evolution, large language models, sequence embedding, Bayesian optimization

Abstrakt

Cílem inženýringu proteinů je návrh proteinů s požadovanými vlastnostmi. Řízená evoluce je iterativní laboratorní proces návrhu takových proteinů pomocí iterativní syntézy nových variant proteinů a vyhodnocování míry požadované vlastnosti drahými a časově náročnými biochemickými experimenty. Metody strojového učení mohou pomoci s výběrem informativních a slibných variant k experimentálnímu ověření a tím zvýšit kvalitu objevených variant a snížit množství provedených experimentů. Cílem této práce je navrhnout efektivní způsob využití předučených proteinových modelů pro řízenou evoluci proteinů. Práce poskytuje rešerši existujících předučených proteinových modelů a jejich aplikací v proteinovém inženýrství. Dále poskytuje úvod do využití Bayesovské optimalizace pro řízenou evoluci proteinů. Následně jsou navrženy tři metody pro řízenou evoluci proteinů asistované strojovým učením, které jsou porovnány s klasickými metodami řízené evoluce i ostatními moderními metodami s asistencí strojového učení. Navržené metody využívají representaci proteinových sekvencí pomocí předučeného proteinového modelu. Nejslibnější z navržených metod, Bayesovká optimalizace v prostoru embeddingů (zkratkou BOES), kombinuje tuto vysoko-dimenzionální representaci s Bayesovskou optimalizací pomocí kernelu, který snižuje efektivní počet dimenzí na jednu. BOES překonává moderní metody regrese modelu o 17 % při zachování stejného množství provedených experimentů a může ušetřit 44 % nutných experimentů ve srovnání s metodami Bayesovké optimalizace s jinou informativní representací proteinových sekvencí.

Klíčová slova: inženýring proteinů, řízená evoluce, velké jazykové modely, embedding sekvencí, Bayesovská optimalizace

Contents

1 Introduction

Protein Engineering is a process of designing proteins with desired properties related to their function such as the protein's stability, catalytic function, or binding to a specific molecule [1]. This can be leveraged in industrial applications, environmental applications, medicine, nanobiotechnology, and other fields [1]. However, due to the combinatorial nature of protein sequences, the number of possible variants is often too large for an exhaustive search [1, 2]. To solve this problem, biologists use directed evolution (DE), an iterative laboratory process of creating new biomolecules of desired properties, that mimics Darwinian evolution in a controlled environment [3]. During directed evolution, an existing protein is iteratively mutated producing multiple new protein variants, which are screened to identify mutations beneficial to the protein's desired function. The beneficial mutations are kept and the protein is mutated again in the following iteration.

The wet lab experiments required to synthesize and screen the mutated protein variants during a DE procedure are expensive and time-consuming [4]. Because of this, the screening process is a common bottleneck of all DE methods. This motivates the employment of machine learning methods to minimize the amount of conducted screening while maximizing the highest obtained fitness. The utilized machine learning methods usually incorporate information about the already screened variants into a model which predicts a protein's fitness based on its sequence [4, 5]. The model is then used to intelligently select new variants for screening which maximize the predicted fitness and/or minimize uncertainty in the model [4].

A field of machine learning that experienced massive growth in recent years, both in the context of protein engineering and in general, are large, pre-trained language models. These deep neural networks with up to billions of parameters can be trained on enormous amounts of proteins to generate protein sequences. In the process, the models learn to represent protein sequences in a highly informative space [5], which captures properties of the protein important to its structure and function [6, 7, 8]. These models cannot be trained directly during a DE procedure. The amount of protein variants, which can be experimentally screened, is far too small relative to the huge amount of parameters in the model. However, the models' ability to capture important properties of proteins can be exploited to provide informative protein sequence embedding to other machine learning methods [9].

In this thesis, three methods of machine-learning-assisted DE are proposed. All of the methods employ a pre-trained protein language model as a sequence embedding extractor and operate in the more informative sequence embedding space instead of exploring the raw protein sequence space directly. The use of the embedding space also allows for the definition of a sensible metric of similarity between the protein variants, which is a key component of all active learning methods. The first proposed method selects variants for screening to maximize their informativeness and train a two-layer perceptron as a protein

fitness predictor. After the model is trained, it can be utilized to select additional variants with high predicted fitness for screening. The second method, termed NSDE, uses the acquired metric of similarity to construct a sequence similarity network [10] and conducts a greedy graph search from the original protein to optimize its fitness. The third method employs Bayesian optimization with a Gaussian process model [11] to not only predict the protein variants' fitness but also model the variance of the prediction. By maximizing the expected improvement of the current maximum in the modeled fitness landscape, the method strikes a balance between exploitation of known areas of high fitness and exploration of unknown areas with high uncertainty in the prediction.

The methods are tested on two datasets of protein variants mutated at four positions and their experimentally measured fitness. The main metric for evaluation of the proposed methods is the highest fitness identified in screening during the DE procedure relative to the total amount of conducted screening. This metric is grounded in the practical application of methods for DE. The methods are compared to each other as well as two implemented benchmarks. The benchmarks simulate traditional DE procedures which make no use of machine learning methods. Apart from running the proposed algorithms from the original wild-type protein, evaluation of robustness to the position of the starting protein in the sequence space is conducted. The robustness is evaluated by repeating the experiments with randomly sampled variants serving as the starting protein.

2 Background

Given the interdisciplinary nature of this thesis, a concise introduction to the related topics is in place. In this section, background to directed evolution as a means of protein engineering is provided, followed by background of large language models with a review of existing pre-trained protein language models applicable to the task of directed evolution. Lastly, this section touches upon foundations of active learning and optimization with expensive data-point evaluation because, as will be made clear, the goal of machine-learning-assisted directed evolution corresponds closely to the general objective of methods in these fields.

2.1 Protein Engineering

Protein Engineering (PE) is a process of designing proteins with desired properties related to their function such as the protein's stability, catalytic function, or binding to a specific molecule [1]. Because the properties of proteins are determined by their structure and structure is encoded in the sequence of amino acids [12], the task of PE translates to finding a sequence of amino acids with the specified properties/function. However, the number of possible protein sequences is higher than the number of atoms in the universe [2] and non-functional proteins dominate the space [13]. Because of this, searching the space of all possible proteins for the optimal sequence is an NP-hard problem [14]. One

of the most widespread approaches to solving this issue with a vast range of applications is a process called Directed Evolution [3, 15].

2.2 Directed Evolution

Directed Evolution (DE) is an iterative laboratory process of creating new biomolecules of desired properties, which mimics Darwinian evolution in a controlled environment [3]. DE circumvents the problem of the vast protein-sequence space filled with non-functional sequences [2], which cannot be searched comprehensively, by iteratively mutating an existing protein (often called the wild-type variant) to improve its function [16]. This process is inspired by the natural evolution of proteins [13]. It can be viewed as a mutation walk from the original sequence to the final variant with improved properties in the mutational space of all possible variants [17]. A DE iteration consists of two main steps: mutagenesis, in which parent molecule(s) are mutated and/or recombined to create a vast library of variants, and screening/selection, where variants are selected from the obtained library to form a new generation of parents with improved properties [3, 15]. Many methods were designed for the mutagenesis step as well as the screening/selection of variants. The most commonly used methods are described in more detail in [3]. In sections 2.2.1 and 2.2.2, a brief summary of the main approaches discussed in [3] follows. Each approach encompasses a variety of related methods.

2.2.1 Mutagenesis

Considering the 20 amino acids commonly found in living organisms, the mutational space of a protein of length n consists of 20^n unique variants. Given that protein lengths are usually in the order of hundreds of amino acids [18], a library of all possible variants can never be prepared in practice [16]. Instead, the parent sequence(s) are mutated to obtain new variants relatively similar to the parent sequence in the context of the whole mutational space [16]. Mutations are introduced either randomly in the so-called *random mutagenesis* or in a more targeted manner denoted as *focused mutagenesis*. More different variants can be obtained through multiple iterations of the two steps of DE: mutagenesis of parent sequence(s) and selection of new parents from the obtained library. Random single-gene mutagenesis can perform a sparse sampling of the neighboring mutational space [3]. Such sampling can help identify areas highly correlated to the desired protein property when little information about the mutational space is known [3]. However, if some prior knowledge about the mutational space exists, it can be exploited to sample more informative variants or variants more likely to possess the desired properties. For example, if certain positions in the sequence are known to be largely influential to the protein function, targeted mutagenesis can maximize sampling at these positions, and variants with desired properties can be identified from a much smaller library [3]. Similarly to the exploitation of prior knowledge, computational modeling and machine learning can be employed to leverage the information from the already sampled variants to predict beneficial mutations, further reducing the size of the variant library necessary to identify the desired variant [3, 4]. The use of machine learning for DE is discussed in more depth

in section 2.2.3.

2.2.2 Selection / Screening

The two main approaches to obtaining a new set of desirable parents from the library of variants are selection and screening [3]. In screening, each variant from the library is tested individually by the means of biochemical or biophysical analytical methods to determine its fitness. A protein's fitness is typically defined as a numerical value describing the desired property of the protein. The best-performing variants in regards to the chosen fitness function are then selected to be the parents in the next mutagenesis step. Selection-based methods, on the other hand, impose some selective pressure on variants from the library to eliminate undesired variants. Only the surviving, more fit, variants remain to parent the next library in the mutagenesis step. The selection-based methods allow for a high-throughput approach which enables the assessment of larger libraries of variants than in screening-based methods. The downside of selection-based methods is that they are only viable if the evaluated protein function can be linked to the growth or survival of the host organism [3].

In this thesis, all biochemical experiments will be simulated by reading a fitness value from an existing dataset of screened protein variants. This simulation corresponds to a screening approach, where experimental screening of one variant is replaced by reading the corresponding fitness value from the dataset. However, it is important to remember that while reading a single fitness value from the dataset is an almost instantaneous task, the experimental screening it represents is very demanding in both time and financial cost and is the bottleneck of any machine-learning-assisted DE procedure.

2.2.3 Machine-Learning-Assisted Directed Evolution

Because of the aforementioned cost associated with the screening step of DE, the objective of machine-learning-assisted directed evolution (MLDE) is to improve the fitness of the final protein without increasing the amount of carried-out screening. An alternative formulation of the objective is to reduce the number of screened variants necessary to reach the desired fitness level. MLDE accomplishes this goal by exploiting the information about all screened variants instead of discarding the low-fitness variants, which is done in traditional DE procedures. The difference between the traditional DE approach and MLDE approach is illustrated in figure 1. MLDE methods use the fitness of the screened variants to fit a model which can be used to predict the fitness of other variants [4]. This model is then used to intelligently select new variants for screening to maximize the predicted fitness and/or to reduce uncertainty in the model, which in turn improves prediction quality in future iterations. A wide variety of models have been applied to MLDE including simple linear regression models, decision trees/forests, kernel methods, Gaussian Process models, and deep learning [4]. Yang et al. [4] give a simple, general heuristic and reasoning for choosing a model based on the data and specific task. The important point is that no model is universally optimal for any given task. In this thesis, we will focus on

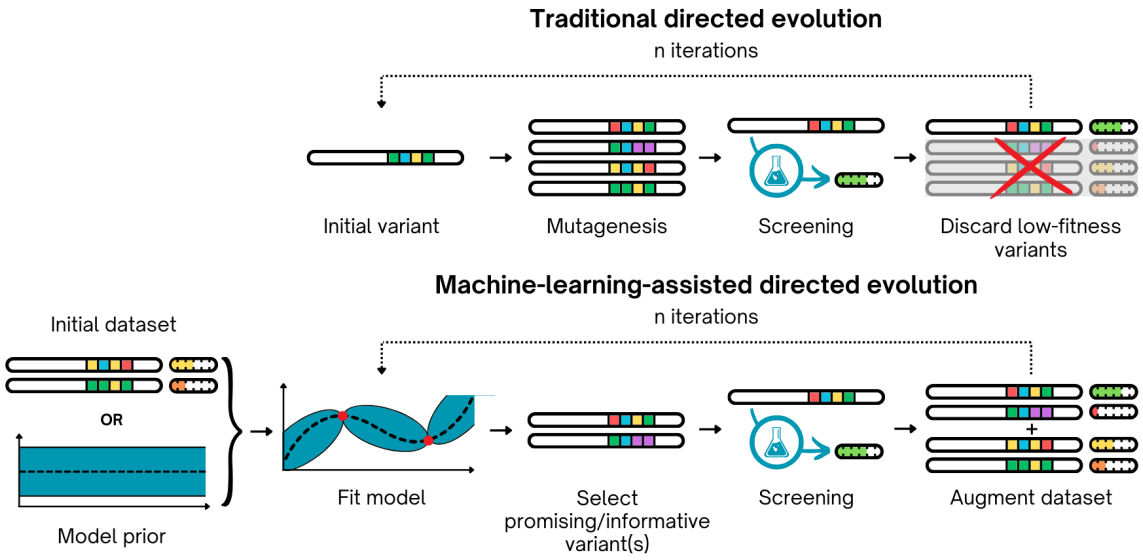


Figure 1: Comparison of traditional directed evolution and machine-learning-assisted directed evolution approaches: Traditional DE discards the low-fitness variants whereas MLDE incorporates all available information into a predictive model, which is used to select promising/informative variants in following iterations.

the application of large language models, a class of very large deep learning models which stem from natural language processing, in MLDE.

2.3 Large Language Models

Natural language processing (NLP) is a field combining artificial intelligence and linguistics which started in the 1950s [19] with the formulation of the Turing test [20]. Originally, NLP research focused on the modeling and interpretation of the human language by computers [19]. However, the conversion of the idea to the biological domain comes naturally, since most of the data analyzed in bioinformatics (i.e. DNA, RNA, and protein sequences) can also be interpreted as a language and has been interpreted as such since the structure of DNA was solved [21]. In recent years, NLP has been increasingly present in bioinformatics [22] and with the recent boom of large language models (LLMs) spearheaded by ChatGPT, LLMs have also made their way into bioinformatics. In this section, the background of LLMs is introduced, followed by a survey of existing LLMs pre-trained on protein sequences.

Language modeling (LM), a broader term encompassing LLMs, aims to construct a model of probabilities to predict the next (or missing) token [23], e.g. the next word in a sentence or an amino acid in a protein sequence. [23] provides a concise description of the development in LM of which LLMs represent the most recent stage. The first development stage, *statistical language models*, are models that predict the next token based on the

most recent context following the Markov assumption [24]. The following stage are *neural language models*, which employ neural networks to characterize the probability of word sequences and introduce the idea of learning effective features from the data, such as the well-known word2vec model [25]. Next stage, language models pre-trained on large datasets, aim to capture the fundamental rules of the language as well as learn to extract effective features. These general models can then be fine-tuned to perform well on a specific downstream task providing a solid foundation for different applications. Lastly, the LLMs are scaled in size, resulting in not only improved performance but also surprising, so-called *emergent abilities*. These abilities are not present in a smaller model but can be observed in larger models (of the same architecture) [26]. An important takeaway from the development in LM is that the focus has gradually shifted from simple *language modeling* to complex *task solving* via text generation. Currently, LLMs typically refer to language models that use the transformer architecture [27] and contain up to hundreds of billions of parameters.

2.3.1 Pre-trained Protein Language Models

This section summarizes existing pre-trained protein language models (PPLMs). These are LLMs pre-trained on large amounts of protein sequence data to predict the next or missing token (amino acid). Even though the training objective is simple token prediction the models need to learn complex representations of the input protein sequence to achieve high accuracy. The learnt representations capture important properties of the protein such as secondary structure, binding site positions, and the protein contact map [6, 7, 8]. After pre-training, PPLMs can be fine-tuned on a specific task or used to extract informative embeddings from protein sequences, which contain potentially useful information for further analysis with other machine learning methods. While PPLMs can be designed for different applications, this survey focuses mainly on models exploitable as protein sequence embedding extractors or models exploitable in the selection step of MLDE to generate protein sequences with statistically higher fitness than random selection.

Evolutionary Scale Modeling (ESM) is a repository of code and weights for PPLMs from the META Fundamental AI Research Protein Team (FAIR). It provides five main transformer protein language models, each designed for a different application, as well as older and/or alternate versions of some models. The models most interesting for usage in MLDE selection step are ESM-2 [28], a state-of-the-art general-purpose PPLM that can be used to predict structure, function and other protein properties directly from individual sequences, and ESM-1v [29], a language model specialized for prediction of variant effects, that enables SOTA zero-shot prediction of the functional effects of sequence variations. Checkpoints of different scales are available for the ESM-2 model with 8M, 35M, 150M, 650M, 3B, and 15B parameters. The ESM-1v model is only available in a 650M-parameter-scaled version. It is also worth mentioning the ESM-1b model [6], which is an older model with the same architecture as ESM-1v and a common choice in MLDE applications [30, 31]. The two models differ in training data. The ESM-1v model was trained on the UniRef90 dataset [32], while the ESM-1b model was trained on UR50/S [6], a high-diversity sparse

dataset which uses the UniRef50 [32] representative sequences. Other notable models from the ESM repository include: ESMFold [28], a SOTA end-to-end single sequence 3D structure predictor. ESM-IF1 [33], an inverse folding model that can be used to design sequences for given structures, or to predict functional effects of sequence variation for given structures and enables SOTA fixed backbone sequence design. And ESM-MSA-1b [34], a model that can be used to extract embeddings from an MSA and enables SOTA inference of structure.

The ESM-2 general-purpose model [28] is the largest developed protein language model to date, lacking only one or two orders of magnitude behind the current largest SOTA language models of text like PALM (540B parameters) [35] or GPT-4 [36], which has hundreds of billions to over a trillion parameters. As such, ESM-2 deserves a little more attention. The model is based on a BERT [37] style encoder-only transformer architecture and it is trained on protein sequences from the UniRef database [32] with masked-token prediction training objective and 15 % masking. The differently scaled checkpoints reveal, among an expected increase in performance, the emergence of protein structure with increasing model size. This observation validates a previously observed phenomenon that the pairwise interaction patterns learned by the attention mechanisms of transformer-based protein language models correspond to protein contact maps [7, 8].

ProtTrans [38] is a suite of six models, each with a different architecture, developed by Rostlab. It contains two auto-regressive model architectures (Transformer-XL [39], XLNet [40]), which are decoder-only, and four auto-encoder architectures (BERT [37], Albert [41], Electra [42], T5 [43]), where T5 uses the original transformer architecture consisting of an encoder and a decoder, while the other three architectures are encoder-only. Furthermore, some of the models are available in multiple variants which are trained on different datasets or have different scales. Three datasets were used to train the models - Uniref50, UniRef100 [32], and BFD (Big Fantastic Database) [44]. For the specific datasets used for each of the models, see the original paper [38]. t-SNE projections of the obtained protein embeddings suggested that all of the models captured essential protein properties such as biophysical amino acid features, protein length, life domains (archaea, bacteria, eukarya, along with viruses) and to some extent protein function and secondary structure.

RITA [45] is a suite of autoregressive models, developed by a collaboration of Lighton, the OATML group at Oxford, and the Debbie Marks Lab at Harvard. It features four models with a common architecture and different scales, having 85M, 300M, 680M, and 1.2B parameters, respectively. The models are trained on the UniRef-100 database [32] as decoder-only transformer models with no conditioning information. The models' hyper-parameters correspond to GPT-3 [46]. The authors compared two positional embedding techniques, Rotary Positional Embeddings (RoPE) [47] and AliBi [48] in an ablation study and concluded that RoPE resulted in lower language modeling loss. Three small models were also pre-trained on different datasets and compared in terms of the transferability of information learned on each dataset to the others. The UniRef-100 database was con-

cluded to yield the best results but the authors note that using a combination of several datasets may be beneficial.

ProGen [49] is a conditional PPLM for controllable protein generation with 1.2B parameters. It is trained on 280 million protein sequences with conditioning tags which encode the protein's annotation such as taxonomic, functional, and locational information. These conditioning tags can be used to generate new protein sequences with selected desirable properties. The model is shown to sample variants from the GB1 library [50] with statistically higher fitness than a random sampling procedure in a zero-shot setting. The GB1 library is a challenging dataset consisting of variants of the protein G domain B1, mutated at four positions with non-linear epistasis. The fitness landscape contains multiple fitness peaks and is heavily populated by zero-fitness and low-fitness variants. However, the median fitness sampled by ProGen is still lower than 1.0 (which corresponds to the fitness of the wild-type variant), while the upper quartile fitness barely surpasses 1.0, revealing that a PPLM alone might not be an exceptionally effective PE solution, but can be helpful as a sampling tool in a MLDE framework.

ProGen2 [51] is a suite of autoregressive transformers with common architecture. The individual models differ in scale having 151M, 764M, 2.7B, and 6.4B parameters, respectively. A mixture of Uniref90 and BFD30 [52] datasets was used for pre-training with next-token prediction language modeling as the learning objective. Unlike its predecessor, ProGen2 does not make use of conditional tags as inputs, only 24 tokens for amino acids and two extra tokens to mark the protein's N-terminus and C-terminus. The authors evaluate the models' performance in zero-shot fitness prediction on four different protein landscapes. An interesting observation in terms of PE is that the largest model may demonstrate emergent behavior in the identification of the highest fitness variants in the challenging GB1 dataset, even though the metric used to evaluate the performance on the GB1 dataset is unclear.

Other PPLMs exist, which were not described in greater detail. A few of the models are worth at least mentioning. Namely, ProtGPT2 [53] - an autoregressive transformer model with 738 million parameters aimed to generate de novo protein sequences, and ProteinBERT [54], a PPLM with innovative architecture inspired by BERT which combines language modeling with a novel task of Gene Ontology annotation prediction.

2.4 Pre-trained Protein Language Models in Protein Engineering

Given the amount of data PPLMs need for training, the models cannot be fine-tuned using the variants biochemically screened during DE. The screening process is the bottleneck of DE and the number of screened variants is too low for PPLM training by many orders of magnitude. Thus, PPLMs are generally employed in DE without additional fine-tuning on the specific protein's variants or a family of related protein sequences. There are two common ways, in which MLDE methods benefit from PPLMs even with no additional

fine-tuning. Firstly, because of their generative nature, PPLMs can be used to generate suitable protein variants for screening. PPLM-generated variants have been shown to have statistically higher fitness than randomly sampled variants [49, 51]. Secondly, representations of protein sequences can be extracted from hidden layers of PPLMs and used as embeddings. The representations learned by PPLMs carry useful information about the function of the variants [6, 7, 8, 38]. Additionally, a distance metric can be defined on the embeddings, because variants with similar properties can be expected to produce similar embeddings. A sensible distance metric opens the doors for the employment of a plethora of active learning methods, which require a metric of similarity between data points to extrapolate information to unseen data.

The first of the aforementioned uses of PPLMs, sampling of suitable protein variants, is carried out by masking positions in the protein sequence and letting the PPLM predict amino acids with high pseudolikelihood at those positions [55]. The goal is to increase the quality of screened variants and reduce the total amount of screening needed to reach a variant of satisfactory fitness. It has been shown, that even with no fine-tuning, PPLMs generate variants with statistically higher fitness than random selection [49]. Additionally, interpreting the output likelihood of amino acids at a mutation site from a PPLM as variant fitness can be beneficial to ranking the variants [45, 51]. Hie et al. have used six PPLMs (ESM-1b and an ensemble of the five ESM-1v models) to evolve clinically relevant antibodies in a two-round DE [56]. In the first round of an evolution of a wild-type antibody, all single-residue substitutions with higher computed pseudolikelihood than the wild type across a consensus of the six models were selected for screening (the antigen-binding affinity by biolayer interferometry (BLI) was measured). In the second round, single-residue substitutions with preserved or improved fitness over the wild type were combined into a second generation of variants for screening.

The second mentioned way to benefit from a PPLM in DE without the need for additional fine-tuning is to take advantage of the protein representation learned in the hidden layers of the PPLM and use the model to encode protein sequences into the informative embedding in combination with other MLDE methods [9]. In the process of pre-training, the model is trained on a large quantity of unlabeled data to represent protein sequences in a highly informative space [5], which captures properties of the protein important to its structure and function [6, 7, 8]. This representation can be used to train a fitness predictor on a small amount of labeled variants of the target protein [5]. Alternatively, protein embeddings have been used to construct a *manifold* of sequence variation from a sequence similarity network [10], where each sequence is represented by a node and connected by an edge to k-nearest-neighbouring sequences quantified by Euclidean distance in the model-embedding space [30]. Then each edge is assigned a direction and "velocity" based on the language model pseudolikelihood between the two sequences in that edge and an evolutionary vector field [57] is assembled. The constructed *manifold* and the subsequent evolutionary vector field have been previously utilized for fitness landscape visualization [10] and reconstruction of the evolutionary order of protein mutations [30],

respectively. The construction of the similarity networks relies on the fact that sequences with similar embedding can be inferred to have similar properties [5]. Because of this, the networks can also be used to predict functional mutations usable in DE.

An alternative way to utilize the information from the hidden layers of a PPLM for prediction of promising protein variants is to exploit the model's self-attention map instead of the protein sequence embedding as suggested in [58]. A concatenated sequence of a protein variant and its binding target is passed to the ESM-1b model, and *Promise Score*, a metric quantifying how promising the protein variant is in terms of binding with the target sequence, is calculated from the extracted self-attention map.

As we have established, most MLDE methods use PPLMs for protein sampling directly, or to encode proteins into a higher-dimensional embedding. However, Qin et al. have recently exploited PPLMs further in a novel method, the Actively-Finetuned Protein language model for Directed Evolution (AFP-DE) [31]. Apart from employing the PPLM to sample informative variants for screening and as an embedding extractor, they also use the already screened variants to train a multi-layer perceptron to predict fitness from the extracted embeddings. This fitness predictor is then used to select a large number of variants with high predicted fitness to fine-tune the PPLM sampler / embedding extractor. The result is an iterative process which continuously improves both the predictive performance of the perceptron fitness predictor and the generative performance of the PPLM. This method cleverly bypasses the unattainable data requirement to fine-tune a PPLM by using variants with predicted high fitness for fine-tuning, rather than using variants with biochemically screened high fitness, while also exploiting both of the common uses of PPLMs in DE: variant sampling and embedding extraction.

2.5 Passive Sampling & Active Learning

A subfield of machine learning where the learning algorithm is allowed to choose the data for annotation and subsequent learning, often called *queries*, is called active learning [59]. Active learning methods are typically employed when labeling data is expensive and there is an incentive to minimize the amount of data that needs to be labeled to train the model. This corresponds partially to the objective of MLDE. While the goal to minimize labeled data (i.e. screened variants) is present, the objective of MLDE is to obtain a variant with the highest possible fitness rather than training the best possible model (i.e. fitness predictor). With this distinction made, we can label MLDE as an optimization problem rather than a model regression problem. However, active learning algorithms which employ regression loss can still aid DE [31]. With a well-trained fitness predictor, a part of the screening budget can be utilized to screen a number of variants with the highest predicted fitness after the training is complete. These two stages of a model-regression-oriented MLDE method are often referred to as the exploration stage and the exploitation stage [4, 31]. This approach allows for the application of active learning methods to MLDE even though the objectives do not coincide perfectly at first glance.

Depending on the problem, different active learning scenarios can be distinguished. The main scenarios considered in the literature are: *membership query synthesis*, where the model generates a query de novo, *stream-based selective sampling*, in which unlabeled data are sampled sequentially and the model decides whether to query the sampled data or discard them, and *pool-based active learning*, where a large pool of unlabeled data is predefined (or sampled) beforehand and the model selects the best query from the pool [59]. When applying active learning to protein engineering, we would ideally want to consider the complete sequence space and choose the best protein variant to be screened. This approach corresponds to the *pool-based active learning* with the complete protein sequence space acting as the pool of possible queries. In full generality, this is not possible since we have to consider arbitrary sequence length resulting in an infinitely large sequence space. In practice, however, the task is usually restricted to mutating the wild-type sequence by substituting amino acids only, with no additions or deletions, resulting in a finite sequence space. The size of the sequence space is $|\Sigma|^l$, where $|\Sigma|$ is the size of the alphabet (usually the 20 common amino acids are considered [60]) and l is the length of the wild type sequence. While finite, this space is still much larger than the number of atoms in the universe even for relatively short sequences of 100 amino acids [2]. Because of this combinatorial explosion, the task in protein engineering can only be solved as *pool-based active learning* when a small amount of influential mutational sites is predefined by an informed oracle. Only substitutions at these sites are then considered in the active-learning procedure, to heavily limit the size of the sequence space. If such sites are not previously identified, we have to limit the pool to a sample from the sequence space or resort to a *membership query synthesis* or *stream-based selective sampling* approach.

In contrast to active learning approaches, passive sampling does not require updating the trained model and relies entirely on the location of data in the feature space [61]. Passive sampling methods can be especially useful in combination with active learning approaches to select the first K samples, which are necessary to build the model as demonstrated in [61]. The authors combine a simple passive sampling approach, *greedy sampling on the inputs* [62], which chooses new samples located farthest away from all previously selected samples in the feature space, with a similar active learning approach, *greedy sampling on the output* [61], which uses the already labeled samples to construct a regression model and chooses new samples with the maximum distance of their predicted output from the labels of all previously selected samples.

2.6 Bayesian Optimization

We have established that MLDE is an optimization problem at its core. This hints at the advantage of methods with an optimization objective rather than a regression, model-learning approach [63]. Bayesian optimization is a prime candidate because it is very data efficient, making it an ideal choice in problems where the evaluation of data points is costly and the objective function space is multimodal [11]. Both of these properties are one of the key difficulties in exploring fitness landscapes [50, 64].

Mathematically, Bayesian optimization finds a global maximizer (or minimizer) of an unknown objective function f

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \tag{1}$$

where $\mathcal{X}$ is the design space of interest [11]. In the case of MLDE, $\mathcal{X}$ represents the space of all of the protein's variants and the objective function f can be viewed as the screening process which returns a fitness value for the input variant $\boldsymbol{x}$. Equation 1 then corresponds perfectly to the objective of MLDE, that is to find the variant with maximum fitness $\boldsymbol{x}^*$.

To solve the problem defined by equation 1 with Bayesian optimization, two things need to be defined first. One of them is a probabilistic surrogate model over the possible objective functions which describes the distribution of objective functions f since we do not know the true shape of the function. The second is an acquisition function that describes how optimal a new data point $\boldsymbol{x}$, or more generally a set of data points, is for evaluation [11]. With these two key ingredients defined, the algorithm starts with the *prior* probabilistic surrogate model. The prior model captures our beliefs about the behavior of the unknown objective function before annotating any data. It is used to select the first data point, or more generally a set of data points, for evaluation, by computing the loss of each data point with the loss function. With the new data point evaluated by the objective function, the *posterior* surrogate model is updated and the next data point for evaluation can be selected in the next iteration.

2.6.1 Gaussian Process

One of the most popular surrogate models used in Bayesian optimization is the Gaussian process (GP) [11, 65, 66]. GP a nonparametric model that is fully characterized by its prior mean function $\mu_0 : \mathcal{X} \to \mathbb{R}$ and its covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ [11, 67]. The mean function represents our prior expectation for the value of the objective function at unobserved data points and the covariance function encodes a metric of similarity between the data. GP has two assumptions. Firstly, for a finite collection of n data points $\boldsymbol{x}_{1:n}$[1], the unknown true objective function values $f_{1:n}$, where $f_i := f(\boldsymbol{x}_i)$, are assumed to be jointly Gaussian. In accordance with this assumption, the prior distribution introduced by the GP is

$$f_{1:n} | \boldsymbol{x}_{1:n} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{K}) \tag{2}$$

where the mean vector $\boldsymbol{m}$ is defined as $m_i := \mu_0(\boldsymbol{x}_i)$ and the covariance matrix $\boldsymbol{K}$ is defined as $K_{i,j} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [11]. This first assumption represents our prior knowledge about the *smoothness* of the objective function, which is introduced into the model by the selection of the covariance function k. Secondly, the (noisy) values $y_{1:n}$, observed at data points $\boldsymbol{x}_{1:n}$, are normally distributed given $f_{1:n}$ [11].

$$y_{1:n} | f_{1:n}, \sigma^2 \sim \mathcal{N}(f_{1:n}, \sigma^2 \boldsymbol{I}) \tag{3}$$

[1]Throughout this thesis $\boldsymbol{x}_{1:n}$ is used as a short-hand notation for $\{\boldsymbol{x}_i\}_{i=1}^n$.

This second assumption allows the GP to model noise on the evaluation of the objective function f characterized by its variance σ^2.

In PE, the first assumption can be viewed as the belief that similar proteins have similar extent of the desired property. This is a reasonable assumption and active learning or non-exhaustive optimization methods could not be applied to the problem if it did not hold, since we could not extract information about unobserved proteins from the already evaluated dataset. As with most active learning approaches, the choice of a good similarity metric is key. However, defining such a metric on proteins is not trivial. Employing a sequence distance metric such as the Levenshtein distance [68] on raw protein sequences might not be ideal, especially for use in DE where all of the mutated variants often only differ in a handful of amino acids, while their function can be vastly different. With this thought process, we are coming back to the important advantage that PPLMs provide in PE, that is a sequence embedding space where a more sensible metric of similarity can be defined, as discussed in section 2.4.

The second assumption can be understood in the context of PE as the noise present in the biochemical screening experiments. For the sake of evaluation of proposed PE methods, the screening process is often replaced by simply reading a fitness value from a previously measured dataset [49, 51, 9, 31, 63, 69], which understandably has no noise. However, in practice, any biochemical experiment will introduce noise into the measurement and the ability to choose the next variants for screening based on the modeled noise might prove beneficial.

Equation 2 describes the *prior* distribution $p(f_{1:n})$ induced by the GP [11]. After n data points are evaluated and the GP is updated with observations $D_n := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the *posterior* distribution of the GP is given by the posterior mean and variance functions:

$$\mu_n(\boldsymbol{x}) = \mu_0(\boldsymbol{x}) + \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{m}) \tag{4}$$

$$\sigma_n^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}(\boldsymbol{x}) \tag{5}$$

where $\boldsymbol{k}(\boldsymbol{x})$ is a vector of covariance terms between $\boldsymbol{x}$ and $\boldsymbol{x}_{1:n}$ [11]. The posterior mean $\mu_n(\boldsymbol{x})$ represents the model's prediction of the objective function $f(\boldsymbol{x})$ at the point $\boldsymbol{x}$ and the posterior variance $\sigma_n^2(\boldsymbol{x})$ represents the uncertainty of the prediction at the point $\boldsymbol{x}$ [11]. The posterior functions from equations 4 and 5 are used in each iteration of BO to compute the value of the acquisition function at each data point and select the optimal data point $\boldsymbol{x}_n + 1$ (or a set of data points) for evaluation.

2.6.2 Bayesian Optimization in Protein Engineering

In a review of machine-learning-assisted PE methods [65], the authors state that BO can be beneficial in guiding the exploration-exploitation trade-off [70] based on the selected acquisition function and that the modeled uncertainty can help improve exploration in

batched acquisition [71, 72, 73, 74, 75]. They also list two notable acquisition functions widely used in PE applications. The upper confidence bound (UCB) acquisition function selects the data point with the largest upper confidence bound for evaluation, prioritizing data points that are predicted to be both optimized and uncertain [72]. The relative importance of the prediction and uncertainty can be manually controlled with a weighting parameter [65, 76]. The second notable acquisition function, expected improvement (EI), selects the data point where the expectation over the possible values of the objective function is predicted to have the largest improvement over the current best observation [11, 65, 77]. Similarly to UCB, this approach also strikes a balance between prioritizing data points predicted to be optimized and unexplored data points where the prediction is uncertain. Both methods have been shown to be efficient in the number of function evaluations required to find the global optimum of multi-modal black-box objective functions [77, 78, 79].

UCB has been used in GP regression with a structure-based metric of similarity to provide a probabilistic description of the landscapes for various properties of proteins and to design a cytochrome P450 [80] variant that is more than 5 °C more thermostable than P450 variants previously optimized by different methods and 14 °C more stable than the most stable parent from which it was made [72]. In another work, GP classification and regression models were trained with UCB on expression and localization data from 218 channelrhodopsin [81] variants [82]. Structural similarity obtained by aligning residue-residue contact maps of each variant and counting the number of identical contacts were used as a metric of sequence similarity. In addition to GP regression with UCB crietrion, in [83], the authors first sampled 20 variants from the sequence space that maximized the Gaussian mutual information, which they use to fit the GP before the first iteration of sequential optimization. Lastly, in [76], the authors compare GP trained with UCB to other methods which model uncertainty differently or do not model uncertainty at all. The work highlights GP-based methods as particularly useful and shows consistently strong performance of the GP model as well as a GP fit to multi-layer perceptron residuals [84], which suggests a relatively straightforward way to introduce uncertainty into a neural network.

A GP model with the EI acquisition function has been shown to outperform traditional DE methods in an *in silico* experiment [69]. The authors selected variants for evaluation in batches of 19 and used the squared exponential kernel with Euclidean distances computed from one-hot encoding of the variants at mutated positions. Additionally, before employing BO, 20 variants were randomly sampled and used to fit the GP model in the first iteration. The recent optimization framework for protein DE, termed ODBO [63], combines GP and EI acquisition function with a novel low-dimensional, function-value-based protein encoding strategy and prescreening outlier detection. A protein variant is represented by a feature vector, where each amino acid from the sequence is replaced by the mean or maximum value of the fitness measurements of all variants with the amino acid at that position. Then, in each iteration, the vector representations are inputted into the

prescreening via *Extreme Gradient Boosting Outlier Detection* (XGBOD) [85] which filters out potential low fitness samples before the BO step. The authors argue that the novel representation creates a smoother local variable for regression while the prescreening aims to perform more efficient acquisitions in each iteration.

3 Methods

This section covers datasets used in conducted experiments, proposed MLDE methods and implemented simulations of traditional DE methods (not assisted by machine learning) which serve as benchmarks for the MLDE methods. Code for the implemented MLDE procedures and DE simulation benchmarks, as well as the used datasets, are available at `https://github.com/soldatmat/PELLM`. The proposed MLDE methods were implemented in a unified modular framework for in silico DE, which is made available separately as the DESilico.jl package [86].

3.1 Data

Two datasets were used to evaluate the implemented methods and benchmarks. Each dataset maps the fitness landscape of a different wild-type protein. The datasets consist of variant-fitness pairs of (nearly) all possible variants of the wild-type protein mutated at 4 positions. These positions were selected as largely influential to the structure and function of the protein. In both datasets, the 20 common amino acids are considered [60], resulting in 160,000 possible variants (20^4 for 20 common amino acids at 4 mutated positions). Not all 160,000 variants are measured in each dataset. The number of missing variants in each dataset is specified in the following subsections 3.1.1 and 3.1.2. The fitness of each unmeasured variant is assumed to be zero in all conducted experiments and benchmarks since the unmeasured variants are considered meaningless to biologists [9, 31].

3.1.1 GB1 dataset

The GB1 library [50] is a dataset of protein sequences of variants of the protein G domain B1, mutated at four positions with non-linear epistasis (V39, D40, G41, V54). This means that all of the variants differ at most in four amino acids from the wild-type protein and no insertions or deletions are present in the dataset. The dataset contains fitness measurements for 149,361 of the 160,000 possible variants. The fitness values represent the binding ability of different GB1 variants to the antibody IgG-Fc and range from 0.0 (minimum fitness, variants with the worst function) to 8.76 (maximum fitness of the variant with the best function), where a fitness value of 1.0 corresponds to the binding ability of the wild type protein. The GB1 fitness landscape contains multiple fitness peaks and is heavily populated by zero-fitness and low-fitness variants. This makes it a challenging dataset in terms of DE. Only 3,644 of the 149,361 measured variants have a fitness value of 1.0 or greater.

3.1.2 PhoQ dataset

The PhoQ dataset [87] consists of fitness values for variants of protein kinase PhoQ obtained by mutating the wild-type sequence at four positions critical to the function of the protein (A284, V285, S288, T289). The dataset contains fitness measurements for 140,517 of the 160,000 possible variants. The fitness values in the dataset refer to the phosphatase or kinase activity of different PhoQ variants and range from 0.0 (minimum fitness, variants with the worst function) to 133.59 (maximum fitness of the variant with the best function). The wild-type protein kinase PhoQ has a measured fitness of 3.29. Only 1,659 of the 140,517 measured variants were identified as functional. As such, the dataset is even more heavily populated by non-functional variants and is considered more challenging in terms of DE than the GB1 dataset.

3.2 Benchmarks

Two traditional (not assisted by machine learning) DE procedure simulations were implemented to serve as benchmarks for the efficiency of proposed MLDE procedures in future work. The simulations represent possible DE procedures as they would be carried out in a laboratory without the help of machine learning with the distinction that the biochemical tests of fitness are replaced with simply reading fitness data from an already measured dataset of the fitness of variants of a protein.

3.2.1 Single Mutation Walk

The first implemented procedure simulates a single mutation walk as described in [88]. The procedure starts with a starting protein variant (e.g. the wild-type protein) and a list of pre-selected mutation positions (typically positions known to be influential to the protein's function). In the first round, all possible single amino acid mutations are tested at each mutation position, and the mutation resulting in the highest-fitness variant is fixed (the mutated amino acid is fixed at the mutation position and the position is removed from the list of mutation positions). In consecutive rounds of the procedure, all possible single amino acid mutations are tested again, but only at the remaining mutation positions in the list. The procedure ends after n rounds, where n is the number of the pre-selected mutation positions, because an amino acid is fixed at one position in each round. Alternatively, the procedure can end prematurely, if there are no single amino acid mutations which would improve the fitness of the currently mutated protein variant in the current round. In that case, the current variant is outputted as the result. Unless the procedure ends prematurely, the number of screened variants N is

$$N = \frac{n(n+1)}{2} \cdot (|\Sigma| - 1) \tag{6a}$$

where the fraction represents the sum of all positions at which the protein was mutated during all rounds $(n + (n-1) + ... + 1)$ and $|\Sigma|$ is the size of the alphabet. Considering

the 20 common amino acids [60], $|\Sigma|$ is equal to 20 and the number of screened variants is

$$N = \frac{n(n+1)}{2} \cdot 19 \tag{6b}$$

The procedure can be interpreted in terms of the two DE steps defined in section 2.2 (mutagenesis and screening) as follows. The mutagenesis step starts with a single sequence and prepares a library of all single amino acid mutations of the starting sequence. In the screening step, the fitness of each mutated variant is evaluated and only the highest-scoring variant is selected to parent the next library in the following iteration with the difference that the already mutated positions are being gradually fixed until no position remains to be mutated.

Since the single mutation walk simulation is deterministic for a fixed starting protein variant, the simulation was run with each of the GB1 variants as the starting variant for a total of 160,000 runs. During each run of the simulation, 190 variants were screened, as is evident from equation 6b by substituting $n = 4$ for the four preselected mutation sites in GB1.

3.2.2 Recombining Mutations

The second simulation is inspired by a procedure described in [88] as *Recombining Mutations in Best Variants*, but differs slightly. This procedure, which will be referred to as *Recombination*, is not iterative. It starts by sampling m variants from the whole combinatorial space over n pre-defined mutation positions. The fitness of all m variants is tested and top k variants are recombined for a maximum recombinational library size of k^n (maximum, because the library can be smaller if some of the top k variants share amino acids at one or more positions). The output variant is selected by screening all variants in the recombinational library. This procedure results in a maximum of N variants needed to be screened, where

$$N \leq m + k^n - k \tag{7}$$

m represents the initially sampled variants, k^n are all variants recombined from the top k selected parents and $-k$ stands for the k parent variants, which are included in both m and k^n.

The non-deterministic recombination simulation was repeated 160,000 times to obtain results comparable to the single mutation walk simulation. The maximum value of N was set to 190 to maintain the same cost of the DE process across both implemented benchmarks. The number of the top variants which are to be recombined was set to $k = 3$, which results in at most 78 ($k^n - k$) newly recombined variants that will be screened. The choice of k is motivated by striking a balance between resources allocated to the initial sampling of variants and the newly recombined variants. For $N \leq 190$ and $k = 3$, equation 7 gives us $m = 112$. This means that 112 variants can be sampled in the

initial step of the procedure to guarantee that the total number of screened variants does not exceed 190.

3.3 Protein Embedding Fitness Predictor

The first proposed MLDE method is based on the AFP-DE procedure but implements a different *exploration* stage than the original paper [31]. It employs an active learning approach with model regression as its objective. The goal is to train a two-layer perceptron to predict the fitness of variants from their embedding extracted by a PPLM and then screen a number of variants with the highest predicted fitness. Therefore, in further sections, this method is referred to as the perceptron-training algortihm. The ESM-1b model [6] was chosen as the embedding extractor as a common choice in the literature [30, 31]. The choice is motivated by elimination of potential problems with novel models where the informativeness of the extracted embeddings for fitness prediction might not be verified by conducted experiments. The benefit of predicting protein fitness from the extracted embedding rather than the raw sequence is twofold. Firstly, the embedding carries additional useful information about the variant's structure and function learned during the pre-training of the PPLM [6, 7, 8, 38]. Secondly, the embedding space allows for a definition of a more sensible and informative metric of similarity between variants than the raw sequence space.

The algorithm starts with the known wild-type sequence and the chosen pre-trained LM which will serve as the embedding extractor. In the first stage of the algorithm, 23 variants are passively sampled to obtain an initial dataset of 24 screened sequences. The passive sampling is carried out via *greedy sampling on the inputs* [62] which iteratively selects the variant with maximum distance from all previously selected variants in the input space. In the context of this method, the distance of two variants in the input space is understood as the Euclidean distance between the two variants' extracted embeddings.

The initial passive sampling is followed by an iterative active-learning procedure via *greedy sampling on the output* [61]. The procedure starts by training the fitness predictor on the 24 passively sampled and screened variants. Then, an iteration of the active-learning procedure begins by predicting the fitness of all un-screened variants by the fitness predictor. From the un-screened variants, 24 new variants with maximum difference in fitness to all previously screened variants are selected for screening. To evaluate the difference in fitness, predicted fitness is used for the un-screened, candidate variants, and the true screened fitness is used for the variants already in the training set. An iteration is completed by screening the selected variants and training the predictor with these 24 newly screened variants.

Lastly, a part of the screening budget needs to be spent on evaluation of variants with high predicted fitness. The algorithm itself, as any model regression active learning ap-

Algorithm 1 Greedy sampling on the input

Input: All variants $\mathcal{X}$

Output: Set of k selected variants

procedure InputDistmax($\mathcal{X}$)
  Initiate set of candidates $\mathcal{C}_1 \leftarrow \{\boldsymbol{x}_{wt}\}$ with the wild-type protein
  for i = 1, 2, ..., k **do**
    Select variant $\widehat{\boldsymbol{x}}$ with maximum distance from variants in $\mathcal{C}_i$

$$\widehat{\boldsymbol{x}} \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{X}} \min_{\boldsymbol{x}' \in \mathcal{C}_i} d(\boldsymbol{x}, \boldsymbol{x}')$$

    Update set of candidates $\mathcal{C}_{i+1} \leftarrow \mathcal{C}_i \cup \{\widehat{\boldsymbol{x}}\}$
  end for
  return $\mathcal{C}_{k+1}$
end procedure

$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ... Distance function between variants.
We use Euclidean distance of the variants' embeddings.

Algorithm 2 Greedy sampling on the output

Input: Fitness values of screened variants T
Fitness predictor $\mathcal{F}$
Un-screened variants $\mathcal{X}$

Output: Set of k selected variants

procedure OutputDistmax($T, \mathcal{F}, \mathcal{X}$)
  Initialize set of candidates $\mathcal{C}_1 \leftarrow \varnothing$
  Initialize set of observed fitness values $\mathcal{V}_1 \leftarrow T$
  Predict fitness of unscreened variants $\mathcal{P} \leftarrow \{(\boldsymbol{x}, \mathcal{F}(\boldsymbol{x})) | \boldsymbol{x} \in \mathcal{X}\}$
  for i = 1, 2, ..., k **do**
    Select variant $\boldsymbol{x}_j$ with maximum distance from variants in $\mathcal{V}_i$

$$(\boldsymbol{x}_j, y'_j) \leftarrow \arg\max_{(\boldsymbol{x}_j, y'_j) \in \mathcal{P}} \min_{y \in \mathcal{V}_i} |y'_j - y|$$

    Update set of candidates $\mathcal{C}_{i+1} \leftarrow \mathcal{C}_i \cup \{\boldsymbol{x}_j\}$
    Update set of observed fitness values $\mathcal{V}_{i+1} \leftarrow \mathcal{V}_i \cup \{y'_j\}$
  end for
  return $\mathcal{C}_{k+1}$
end procedure

proach, does not prioritize high-fitness variants for screening but chooses variants that will be informative to the trained model instead. These two objectives usually do not coincide as the former pushes more for exploitation, while the latter rewards exploration in the exploration-exploitation trade-off [70].

Algorithm 3 Perceptron-training method

Input: All variants $\mathcal{X}$

Output: Best screened variant $(\boldsymbol{x}, y)$

1: Select set of initial variants $\mathcal{X}_0$ by InputDistmax (Alg. 1)
2: Initialize set of screened variants $\mathcal{D}_0 \leftarrow \{(\boldsymbol{x}, f(\boldsymbol{x}))|\boldsymbol{x} \in \mathcal{X}_0\}$
3: Extract embeddings $\mathcal{E}_0 \leftarrow \{(\mathcal{H}(\boldsymbol{x}), y)|(\boldsymbol{x}, y) \in \mathcal{D}_0\}$
4: Train predictor $\mathcal{G}$ on $\mathcal{E}_0$ embeddings with fitness values
5: **for** $i = 1, 2, \ldots, n$ **do**
6: Select informative variants $\mathcal{X}_i$ by OutputDistmax (Alg. 2)
7: Screen selected variants $\mathcal{C}_i \leftarrow \{(\boldsymbol{x}, f(\boldsymbol{x}))|\boldsymbol{x} \in \mathcal{X}_i\}$
8: Extract embeddings $\mathcal{E}_i \leftarrow \{(\mathcal{H}(\boldsymbol{x}), y)|(\boldsymbol{x}, y) \in \mathcal{C}_i\}$
9: Train predictor $\mathcal{G}$ on $\mathcal{E}_i$ embeddings with fitness values
10: **optional:** Train embedding extractor $\mathcal{H}$ on m variants with top predicted fitness
11: Update set of screened variants $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup \mathcal{C}_i$
12: **end for**
13: Screen $\mathcal{X}_p$ set of p variants with top predicted fitness $\mathcal{D} \leftarrow \mathcal{D}_n \cup \{(\boldsymbol{x}, f(\boldsymbol{x}))|\boldsymbol{x} \in \mathcal{X}_p\}$
14: **return** best screened variant $(\boldsymbol{x}, y) \leftarrow \arg\max_{(\boldsymbol{x},y)\in\mathcal{D}} y$

$\mathcal{H} : \mathcal{X} \to \mathcal{E}$	...	Embedding extractor, uses PPLM to extract embedding from protein sequence.
$\mathcal{G} : \mathcal{E} \to \mathbb{R}$	...	Perceptron network, predicts fitness from sequence embedding.
$\mathcal{F} : \mathcal{X} \to \mathbb{R}$	...	Final fitness predictor $\mathcal{F} = \mathcal{G}(\mathcal{H}(\boldsymbol{x}))$.
$f : \mathcal{X} \to \mathbb{R}$	...	Screening, assigns fitness to a variant.

3.3.1 Fitness Predictor Architecture

The two-layer perceptron trained as a fitness predictor consists of two linear layers and an activation function. The first layer is a fully connected linear layer with equal number of input and output features, which corresponds to the size of the sequence embedding extracted by the chosen PPLM. The used PPLM, ESM-1b, produces embedding with 1280 dimensions, resulting in a 1280×1280 sized first layer. The second linear layer is a fitness-prediction head with input size equal to the size of the sequence embedding and a scalar output. Lastly, a sigmoid function is used for the activation function, since the fitness values are normalized to 0 to 1 range before training.

3.3.2 Training the Fitness Predictor

The fitness predictor was trained in 5 epochs with a batch size of 1 variant using the L1 regression loss. To update the model's parameters, the Adam algorithm [89] implementation from PyTorch [90] with 10^{-5} learning rate was used. The choice of the optimizer as

well as the learning rate value was motivated in a different active-learning approach from the literature, in which the authors also employed a two-layer perceptron fitness predictor [31].

3.3.3 Fine-tuning the Language Model

In a variation of the proposed algorithm, the PPLM used as a sequence embedding extractor is actively fine-tuned in each iteration of the active learning procedure. The fine-tuning takes place after the training of the fitness predictor. The fitness predictor is used to select the top 8000 variants with the highest predicted fitness (from among the screened and un-screened variants without distinction) and the embedding extractor is fine-tuned on these 8000 selected variants. Then, the fine-tuned embedding extractor provides new embeddings to the fitness predictor before 24 new variants for screening are selected.

The fine-tuning of the embedding extractor is conducted through a masked-language modeling procedure. The training sequences are masked at the mutation positions of interest and the model is trained to predict the masked tokens with cross-entropy loss. The Adam algorithm [89] with decoupled weight decay regularization [91], implemented in PyTorch [90] as *AdamW*, is used to update the model's parameters with 10^{-5} learning rate. The training is conducted in one epoch with batches of 10 sequences.

3.4 Neighborhood Search Directed Evolution

The second proposed method termed Neighborhood Search Directed Evolution (NSDE) takes inspiration from the similarity networks previously used in fitness landscape visualization [10] and reconstruction of evolutionary paths of proteins [30]. The algorithm starts with the computation of the distance between each pair of variants in the sequence space. The distance between two variants is defined as the Euclidean distance of their sequence embedding extracted by a PPLM. A neighborhood of each variant is constructed by selecting its k nearest neighbors and an oriented edge is constructed from the variant to all of its neighbors.

The algorithm proceeds as a greedy graph search. The search starts with the wild-type protein and 9 more passively sampled variants for a total of 10 initially screened variants in the *open* set of unexplored variant-fitness pairs. The passive sampling is carried out via *greedy sampling on the inputs* [62], identically to the passive-sampling procedure used for initial training of the perceptron described in section 3.3. In each iteration of the graph search, the variant with the highest fitness among the variants in the *open* set is explored. Exploring a variant consists of two actions. First, the explored variant is removed from the *open* set and added to the *closed* set of already explored variants. Second, all neighbors of the explored variant, which are not present in either the *open* set or the *closed* set are screened and added to the *open* set together with their true fitness.

Algorithm 4 NSDE

Input: All variants $\mathcal{X}$ represented in embedding space
Output: Best screened variant $(\boldsymbol{x}, y)$

Compute pair-wise distances between variants $d_{ij} \leftarrow dist(\boldsymbol{x}_i, \boldsymbol{x}_j)$
Construct neighborhoods $\mathcal{N}_i \leftarrow \{\boldsymbol{x}_j | d_{ij} \in k \text{ smallest distances among } d_i.\}$
Select set of initial variants $\mathcal{I}$ by InputDistmax (Alg. 1)
Initialize open set $\mathcal{O} \leftarrow \{(\boldsymbol{x}, f(\boldsymbol{x})) | \boldsymbol{x} \in \mathcal{I}\}$
Initialize closed set $\mathcal{C} \leftarrow \varnothing$
for $n = 1, 2, \ldots$ **do**
    Select variant with highest fitness $(\boldsymbol{x}_i, y_i) \leftarrow \arg\max_{(\boldsymbol{x},y)\in\mathcal{O}} y$
    Screen the neighborhood $\mathcal{O} \leftarrow \mathcal{O} \cup \{(\boldsymbol{x}, f(\boldsymbol{x})) | \boldsymbol{x} \in \mathcal{N}_i \setminus \mathcal{C}\}$
    Close the variant $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{x}_i\}$
end for
return highest-fitness variant among variants screened with f

$f : \mathcal{X} \to \mathbb{R}$ ... Screening, assigns fitness to a variant.

3.4.1 Implementation Details of NSDE

The ESM-1b model [6] was chosen as the embedding extractor to unify the embedding extractors across the proposed methods. The size of the neighborhoods k was set to 16. The choice of k is motivated in order of magnitude by the values used in [30], where values of k around 30 to 50 are observed to provide a good balance between robustness and computational efficiency in different KNN-based analyses. A considerably lower value relative to the consulted literature was selected to accommodate for the objective of MLDE to minimize the amount of conducted screening. The analyses in the literature are not motivated by the minimization of screening effort. With a lower k the algorithm can run for more iterations with the same screening budget.

3.5 Bayesian Optimization in Embedding Space

The third proposed method, abbreviated to BOES, performs BO in the embedding space extracted by a PPLM instead of using the uninformative, categorical space defined by the amino acids at the mutated positions. BOES employs BO with a GP as the fitted model and EI as the acquisition function. The GP model is defined with zero prior mean function $\mu_0 : \mathcal{X} \to 0$ and Euclidean distance of the variants' PPLM-extracted embeddings as the covariance function k (see section 2.6.1). Zero variance σ^2 is used for noise, effectively removing noise from the model, because the experiments are conducted on a noiseless dataset. The algorithm starts with only the wild-type protein in the set of observations $D_1 = \{(\boldsymbol{x}_{wt}, y_{wt})\}$. In each iteration of BO, the GP is fitted to the set of observations (already screened variants), the EI acquisition function is evaluated at each data point (each variant) and the variant with maximal acquisition function value is selected, screened, and added to the set of observations with its true fitness value.

A fundamental problem of employing BO in the embedding space of a PPLM, and the probable reason why this approach is not used widely in the literature, is that BO struggles with high dimensional input spaces [11, 63]. This is problematic because PPLM embeddings tend to have a size in orders of 10^2 to 10^3, depending on the architecture of the language model. This means that we are trying to run BO in an input space with potentially thousands of dimensions. To solve this issue, BOES proposes a custom kernel to limit the effective number of dimensions to one, so that the surrogate model only fits one length scale hyperparameter instead of 1280 hyperparameters (1280 is the size of the ESM-1b embedding). This singular length scale corresponds to the scalar result of the Euclidean distance computed on the extracted embeddings. For the singular length scale's prior distribution, a normal distribution with zero mean and standard deviation σ of $\frac{\sqrt{1280}}{3}$, truncated (and normalized) to $[0; \infty)$ interval, is used. The value used for the standard deviation was chosen so that the diagonal across the high-dimensional embedding space corresponds approximately to 3σ. Since the embedding space of the used model, ESM-1b, has 1280 dimensions and the absolute values of the elements in the protein embeddings rarely exceed 1 (0.3 % of the elements from all GB1 embeddings have absolute values higher than 1), the size of the diagonal is roughly $\sqrt{1280}$. If we want the diagonal to correspond to three standard deviations, the standard deviation has to be a third of the size of the diagonal.

Algorithm 5 BOES

Input: All variants $\mathcal{X}$ represented in embedding space
Output: Best screened variant $(\boldsymbol{x}, y)$

1: Initialize dataset $\mathcal{D}_1 \leftarrow \{(\boldsymbol{x}_{wt}, f(\boldsymbol{x}_{wt}))\}$ with the wild-type protein
2: Fit the model GP_1 given $\mathcal{D}_1$
3: **for** $n = 1, 2, \ldots, k$ **do**
4: Select new variant for screening by optimizing EI

$$\boldsymbol{x}_{n+1} \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{X}} EI(\boldsymbol{x}; GP_n)$$

5: Screen selected variant $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n \cup \{(\boldsymbol{x}_{n+1}, f(\boldsymbol{x}_{n+1}))\}$
6: Fit the model GP_{n+1} given $\mathcal{D}_{n+1}$
7: **end for**
8: **return** best screened variant $(\boldsymbol{x}, y) \leftarrow \arg\max_{(\boldsymbol{x},y) \in \mathcal{D}_{k+1}} y$

EI	...	Expected Improvement acquisition function.
GP_n	...	Gaussian process model fitted to dataset D_n.
$f : \mathcal{X} \to \mathbb{R}$	...	Screening, assigns fitness to a variant.

3.5.1 Implementation Details of BOES

The BOES procedure is implemented with the BOSS.jl package [92]. The model is fitted with MLE by the NEWUOA algorithm [93] with 20 starts in a multi-start setting and lower bound on the trust region radius $\rho_{end} = 10^{-4}$. The zero noise variance σ^2 is replaced with

a very small positive value by the BOSS.jl package to ensure numerical stability of the model. To avoid wasting the screening budget on already screened variants, the value of the acquisition function computed for each already screened variant is replaced by zero before the next variant for screening is chosen. This ensures that the screened variants cannot be chosen again unless the acquisition function value of all variants in the sequence space is also zero, which is practically impossible. The ESM-1b model [6] is used as the embedding extractor, same as in the other proposed methods.

3.6 Evaluation of the Proposed Methods

The most important metric in evaluation of MLDE methods is the highest fitness among the screened variants related to the number of screened variants. This metric corresponds directly to the objective of MLDE which is to minimize the number of conducted screening while maximizing the highest observed fitness. This section describes the experiments designed to compare the performance of the proposed MLDE methods and implemented DE benchmarks.

3.6.1 Performance on the Wild-type Protein

A straightforward method of evaluation of the proposed methods, which is the most informative in terms of the objective of MLDE, is a simulation of the DE procedure started from the wild-type protein. This evaluation method represents the practical application of the proposed MLDE algorithms, where a wild-type protein with a desirable function is already known and the goal is to find mutations of the protein which improve upon that function.

The evaluation on the wild-type protein was carried out by running the evaluated MLDE method with the wild-type protein and its true fitness known at the start of the algorithm. In the reported results, the wild-type protein is counted as the first screened variant, i.e. it is also counted towards the screening budget. During the execution of the evaluated algorithm, the fitness of the best-so-far variant among the screened variants was recorded after each conducted screening. The result of this evaluation is a non-decreasing fitness progression of length equal to the number of conducted screening, where the ith element in the progression represents the fitness of the best-so-far variant among the first i screened variants. This fitness corresponds to the result which the evaluated MLDE algorithm would obtain given a screening budget of i screened variants.

This evaluation procedure was conducted with each of the proposed MLDE algorithms as well as the SMW benchmark on both the GB1 and PhoQ datasets. Note that the recombination benchmark, described in section 3.2.2, was not considered for this evaluation because the procedure revolves around random sampling of variants and the starting protein plays no role. Furthermore, the procedure is highly non-deterministic and the result of a single run of the algorithm bears very little information.

3.6.2 Robustness to the Starting Protein

While the performance on the wild-type protein corresponds the best to the practical use of MLDE algorithms, comparing the proposed methods based on a single run, albeit in two different datasets, might not be very informative. The results obtained from such limited evaluation can be strongly skewed by properties of the specific dataset. Especially local-search methods, like the SMW benchmark or the proposed NSDE method based on a KNN-constructed graph, could potentially show wildly different efficiency based on the relative position of the starting variant, the global optimum, and any local optima in between in the sequence space.

To ensure that the position of the wild-type protein in the two used datasets is not unreasonably beneficial to some of the methods and that the methods' hyperparameters are not over-fitted to the path from the wild-type variant to the global optimum, a test of robustness to the starting protein was conducted. This test also gives helpful insight into the variance in performance of any given combination of DE method and size of the screening budget, how often a given method be expected to perform significantly better or worse than usual, and how different are the outlying results from the usual performance of the method.

The evaluation of the robustness of the proposed methods was carried out as follows. Each method of DE was run repeatedly with a different variant from the used dataset serving as the wild-type protein given to the algorithm at its start. For different methods and benchmarks, all 160,000 variants from the dataset or a smaller sample of variants were used, based on the computational demand of the considered method. From each separate run, a non-decreasing progression of the best-so-far fitness was recorded in the same manner as when measuring performance on the wild type protein, described in section 3.6.1, resulting in a set of $n_b \times n_r$ values where n_b is the size of the screening budget and n_r is the number of runs of the evaluated method. For each number of screened variants from 1 to n_b, first quartile, median, and third quartile values among the n_r runs were computed and reported. The starting variant is also counted towards the number of screened variants, same as in the evaluation of performance on the wild-type protein.

4 Results

In this section, the proposed methods are compared to the implemented DE benchmarks in terms of their performance when started from the wild-type protein and when run from sampled starting variants with zero or close-to-zero fitness. The former form of evaluation corresponds to the real application of MLDE methods while the latter provides more robust results, more suitable for conclusive statements about the methods' performance. Figures 5b and 6b relate to the wild type runs and figures 5a, 6a, 7, 8, 9 and 10 all report results of the sampled runs. Additionally, a visualization of the employed embedding space is included at the beginning of this section. The visualization serves as a proof of feasibility

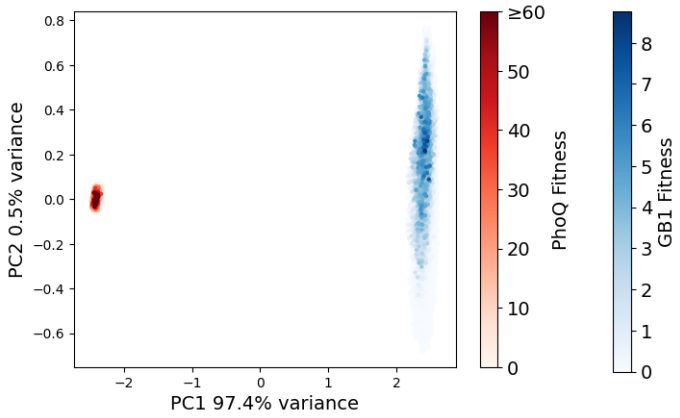


Figure 2: Joint PCA of sequence embeddings from GB1 and PhoQ datasets extracted with ESM-1b PPLM.

of the proposed methods, which all rely heavily on the informativeness of the extracted embeddings. As a secondary result, the fitness landscape predicted by BOES is visualized alongside the original embedding space. The modeled fitness landscape is discussed at the end of this section.

4.1 Visualizing the Embedding Space

All three of the proposed methods operate on a PPLM-extracted sequence embedding space instead of using the raw sequences of amino acids. The ESM-1b model [6] was used as the embedding extractor in each method. The embedding space should provide a sensible metric of similarity between the variants as well as encode useful information about the variants' properties. To ensure that these assumptions hold, the embedding space was visualized with dimensionality reduction methods.

First, a joint principal component analysis (PCA) was conducted on sequence embeddings of all variants from both datasets (GB1 and PhoQ). The goal of this analysis was to confirm that variants of the two completely different proteins are clearly separable in the embedding space. The variants are plotted in a two-dimensional space defined by the first two principal components in figure 2. The results confirm that the two datasets are easily separable. The first principal component alone accounts for 97.4 % of variance in the joint distribution and separates the two datasets into two clear clusters.

Next, a PCA was conducted for each datasets separately to visually confirm whether some expected features of the sequence space, like local maxima and distinguishable areas with low/high fitness, are present in the embedding space. In both the PCA of the GB1 dataset, in figure 3a, and the PCA of the PhoQ dataset, in figure 4a, the first two principal components together explain roughly 40 % of the variance. That is a very large portion,

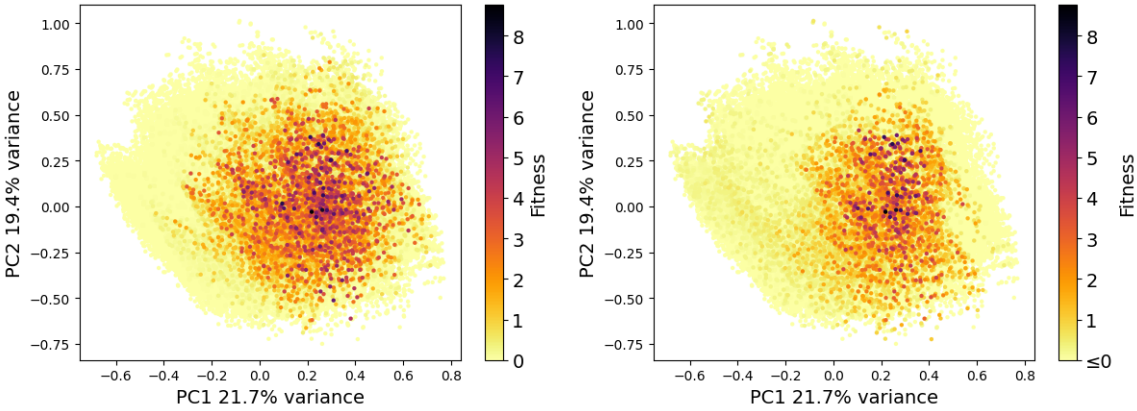
considering that the ESM-1b embedding space has 1280 dimensions. The large portion of explained variance warrants that any observations made can probably be generalized to the high dimensional space to a reasonable extent. The PCA analyses of the standalone datasets both show one large area with functional variants surrounded by many non-functional variants. It should be noted that low-fitness variants are also present in the high-fitness area of the two-dimensional space, even though they are not visible under the high-fitness variants. These results confirm that the functional variants are situated together. Furthermore, in the PCA of the GB1 dataset in figure 3a the highest-fitness variants are situated in the middle of the functional area with decreasing fitness towards the edges of the graph. These properties are very promising in terms of optimization in the embedding space.

However, the PCA analyses did not reveal whether the embedding space is capable of capturing local maxima, which are present in fitness landscapes. To assess this property, the embedding space of each dataset was visualized with the t-SNE method, which emphasizes maintaining low distances between close data points. This way, local clusters of functional variants should be preserved. The results of the t-SNE method are plotted in figure 3c for the GB1 dataset and in figure 4c for the PhoQ dataset. Both figures confirm the presence of local clusters of high-fitness variants. Especially in the GB1 t-SNE plot, very small amount of variants with very high fitness are situated outside of a few clusters. The t-SNE analysis also paints a promising picture for the employment of optimization methods in the embedding space.

4.2 Performance on the Wild-type Protein

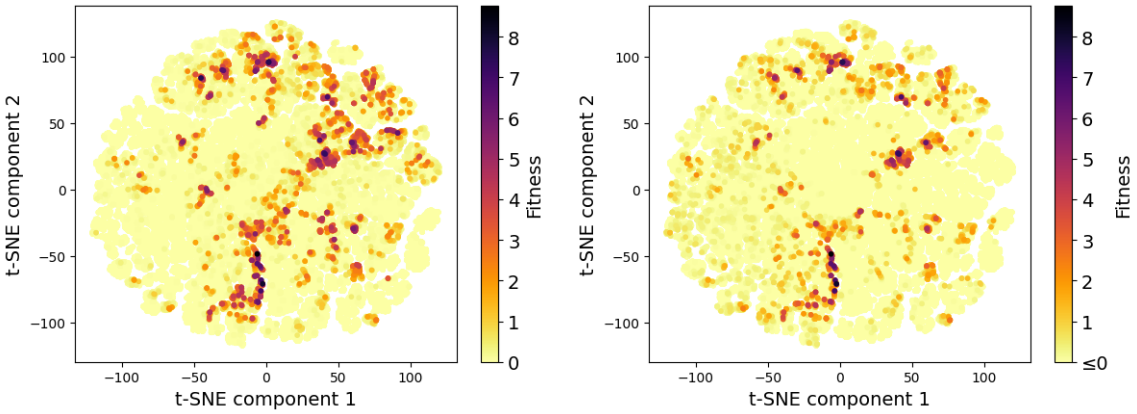
The best-so-far fitness progressions obtained through the DE simulations starting from the wild-type protein are plotted in figures 5b and 6b for the GB1 and PhoQ datasets, respectively. The vertical axis shows the best-so-far fitness obtained after screening a number of variants specified by the horizontal axis. The wild-type protein is included in the graphs as the 0th variant so that the number of screened variants on the horizontal axis corresponds to an experimental screening budget of a real DE procedure, in which the wild-type protein is typically known and not counted towards the budget. Each evaluated method is represented by a single non-decreasing fitness progression starting from the non-zero fitness of the wild-type protein and increasing as the method identifies protein variants with higher fitness. From the two implemented DE benchmarks, only the SMW method is evaluated on the wild-type protein performance, since this form of evaluation is not sensible for the recombination benchmark method as explained in section 3.6.1.

When comparing the results of the evaluated methods, it is important to note, that the objective of the perceptron-training algorithm is not directly tied to optimization of the best-so-far fitness but to training of a fitness predictor. The choice of variants for screening during the run of the algorithm is motivated by the variants' informativeness to the trained model rather than directly by the variants' quality in terms of fitness. The plotted curves



(a) PCA with true fitness.

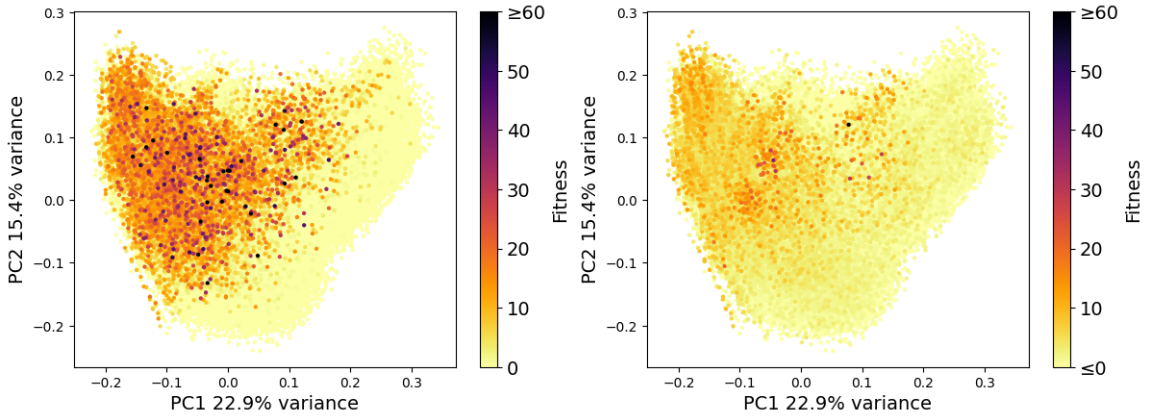
(b) PCA with predicted fitness.



(c) t-SNE with true fitness.

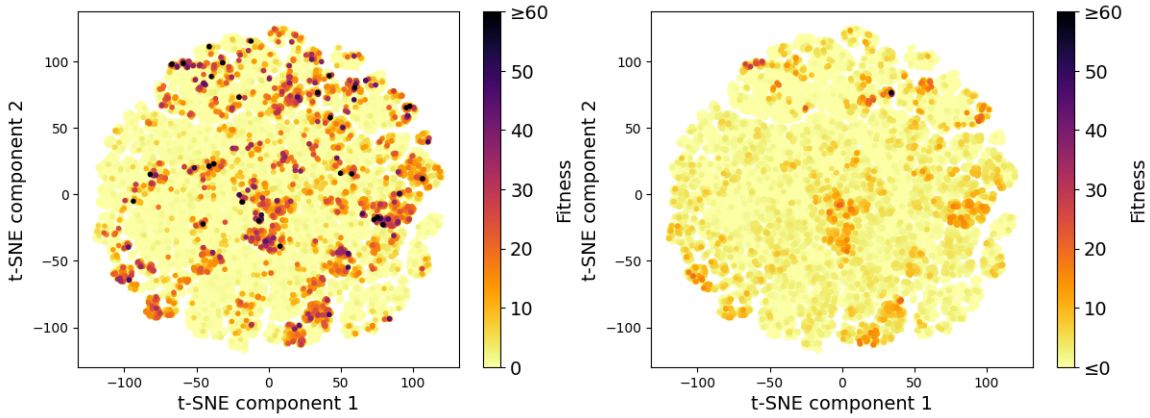
(d) t-SNE with predicted fitness.

Figure 3: Visualisation of GB1 embedding space extracted with ESM-1b PPLM with true fitness (a & c) and fitness predicted by GP model trained on 384 screened variants in BOES run from the wild type protein (b & d).



(a) PCA with true fitness.

(b) PCA with predicted fitness.



(c) t-SNE with true fitness.

(d) t-SNE with predicted fitness.

Figure 4: Visualisation of PhoQ embedding space extracted with ESM-1b PPLM with true fitness (a & c) and fitness predicted by GP model trained on 384 screened variants in BOES run from the wild type protein (b & d).

of the best-so-far fitness are recorded before a part of the screening budget would be used to screen a chosen amount of variants with high fitness predicted by the trained model. To address this concern, the fitness progressions of the perceptron-training algorithm, are accompanied by data points that report maximum true (screened) fitness among the variants already screened for training as well as additional 100 variants with the highest predicted fitness. The size of the additional screening budget dedicated to screening the predicted variants is chosen to be similar to budget sizes dedicated to predicted variants used in literature [31]. These values are reported in intervals of 24 screened variants because 24 is the number of variants screened between training iterations of the fitness predictor.

4.2.1 Performance on GB1 Wild-type Protein

The best-so-far fitness obtained with the SMW procedure on the GB1 dataset, plotted in figure 5b in blue, increases steadily until it plateaus at 3.90 in under 50 variants screened. Then the fitness stagnates and improves considerably again at between 125 and 150 screened variants with a final result of 5.77 after the 190 screened variants, which mark the end of the SMW procedure. To continue the procedure with a larger screening budget, the only option would be to restart the algorithm with the final variant or a randomly sampled variant as the new starting variant. Restarting with a randomly sampled variant introduces non-deterministic behavior where a single run would carry little information, same as with the recombination benchmark. Restarting with the final variant is a more sensible option but the next run of the restarted procedure would partially search the same part of the sequence space making it less effective than an initial run. Because of this, no performance of the SMW benchmark is reported after 191 variants are screened.

The BOES method shows results comparable to the SMW benchmark in approximately the first 40 screened variants. Then, with an increasingly sized screening budget, BOES keeps improving the best-so-far variant in an efficient manner, while the result of the SMW benchmark stagnates. The best-so-far fitness of BOES plateaus at 7.55 around 100 screened variants, after which it takes the method another 150 variants to find the global fitness maximum. The BOES method clearly surpasses the other evaluated methods in performance on the GB1 wild-type protein. Only in the task of finding the globally optimal variant, does the NSDE method more or less tie the number of necessary amount of conducted screening with BOES.

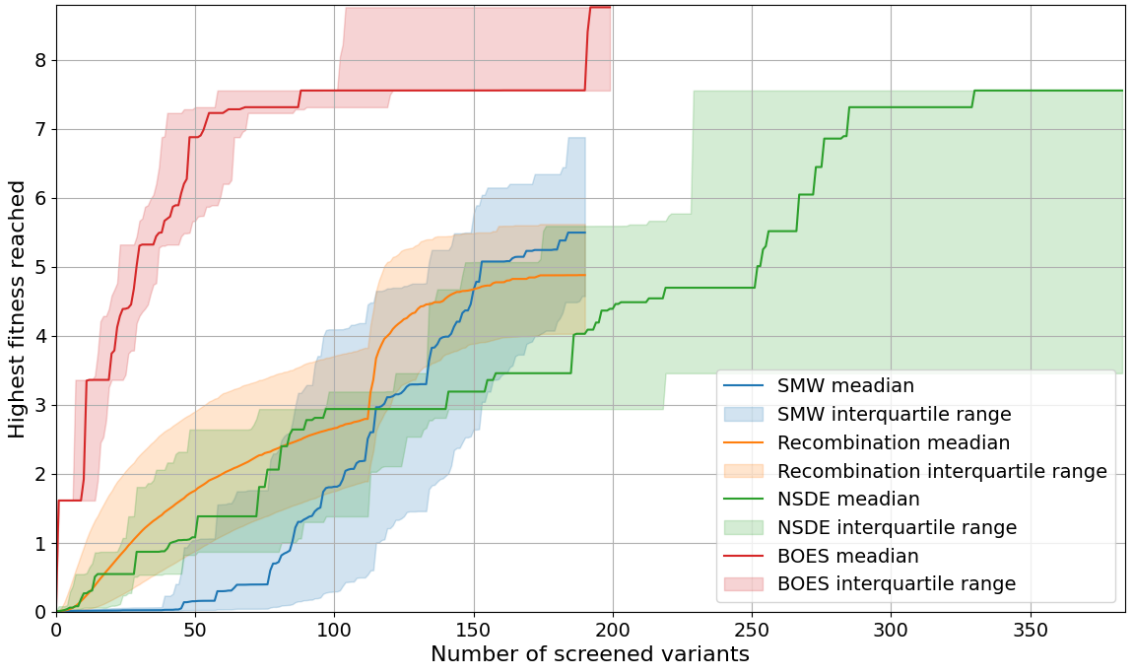
The performance of the NSDE method with the first 190 screened variants follows the results of the SMW benchmark relatively closely. Both of the methods yield slightly better results than the other at different sizes of the screening budget. A considerable advantage of the NSDE method manifests itself after 190 variants are screened, where the SMW benchmark stops while the NSDA method continues and finds the global maximum with a screening budget of 248 variants. That is even slightly smaller screening budget than the BOES method requires to find the global optimum in the GB1 dataset. However, it is not

wise to draw hard conclusions solely from the performance of the method in a single run from the wild-type protein, even though this conformation of the experiment corresponds most closely to the real application of MLDE. More conclusive results are provided by the test of robustness which are discussed in the next section.

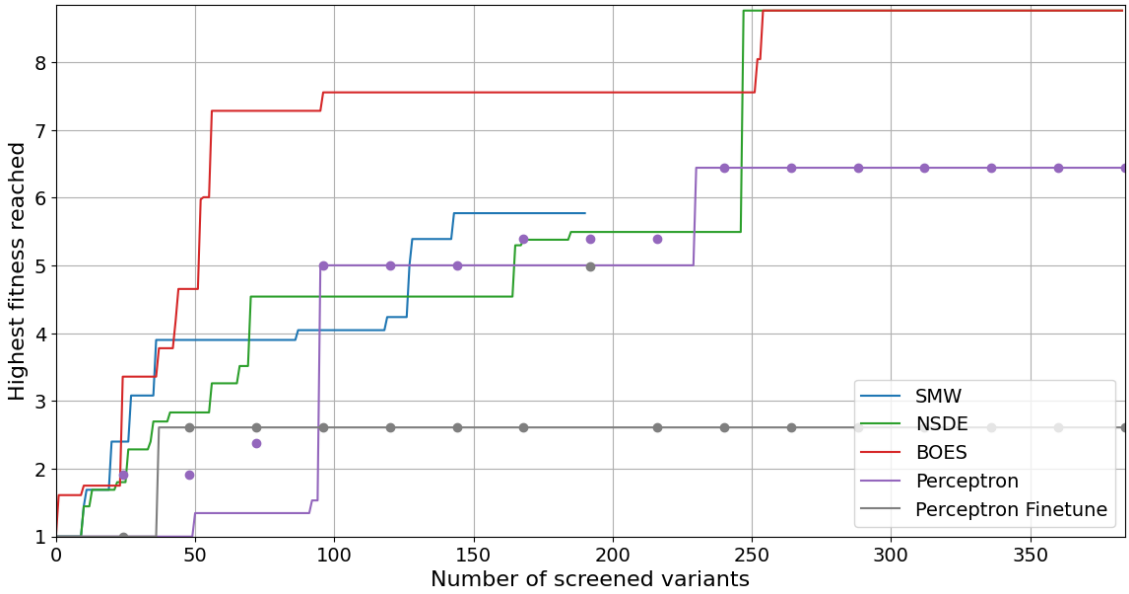
The best-so-far fitness progression produced by the perceptron-training MLDE algorithm shows a few sudden, large increases in the reported fitness. With small screening budget sizes lower than 100 variants, the perceptron training algorithm shows inferior results to the other proposed MLDE methods and the SMW benchmark. With an increased screening budget size between 100 and 250 variants, the perceptron-training method performs on par with the NSDE method, but then it falls short of the other proposed methods again in terms of locating the global optimum. The fitness curve itself represents the performance of the algorithm without exploiting the trained fitness predictor further to suggest extra variants for screening. In this form, the algorithm uses the trained fitness predictor to select diverse protein variants which helps with efficient use of the screening budget. The data points which incorporate an additional screening budget dedicated to predicted variants represent the full version of the algorithm which, in theory, corresponds better to the objective of DE. However, the results of the additional screening of predicted variants at the end of the training algorithm suggest that the extra screening budget is better spent by continuing in the predictor-informed diversity sampling than by screening a large amount of the variants with high predicted fitness.

4.2.2 Performance on PhoQ Wild-type Protein

All evaluated methods, including the SMW benchmark, performed worse on the PhoQ dataset than on the GB1 dataset, relative to the maximum fitness in the fitness landscape. However, some methods performed similarly or better relative to the fitness of the PhoQ wild-type protein, which is 3.29. When looking at the results in figure 6b, it is important to remember that the global optimum of the PhoQ dataset is 133.59, whereas the figure is scaled to fitness range from 0 to 70 for better clarity. This means that even the best-performing BOES method only achieves approximately half of the globally optimal fitness in the PhoQ dataset, while both the BOES and the NSDE methods managed to identify the best variant in the GB1 dataset. The decreased performance consistent across the evaluated methods coincides with the expectation that the PhoQ dataset is more challenging of the two, because of the higher concentration of non-functional variants in the sequence space. However, both the BOES method and the SMW benchmark achieved much better results on the PhoQ dataset relative to the wild-type protein's fitness, yielding an almost 20-times and over 14-times improvement in fitness, respectively. This comparison might not be conclusive to the difficulty of searching the two datasets, because the best variant in the GB1 sequence space has only 8.76 times higher fitness than the GB1 wild-type protein, so such results are infeasible. Nevertheless, it is important to realize that the comparison between two datasets, where the so-called fitness resembles completely different properties of a completely different protein is difficult and even though the best-performing method,



(a) Performance on sampled starting variants from GB1 dataset.



(b) Performance on GB1 wild type protein: The scattered points report the top fitness among the training variants specified by the horizontal axis plus 100 additionally screened variants suggested by the trained perceptron model.

Figure 5: Best-so-far fitness progressions of proposed methods on GB1 dataset.

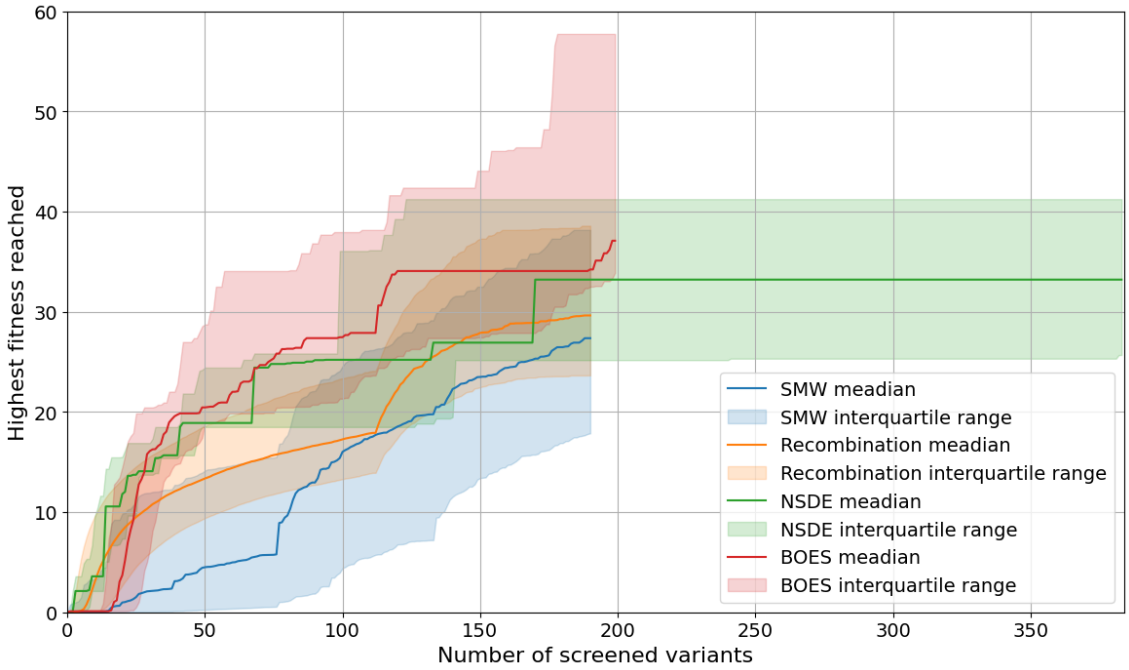
BOES, *only* achieved half of the maximal fitness, all of the methods still produced variants with multiple-times the functionality of the original, wild-type protein.

Same as when evaluated on the GB1 dataset, the performance of BOES is comparable to the SMW benchmark with smaller screening budget sizes and surpasses the benchmark with a larger amount of screened variants. However, the benchmark is surpassed by the BOES method much later, i.e. with a much larger size of the screening budget, than in the GB1 dataset. Interestingly, the other two proposed methods, NSDE, and the fitness predictor, do not show significantly improved obtained fitness over the SMW benchmark. On the contrary, at approximately 100 screened variants, the SMW benchmark locates a variant with higher fitness than the two methods manage to locate in the entire run of the MLDE procedure.

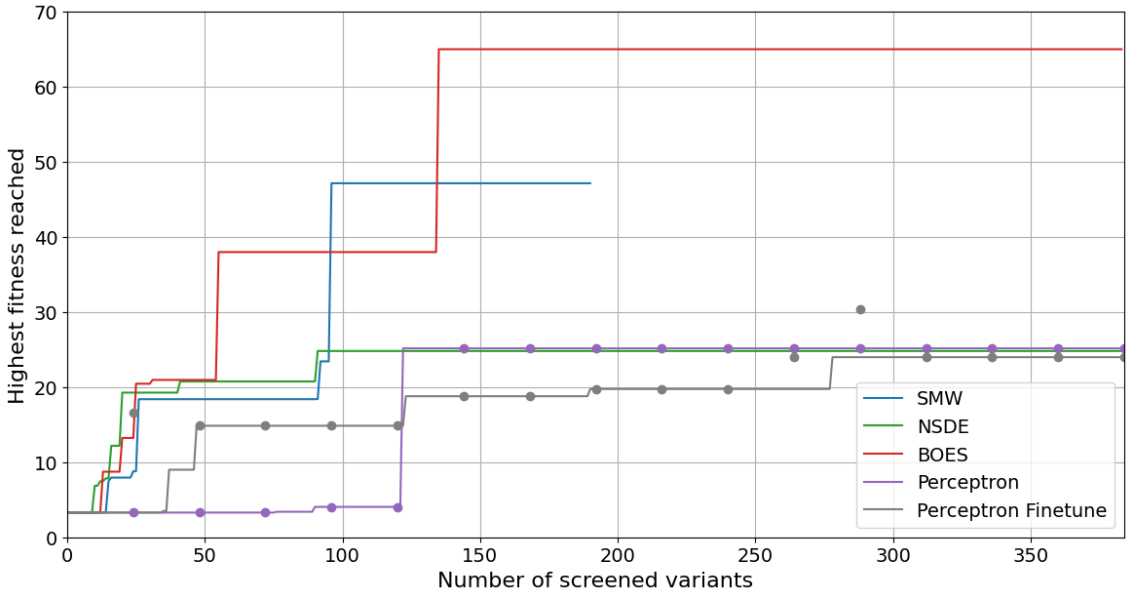
Once again, it is important to note, that this figure is included mainly because it simulates the real-world application, but the evaluation of the methods by the means of a single run of the MLDE procedure is not very credible from a statistical standpoint in the sense that the results cannot be generalized to application to a new, unknown protein with a good conscience. This is especially true in the PhoQ dataset, where the density of functional variants is very low and the identification of a single functional variant can mean a significant difference in the result of the DE procedure, as evident from figure 6b. However, the outlying results of the evaluated methods obtained through this measurement, which will be made evident by comparison to figure 6a, is an important reminder of the strong effect of a fortunate or rather unfortunate position of the wild type variant in the sequence space on the result of any DE procedure.

4.3 Performance on Sampled Starting Variants

The first quartile, median, and third quartile values from among all of the runs conducted with both of the benchmarks and the NSDE and BOES methods are reported in figures 5a and 6a for the GB1 dataset and PhoQ dataset, respectively. The vertical axis relates to the aforementioned fitness values and the horizontal axis stands for the number of variants screened to achieve those results. Same as in the graphs of DE runs from the wild-type protein, the sampled starting variants are plotted as 0th variants on the horizontal axis so that the number of screened variants always corresponds to the number of variants selected by the method, i.e. the experimental screening budget in a real DE procedure. In contrast to figures 5b and 6b showing the methods' performance on the wild-type proteins, the vertical axis starts at 0 rather than the non-zero fitness of the wild-type protein, because when running the algorithms from randomly sampled starting variants, most of the initial variants have a fitness value close to zero. For each method, the median achieved fitness value is plotted as a full, colored line and the interquartile range of the achieved fitness is filled semi-transparently with the same color.



(a) Performance on sampled starting variants from PhoQ dataset.



(b) Performance on PhoQ wild type protein: The scattered points report the top fitness among the training variants specified by the horizontal axis plus 100 additionally screened variants suggested by the trained perceptron model.

Figure 6: Best-so-far fitness progressions of proposed methods on PhoQ dataset.

Different methods and benchmarks were repeated different number of times to obtain the sampled results depending on the given method's computational difficulty. Both of the DE benchmarks, SMW and Recombination, were repeated 160,000 times. The SMW benchmark was run once from each of the 160,000 possible starting variants, whereas the Recombination benchmark, which does not depend on a starting variant but is highly non-deterministic, was simply repeated 160,000 times in the same setting. The three proposed MLDE methods, BOES, NSDE and the perceptron-training algorithm, were each run 200 times from 200 different starting variants. The same sample of starting variants was used for each of the evaluated methods. The perceptron-training method with the addition of finetuning of the PPLM is not included in this evaluation at all, because the fine-tuning process is too computationally demanding for the algorithm to be re-run a reasonable number of times. The specified number of runs for each benchmark and method was carried out two times, once for each of the used datasets.

The results of the perceptron-training algorithm run from sampled starting variants are included separately in figures 7 and 8 for the GB1 and PhoQ datasets, respectively, to avoid making the figure incomprehensible because the results also include additional about the prediction performance of the trained perceptron. The SMW benchmark is plotted in these figures again as a means of comparison of the perceptron method to other methods.

4.3.1 Sampled Performance of SMW Benchmark

For the SMW benchmark, the comparison between the runs from GB1 wild-type protein in figure 5b and the run from sampled GB1 variants in figure 5a shows a massive decrease in the median fitness value over the performance on the wild-type protein in the first 50 screened variants. The SMW benchmark, plotted in blue, does not manage to find any functional variants in any runs in the interquartile range for almost the entire first 50 screened variants. This is a stark contrast to the result obtained from the wild-type protein, where the SMW benchmark was able to identify a variant with fitness of 3.90, i.e. almost four times better than the wild-type protein. With increasing budget size, the median fitness value is consistently lower than the fitness of the wild-type protein run, but slowly approaches the wild-type protein curve. Only the final results at 191 screened variants are comparable at 5.77 fitness reached from the GB1 wild-type protein and 5.49 median fitness on the GB1 dataset.

The results on the PhoQ dataset, plotted in figure 6, confirm the decrease in median fitness relative to the best-so-far fitness starting from the wild-type protein in the first 50 to 70 variants screened, where the median obtained fitness does not grow substantially. At 70 variants screened the median fitness shows a more promising increase, comparable to the result obtained on the wild-type protein, but the final obtained median fitness at 191 conducted screens of 27.37 is significantly lower than the final fitness obtained in run from the wild-type protein, which is 47.12.

Overall, the results from both datasets coincide in a very strong effect of the quality of the starting protein on the resulting fitness in the first 70 screened variants. The results differ between the datasets in the final fitness after the full procedure consisting of 191 conducted screens, where the evaluation on the GB1 dataset shows very little effect, but the evaluation on PhoQ datasets shows a very strong negative effect of non-functional starting protein over the functional, wild type protein on the final result of the DE procedure.

4.3.2 Sampled Performance of Recombination Benchmark

Unlike the SMW benchmark, the median fitness obtained by the recombination benchmark increases steadily even with small sizes of the screening budget, resulting in much better median results with less than 100 screened variants and still significantly better median results with a screening budget of up to 150 variants.

The median and quartile curves produced by the recombination benchmark contain a single, sharp increase in fitness after 112 screened variants. This graph feature is caused by the two steps of the recombination algorithm, where 112 variants are randomly sampled and screened, and than the top 3 highest-fitness variants are recombined into 78 mutants, as described in section 3.2.2.

Another interesting observation is the very smooth shape of the median and quartile fitness values reported for the recombination benchmark in comparison to all of the other evaluated methods. The median and quartile curves of the other methods show many sharp, stair-like increases and flat plateaus of varying length. These features probably relate to dominant local maxima specific to the protein's fitness landscape. For example, between the screening budget size of 100 to 200 variants, the median curve of the BOES method shows only one, sharp increase in fitness, while the interquartile range stays exactly the same. This feature most probably relates to a local maximum of 7.55, which the method identifies consistently in 100 screens, and the global optimum of 8.76 which is much harder to finally locate for the method. Both steps of the recombination method individually screen variants in an uninformed manner where the previously obtained results play no role in the choice of screened variants. In the first step, the method performs random sampling and in the second step, the recombined mutants are screened exhaustively. Because of this, neither of the steps, individually, is affected by the characteristics of the fitness landscape such as relative positions of local maxima.

4.3.3 Sampled Performance of NSDE Method

On the GB1 dataset, in figure 5, the NSDE method shows a slight decrease in the median fitness for smaller sizes of the screening budget relative to the obtained fitness starting from the wild-type protein. However, the decrease in performance relative to the run from the wild-type protein is much less pronounced than in the SMW benchmark. While the median fitness of the SMW method stagnates for almost the first 50 conducted screens, the shape of the median fitness progression curve of the NSDE method looks very similar

to its performance on the wild-type protein. The median curve only starts close to zero fitness, because the median fitness of a randomly sampled variant is close to zero, whereas the wild-type curve starts at the non-zero fitness corresponding to the wild-type protein's fitness. This provides the wild-type run with an initial advantage, which dissipates in approximately 125 screened variants. Moreover, the performance with smaller screening budgets is not present in results on the PhoQ dataset in figure 6. A conclusion can be made, that a functional starting protein can boost the performance of NSDE in the initial iterations but the disadvantage of a non-functional starting variant is effectively negated by the NSDE method within a reasonably sized screening budget.

A slightly decreased median performance, as opposed to the performance on the wild-type protein, can also be observed in terms of finding the global maximum of the GB1 fitness landscape. While starting from the wild-type protein, the NSDE method finds the globally optimal variant in a relatively small amount of conducted screening, the median fitness progression shows that when considering non-functional starting variants with close-to-zero fitness, the method rarely finds the optimal variant in under 384 conducted screens, with the median final fitness value being 7.55, still reasonably close to the 8.76 global optimum. On the PhoQ dataset, however, the NSDE method reports significantly increased median performance in terms of the final achieved fitness as opposed to the run from the PhoQ wild-type protein. It can be concluded, that with a reasonably sized screening budget, the fitness of the starting protein does not have a strong effect on the final obtained fitness.

4.3.4 Sampled Performance of BOES Method

Unlike the SMW benchmark and the proposed NSDE method, the performance of the BOES method actually improves when the starting variant is randomly sampled. Comparison on the GB1 dataset in figures 5a and 5b shows that the median curve achieves the same fitness in the first few screened variants and surpasses the wild type curve with increasing screening budget, reaching a fitness of 6.88 with a screening budget of 50 screened variants, whereas the wild type run only reaches best-so-far fitness 4.65 with the same budget. In terms of the global maximum, the median curve and the wild-type run reach it in under 200 screened variants and just over 250 screened variants, respectively, marking the median fitness achieved by runs from randomly sampled starting variants superior once again. The more robust comparison of the methods run from sampled variants shows a very clear superiority of the BOES method with an arbitrary size of the screening budget.

Comparison of the wild type and median fitness curve on the PhoQ dataset in figures 6b and 6a, respectively, does not fully coincide with the observation made on the GB1 dataset. The BO-based method does not show an increase in performance with the randomly sampled starting variants over the wild-type protein. However, the comparison does show only a slight decrease in performance when starting from the non-functional variants. A more pronounced difference is present with larger screening budget sizes of around 140

screened variants and more, where the wild-type run reaches a fitness of 64.95, while the upper quartile fitness value only approaches values close to 60 in around 200 screened variants.

Another noteworthy observation from figures 5a and 6a is that the interquartile range of the BO-based method is much smaller than most other methods. It can be concluded that the method performs well, and potentially even better, with a randomly sampled, often non-functional starting variant, and is very robust to different positions of the starting variant in the sequence space. The interquartile range in the PhoQ dataset increases with larger sizes of the screening budget but is also extremely small in the very efficient, first 50 variants screened.

The interquartile range is significantly increased after the local optimum of 7.55 is reached in the GB1 dataset around the 100 screened variants and gradually increases after the first 50 screened variants in the PhoQ dataset. The sudden increase in the GB1 dataset can be attributed to the variance in number of conducted screens it takes the individual runs to locate the global optimum, increasing the best-so-far fitness by a considerable amount as seen on both the median curve in figure 5a and the wild type run progression in figure 5b. The stair-like quality of the upper quartile range reported on the PhoQ dataset, plotted in figure 6a, suggests that the increased interquartile range may also be caused, at least partly, by *jumps* in best-so-far fitness tied to locating of a significant local optimum. The method seems to perform exceptionally well, regardless of the nature of the fitness landscape, during the initial mapping of the sequence space with a small screening budget. After that, the difficulty of localization of different local optima is highly affected by the shape of the fitness landscape and both the performance and the robustness to the position of the starting variant in the sequence space may vary based on the mutated protein. However, the BO-based shows superior results to the other evaluated methods in both datasets.

Lastly, a practical comment on the number of screened variants is in place. The sampled runs of the BOES procedure were ended at 200 screened variants to maintain a manageable computational time. Inference of a GP model with n data points has time complexity $\mathcal{O}(n^3)$ [4, 94] and the model has to be fitted in each iteration of BOES while the dataset increases by one data point between iterations. This results in vastly slower iterations at the end of the procedure compared to the initial iterations. In practice, this is not a limiting factor for BOES since the screening experiments are exceptionally expensive and time-consuming and the number of screened variants is usually only in the order of tens or hundreds. Because of this, the run time of an iteration of BOES will never be an issue in real DE applications. The time complexity is, however, helpful to note for the purpose of evaluation of the BOES method in repeated runs of the procedure.

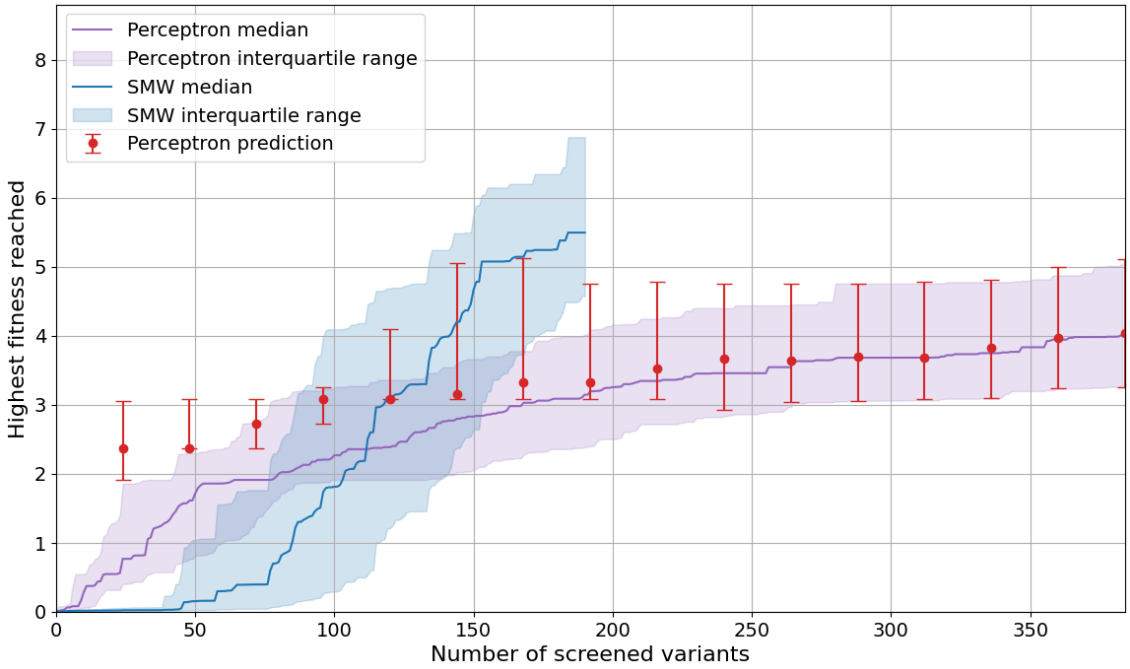


Figure 7: Performance of the proposed perceptron-training method on sampled starting variants from GB1 dataset: *Perceptron prediction* represents median and interquartile range of the top fitness among the training variants specified by the horizontal axis plus 100 additionally screened variants suggested by the trained perceptron model.

4.3.5 Sampled Performance of Perceptron Training

Results of the runs of the perceptron-training algorithm with sampled starting variants are reported in figure 7 and figure 8 for the GB1 and PhoQ datasets, respectively. The median fitness values and interquartile ranges plotted in purple represent the performance of the training algorithm before an additional screening budget is used to screen variants with high predicted fitness. The training algorithm itself exploits the trained perceptron to identify diverse variants for screening, exploring the fitness landscape in a more efficient manner, even before exploiting the perceptron to suggest variants with the highest predicted fitness. The performance of the training algorithm itself on both datasets shows that with small budget sizes, this form of exploration of the fitness landscape outperforms the SMW benchmark. For small budget sizes, the median and quartile curves are similar to the results obtained from the recombination benchmark in the first sampling step of the algorithm. This observation is sensible since both methods perform some sort of diverse sampling in the sequence space.

The performance of the complete method with the additional screening of variants which are predicted to have high fitness is reported in figure 7 and figure 8 in the form of red interquartile ranges in intervals of 24 screened variants. The prediction performance is

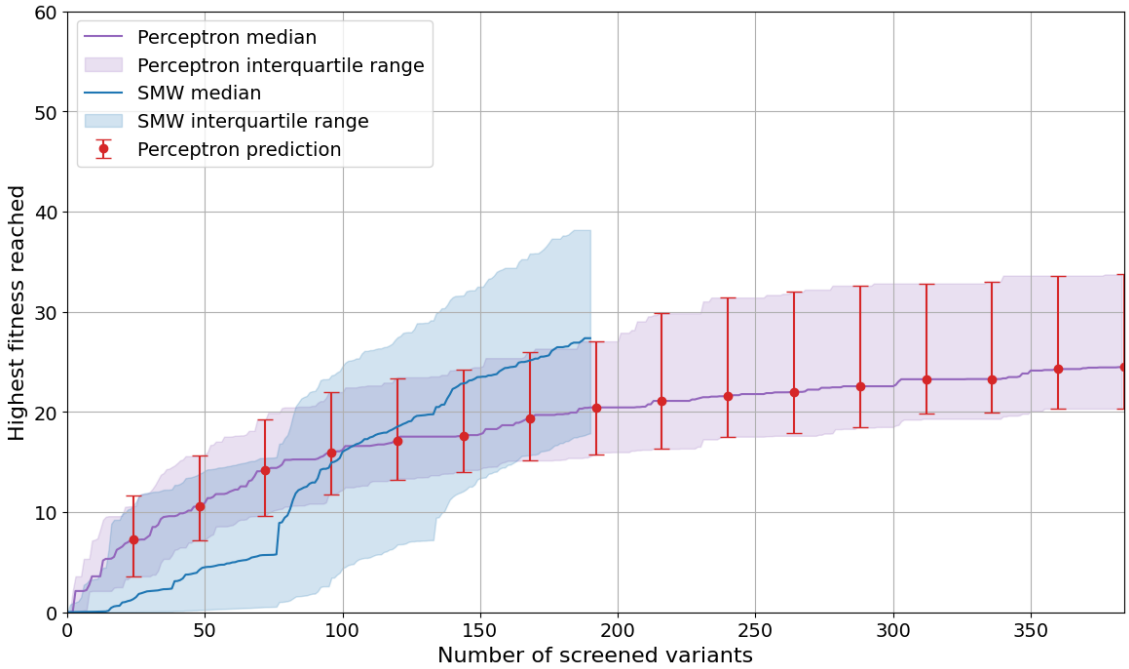


Figure 8: Performance of the proposed perceptron-training method on sampled starting variants from PhoQ dataset: *Perceptron prediction* represents median and interquartile range of the top fitness among the training variants specified by the horizontal axis plus 100 additionally screened variants suggested by the trained perceptron model.

recorded in each run as the maximum true fitness from among the variants screened for training plus 100 variants with the highest predicted fitness. The median and quartile values across the conducted runs are reported. The maximum fitness is recorded after each round of training and 24 variants are screened between the rounds, hence the interval of 24 variants. In the GB1 dataset, the addition of the prediction step improves the results significantly for small screening budget sizes of up to 150 or 200 variants. For sizes of the screening budget larger than 275, the prediction step is rarely able to identify any new variant with improved fitness over the already sampled variants. When exploring the predictions of the perceptron trained with different-sized training samples, it was evident, that the best predicted variant slowly improved in subsequent iterations of the algorithm until approximately the 168-sized training set, after which the predictive ability of the perceptron usually started to decrease. It seems that in the repeated rounds of training, the predictor was overfitted and the predictive qualities collapsed. With a more optimal setting of the training procedure and its hyperparameters, the results of this method with larger screening budgets could be further improved.

In the PhoQ dataset, the prediction step rarely locates improving variants with any size of the screening budget used for training. This is indicated by the manner in which the

plotted ranges of predictive performance follow the results of the sole training algorithm almost perfectly in figure 8. It is possible, that in the more challenging PhoQ dataset, the requirements on a functional variant are more restrictive. As a consequence, more training data are required to train the predictor successfully. In that case, however, the predictor would have never managed to be trained successfully because a sub-optimal choice of the training hyperparameters causes the predictor to collapse in prolonged training, as the results from the GB1 dataset suggest. Another hypothesis is that the requirements on a functional variant are so complicated in the PhoQ dataset, that the proposed model is unable to learn them and predict high fitness variants regardless of the size of the training dataset.

4.4 Final Fitness Distributions

In addition to the median and interquartile fitness curves, the results of the evaluation of proposed methods on sampled starting variants are also reported as final fitness distributions in figure 9 for the GB1 dataset and figure 10 for the PhoQ dataset. Each figure contains four violin plots which report the distributions of the highest obtained fitness by different methods with a screening budget of 50, 100, 150, and 190 variants. These figures do not depict the gradual increase in the best-so-far fitness as well as the median figures 5a and 6a, but they contain much more detailed information about the distribution of the final results than a simple interquartile range can provide. The violin plots are able to report on possible local maxima, which certain methods tend to identify, the entire range of final results, which one can expect to obtain through a given method, and the frequency of outlying results with far higher or lower fitness than the interquartile range might suggest.

4.4.1 Distribution of SMW Results

The results of the SMW benchmark at 50 screened variants in both datasets confirm the observation that the SMW method almost never yields results with a small screening budget. However, the result distributions with subsequent screening budgets of 100, 150, and 190 in the GB1 dataset, plotted in figure 9, provide an interesting insight, which was not obvious from the interquartile fitness progressions. All three budget sizes show that while the distributions have very heavy parts around the median mark, which influence the interquartile range to be relatively small, there are also smaller, but noticeable parts of the distributions outside of the interquartile range plotted in figure 5a. Specifically, the distribution at 100 screened variants shows a light, but long tail into higher fitness values, ending almost at a fitness value of 6, whereas the upper quartile fitness value at 100 screened variants is just over 3. Similarly, the distributions at 150 and 190 screened variants show noticeable peaks above the upper quartile. On the other hand, the distribution at 150 screened variants also has a strong tail leading down to 0 fitness and all four of the distributions show a number of runs which ended with almost zero fitness at all sizes of the screening budget.

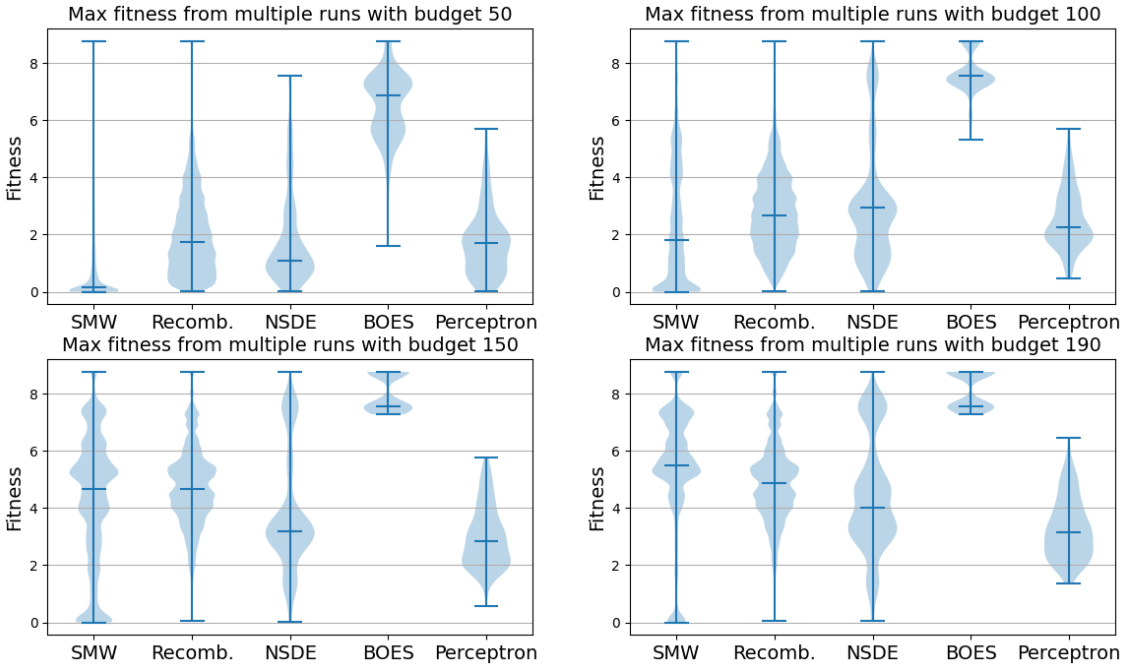


Figure 9: Results of proposed methods on sampled starting variants form GB1 dataset.

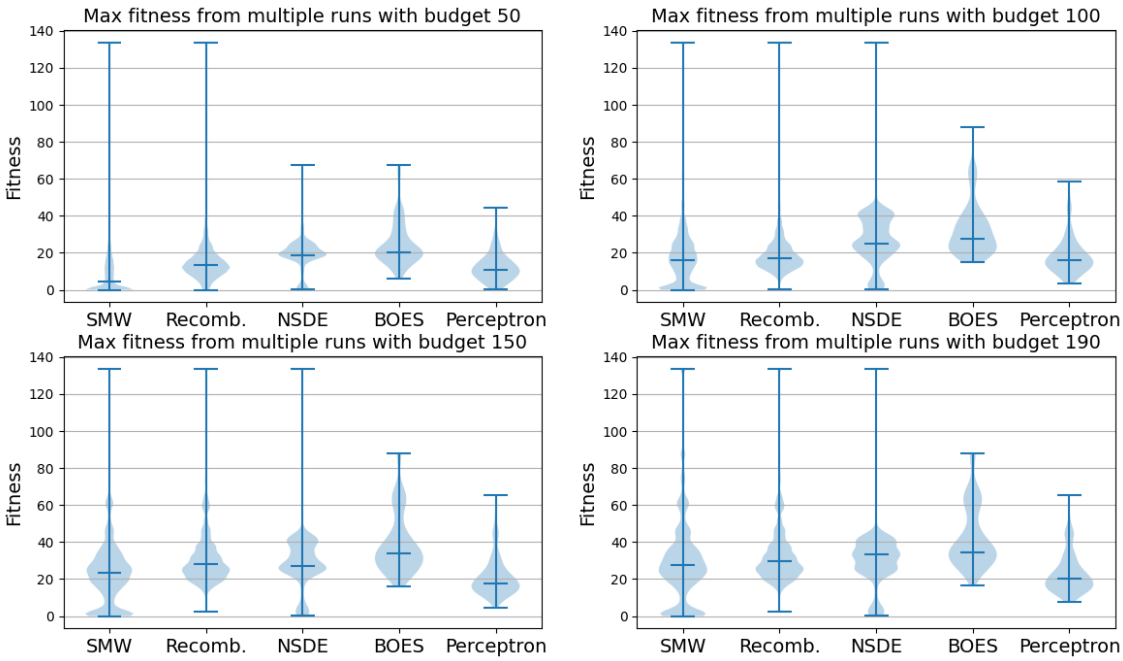


Figure 10: Results of proposed methods on sampled starting variants form PhoQ dataset.

The distributions of SMW results on the Phoq dataset, plotted in figure 10, are less surprising. Apart from confirming the non-feasibility of the method at a screening budget of 50 variants, the distributions also confirm the presence of a large number of runs that end with almost zero fitness with any size of the screening budget. As far as any positive outlying results go, only very small peaks with fitness over the border suggested by the upper quartile are present with the larger screening budgets of 150 and 190 variants. Apart from these very small improving peaks and the large number of non-functional results in all screening budget sizes, the distribution is represented by the interquartile range quite well.

In conclusion, the detailed distribution of the obtained results reveal two important properties of the SMW method. First, the method has a significant chance of producing almost no improvement in fitness and yielding a final variant with very low fitness regardless of the screening budget. This property can be attributed to the strictly local means of exploration of the sequence space, which leads to difficulty in identifying any functional variants if the starting protein itself does not have very high fitness. The method has no means of making large *jumps* in the sequence space or sampling completely unrelated variants which can be problematic when the starting protein is situated in a part of the fitness landscape populated by other low-function variants. Secondly, the SMW method has a very large range of results depending on the starting protein variant on the GB1 dataset. The distribution of results at 150 screened variants shows a significant amount of results from almost the entire range of fitness values in the dataset. When a larger screening budget is provided, it might be wise to split the budget into multiple runs of the procedure from unrelated, possibly sampled, starting variants rather than restarting the procedure from the final highest fitness variant after the 190 variants are screened.

4.4.2 Distribution of Recombination Results

Result distributions of the recombination benchmark coincide between the two datasets very closely. Both datasets show no significant peaks outside of the interquartile range and no extremely long tails of the distribution. Although, it should be noted, that on the GB1 dataset, the distribution is relatively wider than on the Phoq dataset and also relatively wider to the distributions of other methods' results on the GB1 dataset. The only interesting observation, that can be made on the plotted distributions, is that with larger screening budget sizes of 150 and 190, the distribution tends to contain small peaks with improved fitness over the upper quartile in both datasets. These peaks are not very strong and do not pose any implications for the employment of the method.

4.4.3 Distribution of NSDE Results

The distribution plots on the GB1 dataset in figure 9 provide an interesting insight into the results of the NSDE method. The smallest reported screening budget size of 50 already shows a very slim tail of results of high fitness well outside the interquartile range plotted in figure 5a. Then, for the larger sizes of the screening budget, an increasingly large

peak develops at the very top of the fitness scale. The median results in figure 5a prove that the NSDE method can yield excellent results in 225 to over 350 screened variants. Furthermore, in the run from the GB1 wild-type protein in figure 5b the method identifies the global optimum in just under 250 screened variants. However, the plotted result distributions show that from a considerable amount of starting variants, even though possibly non-functional, the NSDE method can yield excellent results with a screening budget of 190, 150, or in some cases just 100 screened variants. The result distributions on the PhoQ dataset in figure 10 do not show peaks with as high relative fitness as in the GB1 dataset, but very large peaks with improved fitness are present with screening budget sizes of 100 and 150, which are still well over the upper quartile fitness value reported in figure 6a.

While the very high-fitness results are not common enough to rely on the method in the proposed form to produce them consistently, they show great potential for possible future improvement in the performance of the method. Firstly, an initial sampling step is already implemented in the proposed form of NSDE to improve the robustness of the method to the position of the starting variant in the constructed neighborhood graph. If the initial sampling step was re-invented, the portion of the runs which fall into the high-fitness peak of the distribution could increase, making the method's performance comparable to the dominant BOES method. An example of an alternative method of sampling the initial variants could be employing the proposed BO method or a different active learning approach. Additionally, an improving measure could be implemented in the NSDE method, which would identify that the method is searching in a close area around a single local maximum for many iterations. The measure would nudge the search algorithm to abandon the greedy approach of choosing the highest-scoring variant in the *open* set and choose a variant with greater distance in the sequence space instead. This second improvement would be much trickier to implement in a way, which would not be helpful in certain runs but decrease the method's performance more in other cases.

4.4.4 Distribution of BOES Results

Results of the BOES method on the GB1 dataset plotted in the violin graphs in figure 9 confirm once again its superior performance among the other evaluated methods. Even with a relatively small screening budget of 50 variants, the method almost always manages to identify a variant with fitness 5 or higher and the distribution only moves upward and its tails shorten as the screening budget increases. Additionally, with a screening budget of 150 variants or more, the BOES method is perfectly consistent in the sense that not a single run was recorded where the method would underperform and fail to identify a variant with an excellent fitness of 7.55 or higher. Both benchmarks and the NSDE method can, at least in some cases, fail to perform completely, as evident from the final fitness distributions. Even with the smallest plotted size of the screening budget of 50 variants, in the worst recorded, outlying run, the BOES method still managed to find a variant with almost twice the fitness of the GB1 wild-type protein. Evaluation on the PhoQ dataset in figure 10 coincides with the observations made on the GB1 protein. Even with 50 screened

variants, the BOES method identified an improving protein variant in each recorded run and with larger screening budgets of 100 variants or more, even the worst-case results were variants with almost 20 fitness, that is approximately 6 times the fitness of the wild type protein.

An interesting observation, which can also be made on both datasets to some extent, are the additional smaller peaks at higher fitness in the result distributions. In the Phoq dataset, the peak at around 65 fitness is not very noticeable until 150 or 190 screened variants, while in the GB1 dataset, a peak at the fitness of 7.55 is present in all four violin plots and a peak at the maximum fitness appears from 100 screened variants onwards. These peaks can be attributed to dominant local maxima in the fitness landscapes and the four plotted distributions illustrate how the mass of the distribution slowly shifts from lower-fitness peaks to higher-fitness peaks with increasing size of the screening budget.

4.4.5 Distribution of Perceptron Results

For the perceptron-training method, the violin plots only report on the distributions of the results obtained through the main, training part of the algorithm without the additional screening of variants with high predicted fitness. The predictive performance is left out of figures 9 and 10 since the prediction can only be obtained at screening budget sizes of multiples of 24. The predictive performance is already reported sufficiently in the separate graphs of sampled runs of the perceptron-training algorithm in figures 7 and 8. This means that the violin plots of the result distributions correspond to the median and quartile values depicted in figures 7 and 8 in purple.

The distributions of the perceptron-training results do not contain any prominent peaks outside of the interquartile range. The shape of the distributions is similar to the results of the recombination benchmark, albeit for larger screening budgets, the results of the perceptron-training algorithm are a bit lower. However, the perceptron-training procedure still provides a multiple-times improvement over the fitness of the wild-type protein in both datasets. One strength of the method, which is not evident solely from the median and quartile fitness values, is that with larger budget sizes, the method always finds a functional variant of the protein. The only other evaluated method that guarantees a functional resulting variant is the BOES method. The remaining three evaluated methods all sometimes finish the DE procedure with zero or close-to-zero fitness when the position of the starting variant in the sequence space is unfortunate.

4.5 Modelling the Embedding Space

As a secondary result, the fitness landscape modeled by the best-performing BOES method was visualized alongside the initial PCA and t-SNE plots in figures 3 and 4. All four of the plots with predicted fitness (figures 3b, 3d, 4b, 4d) show fitness predicted by a GP model trained on 384 screened variants by the BOES method when initiated with the wild type

protein. These visual results are not presented to draw any hard conclusions. Rather, they serve as an insight into the modeling capabilities of the proposed methodology.

The PCA plots in figures 3b and 4b show that BOES was able to model the large area of high fitness and did not predict high fitness outside of it. Comparison with the original plots with true fitness shows that the modeled high-fitness area is slightly smaller than the true high-fitness area or that the fitness decreases more quickly towards the edges. This suggest that the model is more conservative rather than overly optimistic in its prediction, which is reasonable given the overwhelming representation of non-functional variants in the sequence space. Furthermore, the range of predicted fitness values on the PhoQ dataset in figure 4b is significantly smaller than the range of true fitness values. This is caused by the fact that BOES did not identify any variants with fitness of 70 or more, as can be seen in figure 6b.

The visualization of predicted fitness in the t-SNE plots in figures 3d and 4d provides more interesting insights. Results on both datasets show that BOES was able to identify multiple local clusters of high-fitness variants. Especially the results on the GB1 dataset in figure 3d reveal that almost all of the major clusters were identified. The modeled embedding space of the PhoQ dataset also shows many of the local clusters but with considerably lower fitness than their true peaks. This suggests that many of the areas of high-fitness variants were explored but the the method might have over-emphasized exploration over exploitation and failed to locate the peaks with considerably higher fitness. This observation could prove useful in future fine-tuning of BOES. Exploitation could be emphasized by a change of acquisition function. However, changes based on this observation should be applied cautiously. This result might be highly specific to the PhoQ dataset and any significant change to the balance of exploration and exploitation might prove detrimental to the method's performance on different datasets.

5 Discussion

This section relates the obtained results to the literature. Performance of the implemented benchmarks is compared to their original version in [88] and differences in implementation are discussed. NSDE and BOES methods are compared to state-of-the-art MLDE methods with model regression objective to demonstrate the advantage of an optimization-oriented approach. BOES method is also compared to other BO-based MLDE methods to show the advantage of the innovative application of PPLM-extracted sequence representation in BO. Furthermore, options for future development of the proposed optimization methods are indicated. Lastly, possible reasons for the unideal performance of the perceptron-training algorithm are discussed and solutions for future improvement are proposed.

5.1 Benchmarks

The results obtained with the SMW benchmark after the full run of the procedure at 190 screened variants coincide with results reported in literature [88]. The distribution of the SMW results in figure 9 corresponds perfectly to the reported distribution, which also shows two dominant peaks and a significant number of runs that fail to identify any functional variant. The distribution of results of the recombination benchmark after the full procedure, i.e. at 190 screened variants, has very similar shape to the distribution of results from the literature. Both the obtained results and the consulted literature show practically no runs which end with a non-functional variant and both show a similarly wide distribution with two prominent peaks. However, the distribution in the obtained results is shifted towards lower fitness values relative to the literature. This discrepancy also causes the final results of the two benchmarks to show slightly better performance of the SMW benchmark over recombination while the recombination procedure is reported to produce better results [88].

A probable explanation of the slightly different obtained results is that an upper bound of the number of screened variants was used in the recombination procedure, while an exact number of screened variants was used in the single mutation walk. This implementation resulted in a smaller average number of screened variants in the runs of the recombination procedure. In other words, the single mutation walk simulation was provided with more resources on average making the comparison of the results slightly unfair. The small disadvantage imposed on the recombination procedure could be removed by adding extra steps after the procedure in which additional variants would be screened one by one until the exact number of unique variants was screened. The choice was made not to needlessly complicate the benchmark method. The comparison of the two benchmarks is not the goal of their implementation, rather the two benchmarks serve together as a baseline for evaluation of performance of the proposed MLDE methods.

5.2 Optimization vs Model-fitting Approach

The objective of optimization methods, which is in the context of DE to maximize the fitness function, corresponds directly to the goal of DE. On the other hand, model-fitting methods, like the proposed perceptron-training algorithm, define a tangential goal of minimizing the predictive error of a trained model. These objectives align to a certain extent. Both approaches require some extent of exploration of the initially unknown fitness landscape. However, an essential component of optimization methods is exploitation of already discovered regions of high fitness and a balance between the contradictory nature of exploration and exploitation. In contrast, a model-fitting approach does not necessarily have to emphasize exploitation whatsoever and the trained model has to be exploited after the training is complete to suggest some amount of variants with high predicted fitness, which are also screened, wasting additional resources. Common sense suggests that defining a method with the real objective in mind from the get-go might lead to better use of resources.

The two proposed methods with an optimization objective clearly outperform the proposed perceptron-training method. This result is not conclusive by itself since all three of the methods were implemented in a relatively simple form with limited exploration of hyperparameters. Further optimization of individual steps of the algorithms, such as initial sampling in the NSDE method, and grid-search of optimal hyperparameters, such as the batch size and learning rate, would have to be conducted to draw conclusions based solely on the methods proposed in this thesis. This is especially true, because the perceptron-training method arguably needs the additional fine-tuning of the hyperparameters the most since the results suggest overfitting of the trained predictor and collapse of the predictive performance with large screening budgets.

To address this concern, a comparison with model-fitting methods from the literature is reported in table 1. The table includes results of the implemented SMW benchmark to serve as a baseline and the two optimization-oriented methods, NSDE and BOES, compared to results of several state-of-the-art model-fitting DE methods reported in [31], which all train a fitness predictor during the DE procedure. A concise description of the included methods is adapted from [31]:

- MLDE [88] trains an ensemble of shallow neural networks as fitness predictors on randomly sampled variants.
- ftMLDE, focused training MLDE [9], is a strategy for running MLDE with training sets designed to avoid holes. The comparison includes ftMLDE with two sampling strategies, EVmutation [95] and MSA-transformer [34].
- CLADE [96] trains a fitness predictor with high-fitness mutants obtained through a hierarchical clustering sampling method.
- CLADE 2.0 [97] selects the high-fitness mutants with a scoring function which employs an ensemble of methods including a PPLM.
- AFP-DE [31] uses a PPLM to sample variants and extract sequence embeddings. Iteratively trains a fitness predictor with the sampled variants and finetunes the PPLM with variants with high predicted fitness.

The model-fitting methods are tested with a screening budget of 80 variants with two different splits between a part of the budget used for training and the rest of the budget left to screen variants with top-predicted fitness. The first split is 24 training variants and 56 predicted variants and the other is 48 training variants with 32 variants left to exploit the prediction. SMW and the optimization methods do not split the resources, so the table contains just a single result for these methods.

With a screening budget of 80 variants, the comparison with state-of-the-art model-fitting methods shows a clear dominance of the proposed BOES method, while the NSDE method shows similar results to most of the other methods and is outperformed by BOES

Dataset	GB1	PhoQ
Screening budget	(24 + 56) | (48 + 32)	(24 + 56) | (48 + 32)
SMW	0.445	0.138
MLDE	0.448 | 0.506	0.049 | 0.099
ftMLDE (EVmutation)	0.569 | 0.602	0.165 | 0.065
ftMLDE (Transformer)	0.568 | 0.606	0.133 | 0.196
CLADE	0.557 | 0.447	0.161 | 0.192
CLADE 2.0	0.498 | 0.686	0.183 | 0.161
AFP-DE	0.708 | 0.708	0.187 | 0.211
NSDE	0.518	0.155
BOES	**0.831**	**0.284**

Table 1: Comparison of proposed optimization methods with state-of-the-art model-regression methods: Maximum fitness obtained when run from the wild-type protein with a screening budget of 80 variants is reported. The model-regression methods split the screening budget between training and consecutive screening of promising predicted variants (24 + 56 or 48 + 32). The optimization methods screen all 80 variants during the optimization procedure.

and AFP-DE. A comparison in terms of localization of the globally optimal variant in the GB1 dataset highlights the advantage of optimization-oriented methods even more, as both BOES and NSDE find the global maximum in approximately 250 screened variants when started from the GB1 wild-type protein, while the model-fitting AFP-DE method, which also employs a PPLM, only finds a variant with 96.7 % of the maximum fitness with 480 conducted screening experiments (384 in training and 96 to screen predicted variants). That is roughly double the experimental burden. Furthermore, the median fitness curve of the BOES method suggests that it can often find the globally optimal variant in under 200 screened variants even when starting from different, non-functional variants from the GB1 sequence space. The model-fitting methods have the advantage of providing a fitness predictor at the end of the procedure, which can be useful in other applications. In the context of DE, however, the objective to obtain maximum fitness with minimum conducted screening experiments is best fulfilled by the BOES method.

5.3 Bayesian Optimization in Embedding Space

Results from the runs starting from the wild-type protein and the sampled starting variants conducted on both datasets confirm the superior performance of BOES among the evaluated methods. Its strength is especially highlighted by the fitness progression from runs with sampled starting variants on the GB1 dataset in figure 5a and the distribution of obtained results from the same experiment plotted in figure 9. Results in table 1 also show that BOES outperforms state-of-the-art model-fitting algorithms. To evaluate the effect of the informative PPLM embedding and the innovative kernel function used to enable the employment of BO on the high-dimensional embedding, a comparison to BO-based DE methods defined on different input spaces is in place.

GB1 dataset, 191 variants	Avg maximum fitness
SMW	5.35 ± 2.14
GP+EI	7.28
GP+EI+GREMLIN	6.76
GP+EI+HMM	6.74
BOES	**8.14** $\pm$ 0.62

Table 2: Comparison of the proposed BOES method with BO conducted in the original protein sequence space: Mean of the maximum obtained fitness from multiple runs is reported. Results of the implemented methods are accompanied by standard deviation.

To confirm a positive effect of the informative input space on the performance of the BOES method, we first compare the results to a method proposed in [69] which, in its most simple form, performs BO with a GP model and EI acquisition function directly in the protein sequence space. The distance between two variants is computed simply from one-hot encoding of the amino acids at mutated positions. This approach corresponds to the proposed BOES method in terms of the selected model and acquisition function and uses a straightforward definition of the input space and kernel function in place of the embedding space, making it an ideal candidate for evaluation of the effect of the PPLM on the method's performance. The average results of the SMW benchmark, BOES method, and three versions of the method described in [69] are included in table 2. Abbreviation *GP+EI* corresponds to the aforementioned simple version of the proposed method and the other two reported versions introduce two distinct protein fold family regularization techniques at the expense of performance in terms of fitness. It is important to note that in all forms of the method from [69], the model is initially trained on 20 randomly selected variants before the first iteration of BO, which are not counted towards the screening budget, skewing the comparison in their favor. One last note-worthy distinction between BOES and the methods reported in [69] is that, unlike BOES, the methods select new variants for screening in batches of 19.

Comparison in table 2 decidedly confirms a positive effect of employing BO in the embedding space over the original sequence space. The results show that BOES outperforms all three versions of the method with a one-hot encoding-based kernel function. A conclusion about the strength of the boost in performance provided by the informative embedding space would not hold much weight if based on this comparison only, since the method defined [69] has an advantage of 20 randomly screened variants. To measure the effect quantitatively, it would be ideal to implement a new BO-based method with a kernel function defined on the raw sequence space and other parts of the procedure identical to BOES to control as many variables in the comparison as possible.

5.4 Other Informative Input Spaces in BO of Proteins

The conducted comparison to a BO-based method defined on the original sequence space proves the positive effect of the innovative input space qualitatively. Howevr, a comparison

GB1 dataset	Avg maximum fitness	Screening budget
NaiveBO + GP	6.40 ± 0.79	40 + 50
TuRBO + GP	$\mathbf{6.57} \pm 1.02$	40 + 50
BOES	6.47 ± 1.15	**50**

Table 3: Comparison of the proposed BOES method to BO conducted with a different informative sequence representation: Mean of the maximum obtained fitness from multiple runs is reported with the standard deviation.

to another state-of-the-art BO-based method with a different, yet also informative, input space can help assess the performance of the proposed method quantitatively. The ODBO framework [63] employs BO for PE in combination with a novel encoding of amino acids based on the fitness of observed variants with the specific amino acids at the specified mutation position, as described in section 2.6.2. In table 3, results of the BOES method with a screening budget of 50 variants are compared to a classical BO procedure with a GP model and the positional amino-acid encoding of variants and to a trust region BO procedure (TuRBO) with the same model and encoding.

A fundamental difference between the two novel methods of sequence-space representation for BO is that the positional amino-acid encoding proposed in [63] requires an initial dataset of screened variants in which each amino acid appears at each mutation site at least two times, while the PPLM-extracted embedding space requires no screened variants for its construction. The original paper presents a solution to this obstacle in the form of an initial sampling algorithm, which for the GB1 dataset constructs an initial set of 40 variants. This means that while each of the BO procedures compared in table 3 is provided with a screening budget of 50 variants, the construction of the encoding that precedes the two procedures from [63] requires an additional 40 screening experiments, which the BOES method saves in comparison.

Results in table 3 reveal that all three compared BO-based methods are extremely similar in performance with BOES outperforming the other classical BO method, labeled NaiveBO, and the trust region variant slightly outperforming BOES. It should be noted that a trust region variant of BOES could also be implemented, which would most probably improve the original BOES method's performance. Similarly, the authors of the compared BO method [63] propose two additional improvements: prescreening outlier detection via XGBOD [85] and employing a BO procedure robust to outliers [98]. The variant of the authors' method with these improvements outperforms BOES, but the improvements can also be combined with BOES. Adding the prescreening outlier detection step requires a set of already screened variants. To circumvent this, the outlier detection could be enabled after a certain number of BOES iterations. Additionally, the XGBOD method could be replaced with an unsupervised outlier detection method in the initial iterations of BOES. A version of BOES with these two improvements can be expected to yield similar results to the full version of ODBO [63] while saving screening costs on the construction of sequence

representation, based on the comparison of the un-improved versions of the methods.

5.5 Lowering Search Effort of NSDE

Results in figures 5a and 6a show that the NSDE method can achieve great fitness comparable to the proposed BOES method. However, the best-so-far fitness obtained by NSDE increases slower in regards to the number of screened variants. Furthermore, comparison to state-of-the-art model-regression methods for MLDE in table 1 shows comparable results of NSDE to most of the methods, with the exception of AFP-DE [31], even in its current form. Lowering the search effort of NSDE while maintaining great final results could result in performance competitive to the BOES method.

As a graph-search approach, NSDE could benefit from standard methods to lower the search effort. A simple standard method is a common heuristic algorithm for graph exploration named beam search. It implements a pruned version of breadth-first search which can be used in applications, where an exhaustive graph search is infeasible [99]. In NSDE, beam search would be implemented by discarding all but the k (e.g. 2 or 3) top-scoring variants from the open set after each round of screening. Then, neighborhoods of all k variants would be screened in the next iteration. This way, the size of the open set would always be k. This approach could help the method traverse the sequence space more quickly by limiting the number of variants from an area around a single local maximum that would be kept in the open list. Beam search and its variants have been widely used in NLP for decades to approximate the exact search of the full output space[99, 100, 101]. This makes beam search an ideal candidate for experimentation since the nodes of the graph searched in NSDE represent sequence embeddings extracted from a PPLM, marking a similar application.

A more innovative variant of NSDE could employ the proposed BOES method for the initial sampling of variants instead of the currently employed *greedy sampling on the inputs* [62]. The uncertainty sampling conducted by BO guarantees that a good portion of the screening budget will be used to sample variants evenly from the entire sequence space. This makes BO a sensible option for an initial, diverse sampling algorithm. Furthermore, the results in figures 6a and especially 5a suggest that the BOES method can obtain variants of very high quality in a small number of screened variants. Additionally, the results of NSDE in figure 5 suggest that the presence of a high-fitness variant in the set of starting variants could help NSDE in identifying a top-quality variant much faster. This notion is supported by the large interquartile range of the best-so-far fitness when NSDE is started from different variants, plotted in figure 5a. The fact that NSDE locates the global optimum in under 250 screened variants when started from the functional wild-type protein, as shown in figure 5b, also supports the hypothesis. A screening budget of 40 to 50 variants dedicated to initial sampling through BOES could be a good starting point for future experimentation.

Lastly, the PPLM employed as an embedding extractor during the construction of the neighborhood graph could also be exploited to lower the search effort. Instead of screening all neighbors of the variant with the highest fitness in the open set, we could screen the neighbors one by one in the order defined by the pseudolikelihood assigned to the neighbors by the PPLM. As soon as a variant with higher fitness would be screened, the rest of the un-screened neighbors could be ignored and the search would continue from the new, higher fitness variant. Alternatively, the open set of screened variants could be replaced with an open set of the un-screened neighbors of all screened variants. Variants from the open set would be screened in the order defined by a score combining the pseudolikelihood assigned by the PPLM and the true, screened fitness of their highest-scoring screened neighbor. This approach would strike a balance between the verified information obtained through the screening experiments and the cheap prediction from the pre-trained model. However, the definition of the score introduces an influential hyperparameter, whose value would be critical to the performance of the method.

5.6 Future Development of the Fitness Predictor

The proposed perceptron-training algorithm produced results with observable improvements over the DE benchmarks in certain areas. For small budget sizes, it outperforms the SMW benchmark while producing comparable results to the recombination benchmark and for larger budget sizes it guarantees to identify a functional variant whereas the benchmarks, especially SMW, have shown to sometimes fail to find any improving variants on both datasets. However, the overall performance of the perceptron-training algorithm is the weakest among the proposed methods.

There are several possible areas of improvement, which probably all contribute to the suboptimal performance of this method. Firstly, the algorithm and its version with PPLM finetuning contain a large number of hyperparameters and choices which require a comprehensive exploration. A few alterations were tested with no strong effects on performance, but a grid search of all the different combinations of choices could not be conducted in the span of this thesis. The following recapitulation of considered options and hyperparameters should be viewed as a suggestion for future continuation of development for the method. In the context of the fitness predictor itself, single-layer and two-layer perceptron architectures were tested, for the loss function, L1, L2, and a fitness-weighted L1 loss were considered, and learning rates from 10^{-3} to 10^{-6} were tried at some point during the development. Additional key hyperparameters of the training, which were not experimentally explored, include the number of epochs of the training in each iteration of the algorithm and the batch size used for training data. Finally, it would be interesting to experiment with re-training on the screened variants from past iterations, possibly with a differently weighted loss for differently aged training data to simulate a simple attention mechanism.

Secondly, the results suggest that the addition of fine-tuning to the algorithm decreased the predictive performance rather than helping to train the predictor more efficiently. Most of the aforementioned hyperparameters and options for training of the predictor also apply to the finetuning of the used PPLM, where very little exploration of the different settings was conducted. Additionally, it has been suggested in literature, that a number of protein sequences from an unrelated dataset could be included with the variants of the engineered protein during the fine-tuning, which should serve as regularization to prevent the collapse of the model [31]. This could be the answer to the observed negative effect of fine-tuning on the performance of the implemented method.

Lastly, only one method of both the initial passive sampling and the consecutive active sampling was explored. Both of the used methods are relatively simple, which is a good choice for the initial version of the method to limit the already large number of hyperparameters and other choices. However, in the future development of the method, more complex methods of sampling could yield better results. For example, the employed PPLM used as an embedding extractor can be exploited further for the sampling of variants [55, 56, 31]. Alternatively, an active sampling method which prioritizes variants with high predicted fitness could replace the sampling procedure by maximizing distance on the output to boost the number of high fitness variants in the training dataset. However, all of these developments, which would further complicate the method, should probably only be considered after fine-tuning most of the hyperparameters mentioned before at least to some extent.

5.7 Embedding Extractor Model

In terms of the embedding extractor, a different PPLM, the Progen2 model [51], was originally selected instead of the currently suggested ESM-1b model [6]. The perceptron fitness predictor was tested extensively on the embeddings extracted by Progen2. It was concluded that the Progen2 embedding is uninformative and unfit for the prediction task, because in no tested configuration did the perceptron fitness predictor show any significant predictive performance. Even when trained on the whole dataset, or a large amount of embeddings from filtered, high-fitness variants, the fitness predictor always universally predicted a fitness value close to the mean of the training dataset. Afterward, the ESM-1b model was chosen as a replacement for its widespread use in MLDE-related literature.

In the future, employing different PPLMs as embedding extractors in each of the three proposed methods could yield interesting results. Especially larger models than the used ESM-1b model could be an interesting prospect for experimentation since the LLMs are known to show *emergent abilities* with an increasing number of parameters, which do not manifest in smaller models at all [26]. It should be noted, that a smaller version of the Progen2 model was used in most of the conducted experiments. Because of the aforementioned *emergent abilities*, employing a larger checkpoint of the Progen2 model could, in theory, yield results. Experimenting with a different PPLM, however, is probably

more advisable. The ESM repository, which includes the used ESM-1b model, hosts a wide range of models which could serve as an initial point of exploration. Namely, the general-purpose ESM-2 model [28], which in its largest version contains 15B parameters, seems like the most logical next step in experimentation with different PPLMs if technical limitations allow for its employment. With more limited resources, the ESM-1v model [29] could be employed as the newer, less explored version of the ESM-1b model. Alternatively, the review of existing PPLMs in section 2.3.1 can provide a guiding hand in selecting a different model outside of the ESM family.

6 Conclusion

In this thesis, a concise introduction to machine-learning-assisted directed evolution (MLDE) as an approach to protein engineering was provided and the basic concepts of large language models, passive sampling, active learning, and Bayesian optimization were described. A review of existing pre-trained protein language models (PPLMs) was conducted as well as of their applications in in-silico active learning of proteins. The use of the PPLMs as informative embedding extractors was suggested as the primary, most effective way to exploit them in directed evolution. The informative embedding provides active learning and optimization methods with a sensible metric of similarity between variants, which is essential to effective selection of screened variants. Three methods employing a PPLM as an embedding extractor were proposed and evaluated in silico on two datasets of protein variants with known screening outcomes. The performance of the proposed methods was compared to state-of-the-art methods for MLDE and two simulated classical approaches to directed evolution, which do not employ machine learning. The main metric of evaluation was the number of screened variants relative to the fitness of the final protein.

From the three proposed MLDE methods, Bayesian optimization in embedding space (BOES) shows the most potential. The BOES method outperforms state-of-the-art model-fitting methods. Moreover, the novel representation of the input space based on the sequence embeddings extracted by a pre-trained protein language model (PPLM) has been shown to significantly improve the performance of Bayesian optimization (BO) over optimization on the original protein sequence space. The BOES method produces extremely similar results to other state-of-the-art BO-based methods which employ different informative representations of the input space while saving screening costs on its construction. This can mean either saved resources on experimental costs or more resources for additional iterations of DE, yielding better results with the same number of conducted screening in total. For future development, we suggest combining the innovative input space representation proposed in this thesis with three improvements to the standard BO procedure suggested in [63]. We order the suggestions based on their effect on the performance of the ODBO method [63]. Firstly, conducting prescreening outlier detection via *Extreme Gradient Boosting Outlier Detection* [85] in later iterations. Secondly, implementing a BO procedure robust to outliers based on [98] and finally, employing trust region BO.

The second proposed optimization method, NSDE, performs a greedy search in a neighborhood graph based on the similarity of protein sequence embedding extracted from variants by a PPLM. Testing on the GB1 dataset shows that NSDE is capable of identifying the globally optimal variant in under 250 screened variants, outperforming state-of-the-art model-regression MLDE methods. However, the necessary amount of screened variants to identify a high-fitness variant varies greatly between runs with sampled, non-functional starting variants. The method seems to waste resources on over-exploring of areas around local maxima which can lead to slow progression through the sequence space. Three suggestions for lowering the search effort of NSDE in the future have been proposed. The first option is to explore standard methods applicable to graph-search algorithms. Beam search, a common heuristic algorithm, was suggested as a starting point. The second option is to exploit other methods discussed in this thesis. The proposed BOES method could replace the passive sampling at the start of the NSDE algorithm. During the initial sampling, BO would strike a balance between even exploration of the sequence space and identification of functional variants. This would provide NSDE with a functional variant present in the initial open set more regularly than the passive sampling. Finally, the PPLM used as an embedding extractor could be exploited further to help dictate the order in which variants in the open list are explored based on the variants' pseudolikelihood.

The last implemented method is based on AFP-DE procedure [31] but implements a different *exploration* stage. Unlike BOES and NSDE, this method employs regression of a fitness predictor model as its objective rather than directly optimizing maximum fitness among the screened variants. Because of the tangential objective, a number of variants with high predicted fitness have to be screened additionally after the training procedure. The fitness predictor consists of a PPLM sequence embedding extractor and a perceptron which predicts fitness from the informative embedding. In the *exploration* stage proposed in this thesis, the trained fitness predictor is exploited to maximize the diversity in fitness values in the constructed training set. In a variant of the implemented algorithm, the PPLM fitness predictor is also fine-tuned in each iteration on a large number of variants with high predicted fitness. The predictive performance of the predictor collapses with a larger screening budget, suggesting an unideal setting of hyperparameters in the training procedure. In the future, the current implementation of the method requires a wider exploration of the hyperparameters, possibly by grid search. Additionally, the original AFP-DE paper suggests including a number of unrelated protein sequences in the fine-tuning step to prevent a collapse of the PPLM [31]. The alternative version of the fine-tuning step could improve the predictive performance of the trained fitness predictor, whereas the results of the current fine-tuning step suggest a negative effect. Lastly, after the training procedures of the fitness predictor itself and the PPLM will have been optimized, further experimentation with different initial passive sampling and consecutive active selection of training variants is in place. Either the PPLM can be exploited further for sampling of variants or a more complex active selection method can be implemented which would prioritize variants with high predicted fitness to boost their representation in the training set.

References

[1] Qurban Ali et al. "Protein engineering: A brief overview methodologies and applications". In: *Life Science Journal* 13.12 (2016).

[2] James R Beasley and Michael H Hecht. "Protein design: the choice of de novo sequences". In: *Journal of Biological Chemistry* 272.4 (1997), pp. 2031–2034.

[3] Yajie Wang et al. "Directed evolution: methodologies and applications". In: *Chemical reviews* 121.20 (2021), pp. 12384–12444.

[4] 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.

[5] Bruce J Wittmann et al. "Advances in machine learning for directed evolution". In: *Current opinion in structural biology* 69 (2021), pp. 11–18.

[6] 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.

[7] Jesse Vig et al. "Bertology meets biology: Interpreting attention in protein language models". In: *arXiv preprint arXiv:2006.15222* (2020).

[8] Roshan Rao et al. "Transformer protein language models are unsupervised structure learners". In: *Biorxiv* (2020), pp. 2020–12.

[9] Bruce J Wittmann, Yisong Yue, and Frances H Arnold. "Informed training set design enables efficient machine learning-assisted directed protein evolution". In: *Cell systems* 12.11 (2021), pp. 1026–1045.

[10] David M McCandlish. "Visualizing fitness landscapes". In: *Evolution* 65.6 (2011), pp. 1544–1558.

[11] Bobak Shahriari et al. "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[12] Sandhya R Shenoy and B Jayaram. "Proteins: sequence to structure and function-current status". In: *Current Protein and Peptide Science* 11.7 (2010), pp. 498–514.

[13] John Maynard Smith. "Natural selection and the concept of a protein space". In: *Nature* 225.5232 (1970), pp. 563–564.

[14] Niles A Pierce and Erik Winfree. "Protein design is NP-hard". In: *Protein engineering* 15.10 (2002), pp. 779–782.

[15] Ryan E Cobb, Ran Chao, and Huimin Zhao. "Directed evolution: past, present, and future". In: *AIChE Journal* 59.5 (2013), pp. 1432–1440.

[16] Michael S Packer and David R Liu. "Methods for the directed evolution of proteins". In: *Nature Reviews Genetics* 16.7 (2015), pp. 379–394.

[17] Philip A Romero and Frances H Arnold. “Exploring protein fitness landscapes by directed evolution”. In: *Nature reviews Molecular cell biology* 10.12 (2009), pp. 866–876.

[18] Jianzhi Zhang. “Protein-length distributions for the three domains of life”. In: *Trends in Genetics* 16.3 (2000), pp. 107–109.

[19] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. “Natural language processing: an introduction”. In: *Journal of the American Medical Informatics Association* 18.5 (2011), pp. 544–551.

[20] Alan M Turing. “Computing Machinery and Intelligence.” In: *Creative Computing* 6.1 (1980), pp. 44–53.

[21] David B Searls. “The linguistics of DNA”. In: *American Scientist* 80.6 (1992), pp. 579–591.

[22] Shuang Zhang et al. “Applications of transformer-based language models in bioinformatics: a survey”. In: *Bioinformatics Advances* 3.1 (2023), vbad001.

[23] Wayne Xin Zhao et al. “A survey of large language models”. In: *arXiv preprint arXiv:2303.18223* (2023).

[24] Lawrence Rabiner and Biinghwang Juang. “An introduction to hidden Markov models”. In: *ieee assp magazine* 3.1 (1986), pp. 4–16.

[25] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[26] Jason Wei et al. “Emergent abilities of large language models”. In: *arXiv preprint arXiv:2206.07682* (2022).

[27] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[28] Zeming Lin et al. “Language models of protein sequences at the scale of evolution enable accurate structure prediction”. In: *bioRxiv* (2022).

[29] 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.

[30] Brian L Hie, Kevin K Yang, and Peter S Kim. “Evolutionary velocity with protein language models predicts evolutionary dynamics of diverse proteins”. In: *Cell Systems* 13.4 (2022), pp. 274–285.

[31] Ming Qin et al. “Active Finetuning Protein Language Model: A Budget-Friendly Method for Directed Evolution”. In: *ECAI 2023*. IOS Press, 2023, pp. 1914–1921.

[32] Baris E Suzek et al. “UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches”. In: *Bioinformatics* 31.6 (2015), pp. 926–932.

[33] Chloe Hsu et al. “Learning inverse folding from millions of predicted structures”. In: *ICML* (2022). DOI: 10.1101/2022.04.10.487779. URL: https://www.biorxiv.org/content/early/2022/04/10/2022.04.10.487779.

[34] Roshan Rao et al. “MSA Transformer”. In: *bioRxiv* (2021). DOI: 10.1101/2021.02.12.430858. URL: https://www.biorxiv.org/content/10.1101/2021.02.12.430858v1.

[35] Aakanksha Chowdhery et al. “Palm: Scaling language modeling with pathways”. In: *Journal of Machine Learning Research* 24.240 (2023), pp. 1–113.

[36] OpenAI. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[37] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[38] Ahmed Elnaggar et al. “Prottrans: Toward understanding the language of life through self-supervised learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.10 (2021), pp. 7112–7127.

[39] Zihang Dai et al. “Transformer-xl: Attentive language models beyond a fixed-length context”. In: *arXiv preprint arXiv:1901.02860* (2019).

[40] Zhilin Yang et al. “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems* 32 (2019).

[41] Zhenzhong Lan et al. “Albert: A lite bert for self-supervised learning of language representations”. In: *arXiv preprint arXiv:1909.11942* (2019).

[42] Kevin Clark et al. “Electra: Pre-training text encoders as discriminators rather than generators”. In: *arXiv preprint arXiv:2003.10555* (2020).

[43] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[44] Martin Steinegger, Milot Mirdita, and Johannes Söding. “Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold”. In: *Nature methods* 16.7 (2019), pp. 603–606.

[45] Daniel Hesslow et al. “RITA: a Study on Scaling Up Generative Protein Sequence Models”. In: *arXiv preprint arXiv:2205.05789* (2022).

[46] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[47] Jianlin Su et al. “Roformer: Enhanced transformer with rotary position embedding”. In: *Neurocomputing* 568 (2024), p. 127063.

[48] Ofir Press, Noah A Smith, and Mike Lewis. “Train short, test long: Attention with linear biases enables input length extrapolation”. In: *arXiv preprint arXiv:2108.12409* (2021).

[49] Ali Madani et al. “Progen: Language modeling for protein generation”. In: *arXiv preprint arXiv:2004.03497* (2020).

[50] Nicholas C Wu et al. “Adaptation in protein fitness landscapes is facilitated by indirect paths”. In: *Elife* 5 (2016), e16965.

[51] Erik Nijkamp et al. “ProGen2: exploring the boundaries of protein language models”. In: *Cell Systems* 14.11 (2023), pp. 968–978.

[52] Martin Steinegger and Johannes Söding. “Clustering huge protein sequence sets in linear time”. In: *Nature communications* 9.1 (2018), p. 2542.

[53] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. “ProtGPT2 is a deep unsupervised language model for protein design”. In: *Nature communications* 13.1 (2022), p. 4348.

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

[55] Trong Thanh Tran and Truong Son Hy. “Protein Design by Directed Evolution Guided by Large Language Models”. In: *bioRxiv* (2023), pp. 2023–11.

[56] Brian L Hie et al. “Efficient evolution of human antibodies from general protein language models”. In: *Nature Biotechnology* (2023).

[57] Gioele La Manno et al. “RNA velocity of single cells”. In: *Nature* 560.7719 (2018), pp. 494–498.

[58] Trevor S Frisby and Christopher James Langmead. “Identifying promising sequences for protein engineering using a deep transformer protein language model”. In: *Proteins: Structure, Function, and Bioinformatics* (2023).

[59] Burr Settles. “Active learning literature survey”. In: (2009).

[60] Michael J Lopez and Shamim S Mohiuddin. “Biochemistry, essential amino acids”. In: (2020).

[61] Dongrui Wu, Chin-Teng Lin, and Jian Huang. “Active learning for regression using greedy sampling”. In: *Information Sciences* 474 (2019), pp. 90–105.

[62] Hwanjo Yu and Sungchul Kim. “Passive sampling for regression”. In: *2010 IEEE international conference on data mining*. IEEE. 2010, pp. 1151–1156.

[63] Lixue Cheng et al. “ODBO: Bayesian optimization with search space prescreening for directed protein evolution”. In: *arXiv preprint arXiv:2205.09548* (2022).

[64] David Simoncini et al. “Fitness landscape analysis around the optimum in computational protein design”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2018, pp. 355–362.

[65] Brian L Hie and Kevin K Yang. “Adaptive machine learning for protein engineering”. In: *Current opinion in structural biology* 72 (2022), pp. 145–152.

[66] Carl Edward Rasmussen. “Gaussian processes in machine learning”. In: *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[67] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[68] Robert A Wagner and Michael J Fischer. "The string-to-string correction problem". In: *Journal of the ACM (JACM)* 21.1 (1974), pp. 168–173.

[69] Trevor S Frisby and Christopher J Langmead. "Fold family-regularized bayesian optimization for directed protein evolution". In: *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2020.

[70] Richard S Sutton, Andrew G Barto, et al. "Reinforcement learning". In: *Journal of Cognitive Neuroscience* 11.1 (1999), pp. 126–134.

[71] Javad Azimi, Alan Fern, and Xiaoli Fern. "Batch Bayesian optimization via simulation matching". In: *Advances in neural information processing systems* 23 (2010).

[72] Philip A Romero, Andreas Krause, and Frances H Arnold. "Navigating the protein fitness landscape with Gaussian processes". In: *Proceedings of the National Academy of Sciences* 110.3 (2013), E193–E201.

[73] Thomas Desautels, Andreas Krause, and Joel W Burdick. "Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization." In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 3873–3923.

[74] Javier González et al. "Batch Bayesian optimization via local penalization". In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 648–657.

[75] Kevin K Yang et al. "Batched stochastic Bayesian optimization via combinatorial constraints design". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3410–3419.

[76] Brian Hie, Bryan D Bryson, and Bonnie Berger. "Leveraging uncertainty in machine learning accelerates biological discovery and design". In: *Cell systems* 11.5 (2020), pp. 461–477.

[77] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[78] Niranjan Srinivas et al. "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *arXiv preprint arXiv:0912.3995* (2009).

[79] Adam D Bull. "Convergence rates of efficient global optimization algorithms." In: *Journal of Machine Learning Research* 12.10 (2011).

[80] Ilia G Denisov et al. "Structure and chemistry of cytochrome P450". In: *Chemical reviews* 105.6 (2005), pp. 2253–2278.

[81] Karl Deisseroth and Peter Hegemann. "The form and function of channelrhodopsin". In: *Science* 357.6356 (2017), eaan5544.

[82] Claire N Bedbrook et al. "Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization". In: *PLoS computational biology* 13.10 (2017), e1005786.

[83] Jonathan C Greenhalgh et al. "Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production". In: *Nature communications* 12.1 (2021), p. 5825.

[84] Xin Qiu, Elliot Meyerson, and Risto Miikkulainen. "Quantifying point-prediction uncertainty in neural networks via residual estimation with an i/o kernel". In: *arXiv preprint arXiv:1906.00588* (2019).

[85] Yue Zhao and Maciej K Hryniewicki. "Xgbod: improving supervised outlier detection with unsupervised representation learning". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8.

[86] Matouš Soldát. *DESilico.jl (Directed Evolution in Silico)*. URL: `https://github.com/soldatmat/DESilico.jl`.

[87] Anna I Podgornaia and Michael T Laub. "Pervasive degeneracy and epistasis in a protein-protein interface". In: *Science* 347.6222 (2015), pp. 673–677.

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

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

[90] Jason Ansel et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. DOI: `10.1145/3620665.3640366`. URL: `https://pytorch.org/assets/pytorch2-2.pdf`.

[91] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[92] Šimon Soldát. *BOSS.jl (Bayesian Optimization with Semiparametric Surrogate)*. URL: `https://github.com/soldasim/BOSS.jl`.

[93] Michael JD Powell. "The NEWUOA software for unconstrained optimization without derivatives". In: *Large-scale nonlinear optimization* (2006), pp. 255–297.

[94] James Hensman, Nicolo Fusi, and Neil D Lawrence. "Gaussian processes for big data". In: *arXiv preprint arXiv:1309.6835* (2013).

[95] Thomas A Hopf et al. "Mutation effects predicted from sequence co-variation". In: *Nature biotechnology* 35.2 (2017), pp. 128–135.

[96] Yuchi Qiu, Jian Hu, and Guo-Wei Wei. "Cluster learning-assisted directed evolution". In: *Nature computational science* 1.12 (2021), pp. 809–818.

[97] Yuchi Qiu and Guo-Wei Wei. "CLADE 2.0: evolution-driven cluster learning-assisted directed evolution". In: *Journal of chemical information and modeling* 62.19 (2022), pp. 4629–4641.

[98] Ruben Martinez-Cantin, Kevin Tee, and Michael McCourt. "Practical Bayesian optimization in the presence of outliers". In: *International conference on artificial intelligence and statistics*. PMLR. 2018, pp. 1722–1731.

[99] Clara Meister, Tim Vieira, and Ryan Cotterell. "Best-first beam search". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 795–809.

[100] Ashwin K Vijayakumar et al. "Diverse beam search: Decoding diverse solutions from neural sequence models". In: *arXiv preprint arXiv:1610.02424* (2016).

[101] Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. "Improvements in beam search." In: *ICSLP*. Vol. 94. 1994, pp. 2143–2146.