

Artificial Intelligence Methods for Enzyme Engineering

Thesis by
Jason Yang

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy



CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended May 22, 2026

© 2026

Jason Yang

ORCID: 0000-0003-3184-1550

Some rights reserved. This thesis is distributed under a Creative Commons
Attribution-NonCommercial-ShareAlike License

ACKNOWLEDGEMENTS

I will preface this by noting that, for the sake of brevity, I have not explicitly mentioned many people here and have not done justice with the details, but trust that I am grateful for each of you that have supported me during and before my research journey.

First, I would like to thank my advisors, Frances and Yisong. They have given me the freedom to pursue the topics that interest me (in part thanks to funding from the NSF GRFP and Google PhD Fellowship) with the right support and feedback. From Frances, I learned how to choose projects with practical impact and from Yisong I learned how to think more broadly and ambitiously about my work. I could not have asked for better advisors. I'd also like to thank my committee chair Zhen-Gang Wang for his support and mentorship throughout the years and Viviana Gradinaru for her insights on my committee. I am also grateful to have had the privilege to serve as a TA for Steve Mayo twice.

Sabine and Cheryl hold the Arnold lab together, and they have been essential support systems in my day-to-day life. Additionally, I am indebted to all the staff that have facilitated my time at Caltech, including Alma, Sumaia, Allison Kinard, and Mike Vicic. Special thanks to Suresh for helping all things hardware and IT—and the tech transfer office for making my internship at Profluent possible.

I owe much to my first mentors at Caltech, including Kadina, Jennifer Sun, and Bruce. I have also learned much from Raul and Shengchao since then, and Ali Madani, Jeff Ruffolo, and Aadyot Bhatnagar at Profluent. I am also thankful for the previous mentors who supported me in undergrad research: Meny Elimelech, Ying Li, Smita Krishnaswamy, Heather Mayes, Matt Amodio, Ryan DuChanois, Ines Zucker, Zhiqiang Shen, and Lei Tao to name a few. Furthermore, I have had the privilege of collaborating with some of the best people during my time at Caltech, many of whom are friends: Francesca, Ravi, Ariane, Wenda, Yueming, Julie DuCharme, James Bowden, Theo, Amy Lu, Chenghao, Jarrid, Sonia, Masy, Chris Yeh, Amin, Vignesh, Emre, Riccardo, Trevor, and Ariel. Additionally, I have mentored some amazing students whom I am very proud of: Annika, Bryan, Tanvi, Zahra, Daniel Khalil, Kerui, Leo, and Varun. Special thank you to all of the outside collaborations I've had and to the friends I've made in the field of AI for molecules through conferences on online seminar organization – it's truly a special community.

Regarding all of the amazing labmates that I've overlapped with over the years: each of you has made coming into work (and hanging out outside of the office) enjoyable and memorable.

Finally, my time at Caltech wouldn't be complete without my support systems outside of work, namely my family and friends. I'm especially thankful to my parents and younger brother who have always supported me, all my extended family in China, and my grandma in the US. Finally, thank you to all my friends past and present, many of who attended and celebrated my thesis defense. We've built so many memories that I will cherish for the rest of my life. I'm sorry that I cannot list all of you here, but you know who you are. Sending much love.

ABSTRACT

Proteins, such as specialized catalysts called enzymes, offer transformative potential for sustainable chemical synthesis, environmental remediation, and advanced therapeutics. However, engineering proteins for specific industrial or clinical functions remains a formidable challenge due to the expansive and high-dimensional sequence design space. While directed evolution has facilitated significant breakthroughs, the methodology is often constrained by slow iteration cycles and the requirement for a functional starting point. This thesis introduces novel machine learning frameworks, integrated with experimental workflows, to transcend traditional enzyme engineering approaches. We first present Active Learning-Assisted Directed Evolution (ALDE), which employs Bayesian optimization to enable more efficient optimization of protein properties ([Chapter 2](#)). Afterward, Contrastive Reaction-Enzyme Pretraining (CREEP) is introduced for the annotation and discovery of enzymes with desired “new-to-nature” functionalities ([Chapter 3](#)). Finally, a new paradigm for Steering Generation for Protein Optimization (SGPO) is demonstrated, unifying these two perspectives into a holistic generative framework for efficient protein engineering ([Chapter 4](#)). Collectively, these innovations advance the transition toward automated, artificial intelligence-driven biomolecular design—unlocking sustainable synthesis, novel therapeutics, and programmable biology at the molecular level.

PUBLISHED CONTENT AND CONTRIBUTIONS

*denotes equal contribution

1. **Yang, J.;** Li, F.-Z.; Long, Y.; Arnold, F. H. Illuminating the Universe of Enzyme Catalysis in the Era of Artificial Intelligence. *Cell Systems* 2026, 17 (2). <https://doi.org/10.1016/j.cels.2025.101372>

J.Y. participated in the conception of the perspective article and wrote the manuscript with input from all authors.

2. **Yang, J.*;** Chu, W.*; Khalil, D.; Astudillo, R.; Wittmann, B.J.; Arnold, F. H.; Yue, Y. Steering Generative Models with Experimental Data for Protein Fitness Optimization. *Advances in Neural Information Processing Systems* 2025. <https://doi.org/10.48550/arXiv.2505.15093>

J.Y. participated in the conception of the study, determined the datasets and evaluation tasks, participated in the building and training of computational methods, and wrote the manuscript with input from all authors.

3. **Yang, J.*;** Lal, R. G.*; Bowden, J. C.; Astudillo, R.; Hameedi, M. A.; Kaur, S.; Hill, M.; Yue, Y.; Arnold, F. H. Active Learning-Assisted Directed Evolution. *Nat. Commun.* 2025, 16 (1), 714. <https://doi.org/10.1038/s41467-025-55987-8>

J.Y. participated in the conception of the study, analyzed the data, built and trained the computational methods, and wrote the manuscript with input from all authors.

4. **Yang, J.;** Mora, A.; Liu, S.; Wittmann, B. J.; Anandkumar, A.; Arnold, F. H.; Yue, Y. CARE: A Benchmark Suite for the Classification and Retrieval of Enzymes. *Advances in Neural Information Processing Systems* 2024. <https://doi.org/10.48550/arXiv.2406.15669>

J.Y. participated in the conception of the study, compiled the data, participated in determining the evaluation tasks, built and trained the computational methods, and wrote the manuscript with input from all authors.

5. **Yang, J.;** Li, F.-Z.; Arnold, F. H. Opportunities and Challenges for Machine Learning-Assisted Enzyme Engineering. *ACS Cent. Sci.* 2024, 10 (2), 226–241. <https://doi.org/10.1021/acscentsci.3c01275>

J.Y. participated in the conception of the outlook article, performed necessary literature review, and wrote the manuscript with input from all authors.

TABLE OF CONTENTS

Acknowledgements	iii
Abstract	v
Published Content and Contributions	vi
Table of Contents	vi
Chapter I: An Introduction to Enzyme Engineering	1
1.1 Motivation: Illuminating the Universe of Enzyme Catalysis	1
1.2 The Current Approach for Enzyme Engineering	4
1.3 Artificial Intelligence is Revolutionizing Enzyme Engineering	6
Chapter II: Active Learning-Assisted Directed Evolution	10
2.1 Introduction	10
2.2 Results	13
2.3 Discussion	23
Chapter III: Computational Classification and Retrieval of Enzyme Starting Points	26
3.1 Introduction	26
3.2 Related Work	28
3.3 Overview of CARE	30
3.4 Task 1: Enzyme Classification	31
3.5 Task 2: Enzyme Retrieval	35
3.6 Discussion	39
Chapter IV: Steering Generative Models with Experimental Data for Protein Fitness Optimization	43
4.1 Introduction	43
4.2 Related Work	47
4.3 Problem Setup	50
4.4 Results	50
4.5 Discussion	55
Chapter V: Conclusion and Outlook	58
5.1 Toward General, Autonomous Protein Engineering	58
5.2 Closing Thoughts	61
Bibliography	63
Appendix A: Method Details	87
A.1 Methods for Chapter 2	87
A.2 Methods for Chapter 3	93
A.3 Methods for Chapter 4	103
Appendix B: Supplementary Results	117

Chapter 1

AN INTRODUCTION TO ENZYME ENGINEERING

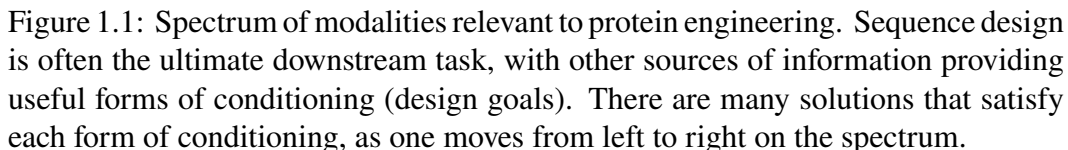
- (1) Yang, J.; Li, F.-Z.; Long, Y.; Arnold, F. H. *Cell Systems* **2026**, *17*, 101372.
- (2) Yang, J.; Li, F.-Z.; Arnold, F. H. *ACS Central Science* **2024**, *10*, 226–241.

Abstract Scientific research has revealed only a minuscule fraction of the enzymes that evolution has generated to power life’s essential chemical reactions—and an even tinier fraction of the vast universe of *possible* enzymes. Beyond the enzymes already annotated lie an astronomical number of biocatalysts that could enable sustainable chemical production, degrade toxic pollutants, and advance disease diagnosis and treatment. For the past few decades, directed evolution has been a powerful strategy for reshaping enzymes to access new chemical transformations: by harnessing nature’s existing diversity as starting points and taking inspiration from nature’s most powerful design process, evolution, to modify enzymes incrementally. Recently, artificial intelligence (AI) methods are revolutionizing how we understand and compose the language of life. In this introduction, we discuss a vision for AI-driven enzyme discovery to unveil a world of enzymes that transcend biological evolution and contextualize the individual components of this thesis that contribute toward that vision.

1.1 Motivation: Illuminating the Universe of Enzyme Catalysis

Nearly all of biology’s chemistry is catalyzed by protein catalysts, the enzymes. Enzymes drive cell growth and development, repair wounds, and generate wonderfully complex and functional materials such as wood or shells—all while deriving energy and starting materials from their environments. Enzymes have an unrivaled ability to catalyze chemical reactions cleanly, selectively, and under mild, physiological conditions. These brilliant natural inventions offer powerful and sustainable chemical routes to make food ingredients, materials, fuels, and specialty chemicals; to degrade pollutants into recyclable or biodegradable forms; and to diagnose and treat disease [1–4].

Imagine that we could expand biology’s DNA-encoded chemistry to encompass even more useful chemical transformations, including the best of those invented by



However, the vast majority of the staggering number of *possible* proteins—orders of magnitude greater than the number of particles in the universe—are either non-functional or unsuitable (Fig. 1.2A) [7, 8]. There are unfortunately many, many more ways to be non-functional in the library of possible genomes, just as the vast majority of Babel’s books contain pure gibberish. The collection of all possible proteins is also vast, and the density of “meaningful” sequences is miniscule.

But, unlike the librarians of Borges’ random book collection, we the librarians of the protein collection have an easier time finding the meaningful sequences, because we have evolution to guide us. John Maynard Smith described a conceptual space such that each protein sequence is surrounded by all its single-mutant neighbors [9]. For evolution to occur, he argued, a functional protein must be surrounded by at least one other functional protein. Evolution could then pass through the network of functional protein sequences to explore new possibilities, one mutation at a time.

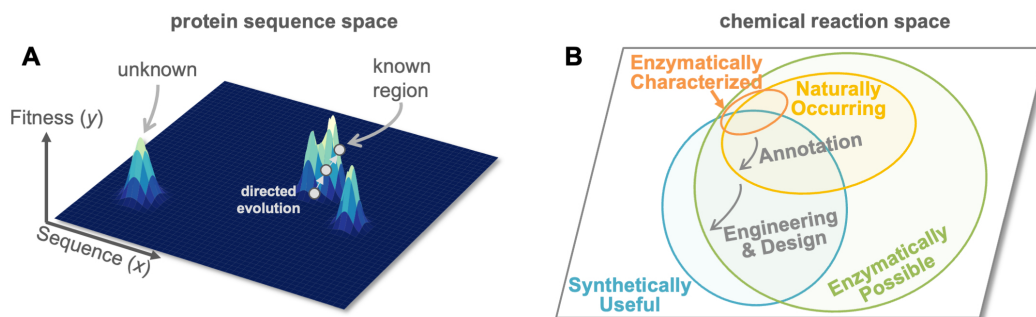


Figure 1.2: (A) Directed evolution has been a powerful tool for expanding the functional space of enzymes and achieving specific fitness goals, in a vast design space where most sequences are not functional. This search has been largely constrained to local search from existing functions, leaving many useful regions of sequence space underexplored and distant regions completely unknown. (B) Only a minuscule fraction of enzymatic chemical reaction space (function space) that would be useful for human purposes has been characterized.

This process has given rise to the billions upon billions of proteins found in nature today—rare meaningful sequences that we can literally scrape from the bottom of our shoes. And it can give rise to many more. Such an exploration, one (or a few) mutations at a time, can discover and improve enzymes, including ones that would be particularly useful to humans [1]. These directed evolution “greedy uphill walks” (Fig. 1.2A) have led to the thousands of improved enzymes used in products from laundry detergents to disease therapies [4].

Directed evolution (DE) tackles the problem of optimizing extant enzymes for specific fitness goals such as enhanced stability or enzymatic activity, but it can also be used to discover and explore whole new functional spaces. By taking advantage of the ability of enzymes to catalyze reactions that are not their native ones, a property called catalytic promiscuity [10], one can obtain enzymes that operate on whole new substrates [11, 12] and even perform chemistry that is unknown in nature, such as the formation or cleavage of carbon-silicon bonds [4, 13–15]. One day, perhaps, all useful chemistry could be encoded in DNA and performed cleanly, as nature does. Of course, we do not know if all relevant chemistry can be accomplished by enzymes—but with so many sequence possibilities, and with the assistance of cofactors like metals—it is likely that many, if not all, useful chemistries could be encoded in DNA and catalyzed by enzymes (Fig. 1.2B).

1.2 The Current Approach for Enzyme Engineering

At its core, protein engineering is a design problem: the goal is to generate and/or alter a protein's amino acid sequence to encode a desired function. "Fitness" is a numerical quantification of that desired function, which may include multiple features that contribute to overall performance. Altering fitness is equivalent to traversing the protein's fitness landscape, which is a surface in high-dimensional space that maps sequence to fitness (Fig. 1.2A). Protein engineering is challenging because accurate biophysical prediction methods for determining protein fitness are rare or nonexistent, and the search space of possible proteins is beyond-astronomically large [16]. To make matters worse, functional proteins are scarce in the space of all protein sequences, and finding an optimal sequence on this protein fitness landscape is NP-hard, as there is no known polynomial-time solution [17].

Under the umbrella of protein engineering, enzyme engineering poses some unique challenges: catalysis is more complex than binding and may involve multiple substrates, cofactors, and elementary steps. Furthermore, typical experimental screening methods for measuring enzymatic fitness are lower throughput than binding assays, for which powerful positive and negative selections can usually be devised. Enzymes are often engineered to enhance their native functions, or alternatively to target "promiscuous" activities, such as reactivity on non-native substrates or even non-native reactivities (Fig. 1.3A) [13].

As a result, engineering an enzyme involves discovering an enzyme with some initial level of activity (satisfying some but not all desired properties), followed by fitness improvement using DE (Fig. 1.3) [18]. Thus, the first step of an enzyme engineering workflow involves identifying (or designing) an enzyme with some measurable fitness. Consider engineering an enzyme to catalyze a new chemical reaction. To find a new activity that is related to a known activity, one might screen previously engineered enzymes for "promiscuous" activity for the desired function (Fig. 1.3A) [19, 20]. If none is detected, it may be necessary to explore other known enzymes or proteins in annotated databases (Fig. 1.3B) [21]. Those with active sites amenable to accommodating a particular substrate, evolvable folds, cofactors relevant to a desired activity, or similar mechanisms may be valid starting points. However, these approaches rely too much on experimental intuition and luck, and such an Edisonian search through existing proteins is inefficient and often ineffective. Even if activity is found, the enzyme might need to be stabilized so that it has suitable behavior for screening or can undergo further evolution, and it must

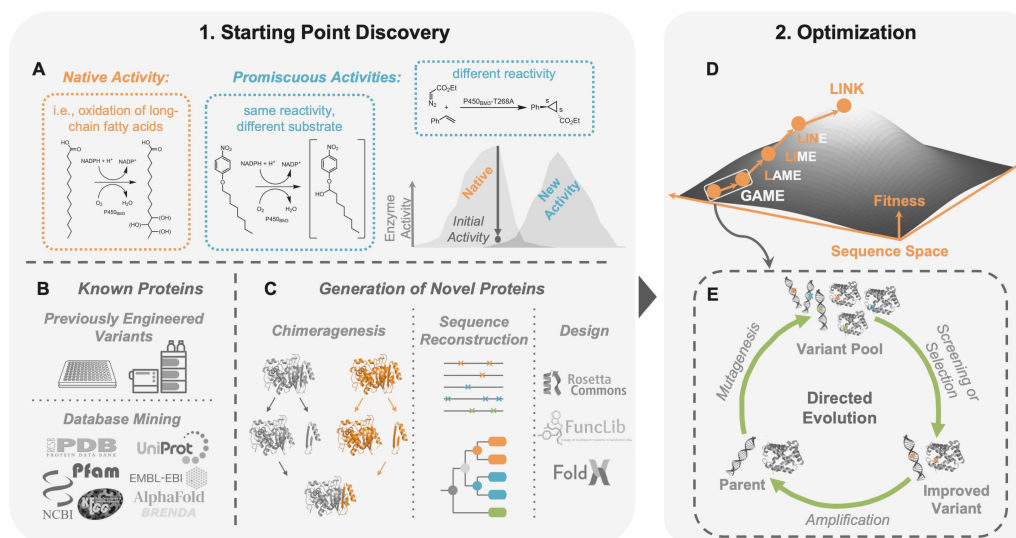


Figure 1.3: The enzyme engineering workflow. Enzyme engineering begins with a discovery phase to identify an enzyme with initial activity (desired function). If fitness is not sufficient, the enzyme is then optimized using DE. **(A)** Enzyme discovery involves screening for desired activities, which could include native activity or promiscuous activities. **(B)** Enzyme starting points can be found in known proteins or by **(C)** diversification of enzymes using various computational methods to generate starting sequences that are more stable and evolvable. **(D-E)** In its simplest form, optimization using DE involves generating a pool of protein variants, identifying one with improved fitness, and using this variant as the starting point for the next generation of mutation and screening. DE can be thought of as a greedy hill climb on a protein fitness landscape. The natural ordering of sequences in the DE fitness landscape is that all sequences are surrounded by their single-mutant neighbors [9].

express well in the host organism, such as bacteria or yeast.

Computationally assisted methods such as chimeragenesis and ancestral sequence reconstruction have emerged to propose diverse protein starting points (sometimes having higher stability, evolvability, different substrate scopes, Fig. 1.3C) [22–24]. Methods aided by software suites such as Rosetta have been successful in re-designing enzymes and enhancing their stabilities [25–30], but *de novo* enzyme design is still nascent and works well only for relatively simple reactions [31–33]. Protein binder design is often framed as a structure design problem, as binding interactions can be modelled relatively accurately with physics, given a target structure [34, 35]. In contrast, enzyme engineering requires precisely positioned catalytic residues and an active site capable of accommodating reactant(s) and intermediates(s), often containing metals and cofactors that can be challenging to

model computationally [36]. Transition state energies can be useful predictors of enzymatic activity [37], but many enzyme reaction mechanisms remain poorly understood. Further complicating matters, enzymes are tasked with simultaneously balancing overall protein stability with dynamic conformation flexibility—an interplay that is poorly understood [38–40]. A single mutation, even one distant from the active site, can affect activity in ways that are difficult to explain, much less to predict [1]. Thus, before the advent of deep learning, impressive examples of rationally designed enzymes were relatively rare [31, 41, 42].

Once a suitable enzyme with measurable function is identified, fitness can be improved by DE and related techniques [43, 44]. DE sidesteps the need to understand protein sequence-fitness relationships and optimizes protein fitness by performing greedy hill climbing on the protein fitness landscape (Fig. 1.3D) [1, 9, 16]. In its simplest form, DE involves accumulating beneficial mutations by making mutations to the protein (mutagenesis) and screening for variant(s) with higher performance on target properties (Fig. 1.3E). The targeted properties can change during optimization by changing the screening criteria, and informative screens can investigate multiple properties simultaneously. Recombination is often used to shuffle beneficial mutations so that screening can identify mutation combinations that further increase fitness [45, 46]. DE takes advantage of the fact that functional sequences are clustered in sequence space, i.e., functional sequences are surrounded by many other functional sequences, and smooth uphill paths exist in the landscape [9]. Due to the complex nature of rational enzyme design, DE has historically been the primary driver for improving enzyme fitness and achieving non-natural activities.

1.3 Artificial Intelligence is Revolutionizing Enzyme Engineering

Despite its many successes, DE is not the most efficient way to explore the vast space of possible enzymes. DE has been effective because many protein-fitness landscapes are smooth in at least some of their many dimensions, but it faces severe limitations when this is not the case [16, 47, 48]. It is also slow and resource-intensive, and, as a local search in sequence space, it will ignore other parts of the space that may hold even greater functional potential (Fig. 1.2A) [16]. Furthermore, directed evolution requires a starting point with some level of the desired function, or at least a strategy for acquiring that function through incremental steps. Artificial intelligence (AI) methods are well-positioned to overcome these major bottlenecks in the quest to unveil new regions of functional enzyme space.

To address the slow, laborious nature of DE (Fig. 1.3D-E), machine learning (ML) has increasingly been used to suggest combinations of beneficial mutations and reduce the burden of having to perform many generations of mutagenesis and screening for protein fitness optimization [49, 50]. Sampled sequence-function data can be used to build a statistical model of the fitness landscape, enabling predictions about combinations of mutations with improved functions, i.e. semi-local expansions in functional and sequence space. Such supervised ML methods can make directed evolution more efficient and effective in the real world [51–54]. Besides learning directly from labeled data collected through experiments, the models constructed from the data can additionally include what we already know of natural proteins, e.g., in the form of zero-shot scores based on evolutionary conservation [55–59] or protein structures and physics [60, 61]. Overall, there are many emerging supervised ML approaches combining the utility of unlabeled data on natural sequences with labeled assay data [53, 55, 58, 59, 61–64]. In this thesis, we extend existing methods into an iterative setting that leverages deep learning and balances future exploration with exploitation of previous knowledge through uncertainty quantification. Namely, in Chapter 2, we demonstrate *active learning-assisted directed evolution (ALDE)*, which advances the ability to navigate protein fitness landscapes efficiently (Optimization, Fig. 1.3D-E).

To attain starting points for protein engineering (Fig. 1.3A-C), scientists are additionally leveraging ML to improve computational methods for protein annotation and discovery [65]. Approximately 250 million protein sequences are catalogued in the UniProt database, but less than 0.3% are annotated with function [66]. Thus, hundreds of millions of known proteins have not been explored as starting points for enzyme engineering. If these proteins could be accurately annotated, protein engineers would have access to a wealth of diverse candidates for engineering. While enzyme engineers have long been using multiple sequence alignments (MSAs) and homology to predict the functions of unannotated protein sequences [67], ML classification models extend these approaches and draw from more complete features describing protein sequences and structures to predict more specific functions, such as type of reactivity and k_{cat} [68–77]. Focusing on known sequences without annotations, many of these methods aim to classify enzyme sequences based on their enzyme commission (EC) numbers, which is a hierarchical classification scheme that divides enzymes into general classes and then further subclasses, based on their catalytic activities. For example, contrastive learning-enabled enzyme annotation (CLEAN) has demonstrated state-of-the-art success at accurately classifying enzyme

sequences based on their EC numbers [75]. Impressively, CLEAN also correctly identified an enzyme with three different EC numbers, corresponding to promiscuous activities, where promiscuity prediction was framed as multitask classification [78]. Still, there is a need to identify enzymes with unannotated, promiscuous activities as starting points for evolving enzymes for non-natural activity [19, 75]. This thesis addresses this challenge by presenting a computational pipeline for functional annotation and retrieval of previously unannotated enzyme activities. Specifically, in [Chapter 3](#), we present a benchmark suite for the classification and retrieval of enzymes (CARE) and an ML method for contrastive reaction-enzyme pretraining (CREEP) (Starting Point Discovery, [Fig. 1.3A-C](#)).

Related to this, data-driven generative AI approaches are revolutionizing our ability to design novel proteins, which could unlock unseen combinations of properties and even new-to-nature activities [50, 65, 79–84]. Generative ML can learn the distribution of proteins that nature has already made and humans have deposited in rapidly growing databases: it learns to compose new proteins using the “language” of nature’s proteins to infer functional sequences that nature has not explored [80, 85–87]. Such generative protein language models, from Potts models [88] to deep learning models [89–95] have generated sequences, for example, of functional lysozymes [93], carbonic anhydrases [94], and even Cas9 nucleases [96] with low sequence identity and activities comparable to naturally occurring enzymes. While generative approaches are powerful techniques for generating proteins with natural functions, these models typically have not utilized labeled sequence-fitness data from real-world assays to optimize proteins for specific fitness properties, or to design proteins with non-native functions. To address this gap, in [Chapter 4](#), we introduce a new paradigm for holistic protein design and optimization that unifies sequence-based generative modeling with supervised learning from labeled sequence-fitness data, called *steered generation for protein optimization (SGPO)*.

Generative ML methods can also integrate structure and physics for *de novo* design of novel folds [97]—mostly binders, but also enzymes, including luciferases [98], serine hydrolases [99], and retro-aldolases [100]. A prevailing approach is to use a generative model [101, 102] to design protein backbone structures that will accommodate a desired active-site conformation (motif scaffolding) [103, 104] or a certain reaction transition state [105, 106]. These structures are then mapped to sequences using inverse folding models based on deep learning [107, 108], as sequence design is generally the ultimate goal. Recently, there have been efforts to

design structures jointly with sequences [109–116] to directly scaffold active-site residues in enzymes (not just the backbone coordinates) and leverage all available data for more reasonable designs [117]. Structure prediction has quickly become an indispensable tool for validating these designs [118, 119], along with models that can predict enzymatic function from sequence [75, 120–122]. Besides conditioning on structure, an interesting direction of research involves using descriptions of function as keywords or natural language descriptions to generate enzymes with desired function [91, 115, 123–125]. While we do not elaborate on specific efforts in this thesis toward function-conditioned *de novo* protein design, we acknowledge that this is an important area of current and future work.

Ultimately, utilizing AI-driven methods, researchers are achieving success rates orders of magnitude higher than those of design methods before the deep learning era, and they are generating real, functional proteins [100, 126]. Furthermore, AI-driven design enables bigger jumps in protein space, allowing for more novelty in functional space or more useful combinations of properties such as high thermostability and catalytic promiscuity, compared to previous approaches. The future of AI-driven protein engineering is bright, and one day, we may in fact genetically encode (via enzymes) the universe of all useful chemical transformations. In Chapter 5, we conclude this thesis with an outlook on how best to use AI to achieve this dream.

Chapter 2

ACTIVE LEARNING-ASSISTED DIRECTED EVOLUTION

- (1) Yang, J.; Lal, R. G.; Bowden, J. C.; Astudillo, R.; Hameedi, M. A.; Kaur, S.; Hill, M.; Yue, Y.; Arnold, F. H. *Nature Communications* **2025**, *16*, 714.

Abstract Directed evolution (DE) is a powerful tool to optimize protein fitness for a specific application. However, DE can be inefficient when mutations exhibit non-additive, or epistatic, behavior. Here, we present Active Learning-assisted Directed Evolution (ALDE), an iterative machine learning-assisted DE workflow that leverages uncertainty quantification to explore the search space of proteins more efficiently than current DE methods. We apply ALDE to an engineering landscape that is challenging for DE: optimization of five epistatic residues in the active site of an enzyme. In three rounds of wet-lab experimentation, we improve the yield of a desired product of a non-native cyclopropanation reaction from 12% to 93%. We also perform computational simulations on existing protein sequence-fitness datasets to support our argument that ALDE can be more effective than DE. Overall, ALDE is a practical and broadly applicable strategy to unlock improved protein engineering outcomes.

2.1 Introduction

Protein engineering is an optimization problem, where the goal is to find the amino acid sequence that maximizes "fitness," a quantitative measurement of the efficacy or functionality for a desired application, from chemical synthesis to bioremediation and therapeutics [4]. Protein fitness optimization can be thought of as navigating a protein fitness landscape, a mapping of amino acid sequences to fitness values, to find higher-fitness variants [16]. However, since protein sequence space is vast, as a protein of length N can take on 20^N distinct sequences and functional proteins are vanishingly rare, finding an optimal sequence is hard. Because functional proteins are surrounded by other functional proteins one mutation away [9], protein engineers often use directed evolution (DE) to optimize protein fitness [43, 44].

In its simplest form, DE involves accumulating beneficial mutations by searching through sequences near one that exhibits some level of desired function for variants that exhibit enhanced performance on a target fitness metric (Fig. 2.1A). This

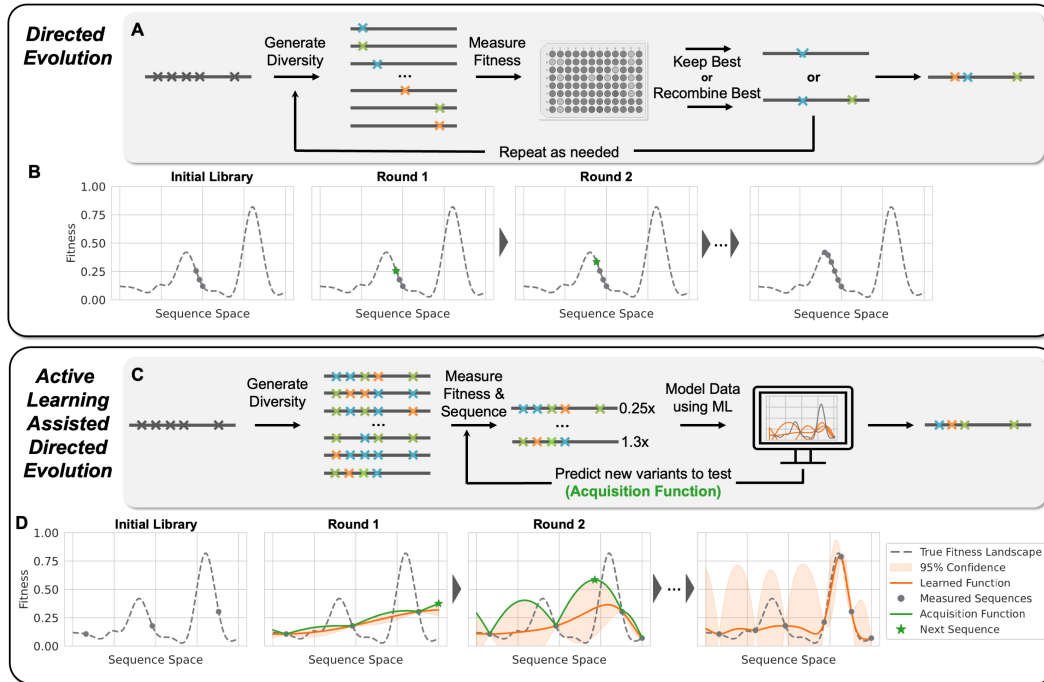


Figure 2.1: Conceptual differences between DE and ALDE. (A) A common workflow for DE, where a starting protein is mutated and fitnesses of variants are measured (screened). The best variant is used as the starting point for the next round of mutation and screening, until desired fitness is achieved. (B) Conceptualization of DE as greedy hill climbing optimization on a hypothetical protein fitness landscape. (C) Workflow for ALDE. An initial training library is generated, where k residues are mutated simultaneously (for example $k=5$). A small subset of this library is randomly picked, after which the variants are sequenced and their fitnesses are screened. A supervised ML model with uncertainty quantification is trained to learn a mapping from sequence to fitness. An acquisition function is used to propose new variants to test, balancing exploration (high uncertainty) and exploitation (high predicted fitness). The process is repeated until desired fitness is achieved. (D) Conceptualization of active learning on a hypothetical protein fitness landscape. Active learning is often more effective than DE for finding optimal combinations of mutations. In these conceptualizations, a single sequence is queried in each round, but in practical settings, active learning operates in batch where multiple sequences are tested in each round.

approach can be thought of as greedy hill climbing optimization across the protein fitness landscape (Fig. 2.1B). DE is limited because screening for performance can only explore a small, local region of sequence space. Additionally, taking one mutational step at a time can cause the experiment to become stuck at a local optimum, especially on rugged protein fitness landscapes where mutation effects exhibit epistasis [48]. Machine learning (ML) techniques offer a pathway to circumvent these obstacles, providing strategies to more efficiently navigate these complex landscapes [49, 50, 127–129].

While supervised ML has been used to propose ideal combinations of mutations—such as in ML-assisted DE (MLDE) [53, 54]—these approaches are often limited to small design spaces as they do not take advantage of the fundamentally iterative manner in which protein engineering can take place in real-world applications. By contrast, active learning is an ML paradigm that gathers data iteratively using a supervised model which is, in turn, updated as new data are acquired (Fig. 2.1C). By leveraging uncertainty quantification to choose which variants should be tested at each step, active learning has the potential to unlock improved engineering outcomes (Fig. 2.1D) [130–134]. Approaches related to active learning have been used in the wet lab to optimize artificial metalloenzymes, nucleases, and other proteins [55, 59, 62, 64]. Past work has also explored the use of Bayesian optimization (BO), a particular class of active learning algorithms, to experimentally improve the thermostability of protein chimeras [51, 135] and to optimize proteins with one to several mutations [134, 136, 137]. However, few studies have explored the utility of active learning methods in comparison to DE, especially where epistatic effects are prevalent [138]. In addition, understanding of the practical role of uncertainty quantification in the context of deep learning [139–141] and high-dimensional [142] representations learned from protein language models [119, 143] is limited.

To address the limitations of existing methods, we introduce Active Learning-Assisted Directed Evolution (ALDE), a computationally assisted workflow for protein engineering that employs batch Bayesian optimization. ALDE alternates between collecting sequence-fitness data using a wet-lab assay and training an ML model to prioritize new sequences to screen in the wet lab (Fig. 2.1C); it resembles existing wet-lab mutagenesis and screening workflows for DE and is generally applicable to any protein engineering objective. In this study, we use ALDE to find the ideal combination of five mutations in the active site of a biocatalyst based on a protoglobin from *Pyrobaculum arsenaticum* (ParPgb) for performing a non-

native cyclopropanation reaction with high yield and stereoselectivity. We chose this model system because the residues of interest are in close structural proximity and there is evidence of negative epistasis, which hinders DE. After performing three rounds of ALDE (exploring only $\sim 0.01\%$ of the design space), the optimal variant has 99% total yield and 14:1 selectivity for the desired diastereomer of the cyclopropane product. The mutations present in the final variant are not expected from the initial screen of single mutations at these positions, demonstrating that the consideration of epistasis through ML-based modeling is important. We solidify our argument that ALDE is more effective than DE by computationally simulating ALDE on two combinatorially complete protein fitness landscapes. We also provide an extensive analysis of the effects of protein sequence encodings, models, acquisition functions, and uncertainty quantification for protein fitness optimization, to determine best practices for real-world engineering campaigns. In short, we find that frequentist uncertainty quantification works more consistently than typical Bayesian approaches, and incorporating deep learning does not always boost performance. Ultimately, we demonstrate that ALDE is a practical and effective tool for navigating protein fitness landscapes and provide experimental and computational tools (<https://github.com/jsunn-y/ALDE>) so that the method is easy to use and broadly applicable.

2.2 Results

Practical implementation of ALDE

Broadly, ALDE alternates between library synthesis/screening in the wet lab to collect sequence-fitness labels and computationally training an ML model to learn a mapping from sequence to fitness in order to suggest a new batch of sequences to test (Fig. 2.1C), resembling batch BO. Before beginning ALDE, a combinatorial design space on k residues is defined, corresponding to 20^k possible variants. The choice of k will vary depending on the system, as larger values of k can consider a greater extent of epistatic effects (allowing for better possible outcomes) but will likely require collecting more data to find an optimal variant. First, those k residues are simultaneously mutated, and an initial round of sequence-fitness data is collected in the wet lab. ALDE is compatible with low-N, batch protein engineering settings where tens to hundreds of sequences are screened in each round. The collected sequence-fitness data are then used to computationally train a supervised ML model that can predict sequence from fitness. Different ways to encode protein sequence numerically and different types of models which can provide uncertainty

quantification are analyzed in this study. Afterward, an acquisition function is applied to the trained model to rank all sequences in the design space, from most to least likely to have high fitness. Several acquisition functions are evaluated in this study, to balance *exploration* of new areas of protein space with *exploitation* of variants that are predicted to have high fitness (Fig. 2.1D). The computational component of ALDE can be performed using the codebase at <https://github.com/jsunn-y/ALDE>. For the next round of ALDE, the top N variants from the ranking are then assayed in the wet-lab to provide additional sequence-fitness data, and the cycle is repeated until fitness is sufficiently optimized.

The active site of ParPgb is a challenging design space for standard DE

To initiate wet lab studies with ALDE, we identified a target enzymatic activity on a protein design space that would be difficult to engineer with simple DE methods. Enzyme-catalyzed carbene transfer reactions have the potential to be useful in many synthetic chemistry applications, and thus we decided to focus on the cyclopropanation of 4-vinylanisole (**1a**) using ethyl diazoacetate (**EDA**) as a carbene precursor to afford the 1,2-disubstituted cyclopropanes *trans*-**2a** and *cis*-**2a** (Fig. 2.2A). Enzyme engineering for styrenyl cyclopropanation poses a stimulating challenge for evolution toward two properties, higher yield *and* improved selectivity toward one of the diastereomers of the cyclopropane product. While this non-native chemistry has been demonstrated with cytochromes P411 [144], we decided to engineer this activity in a protoglobin. Protoglobins are archaeal hemoproteins, which are attractive engineering targets due to their high thermostability ($T_{50} \sim 60^{\circ}\text{C}$), small size (~ 200 amino acids) [145], and ability to perform novel carbene and nitrene transfer chemistries [21, 146–148]. After screening a diverse set of protoglobins, including wild-types and engineered homologs, for cyclopropanation activity, we decided to proceed with ParPgb W59L Y60Q (ParLQ) as a starting point (parent) for ALDE. Because our goal was to arrive at a variant with high yield and high selectivity for the *cis*-product, we defined the objective to be explicitly optimized as the difference between the yield of *cis*-**2a** and the yield of *trans*-**2a**. The ParLQ variant demonstrates only moderate cyclopropanation yield ($\sim 40\%$ yield) and stereoselectivity (3:1 preferring *trans*-**2a**) under screening conditions, and no known protein variant of ParPgb has high fitness for our objective.

Based on previous engineering studies using protoglobin scaffolds, we selected five active-site residues (W56, Y57, L59, Q60, and F89; WYLQF) positioned above the distal face of the heme cofactor, which display epistatic effects and are

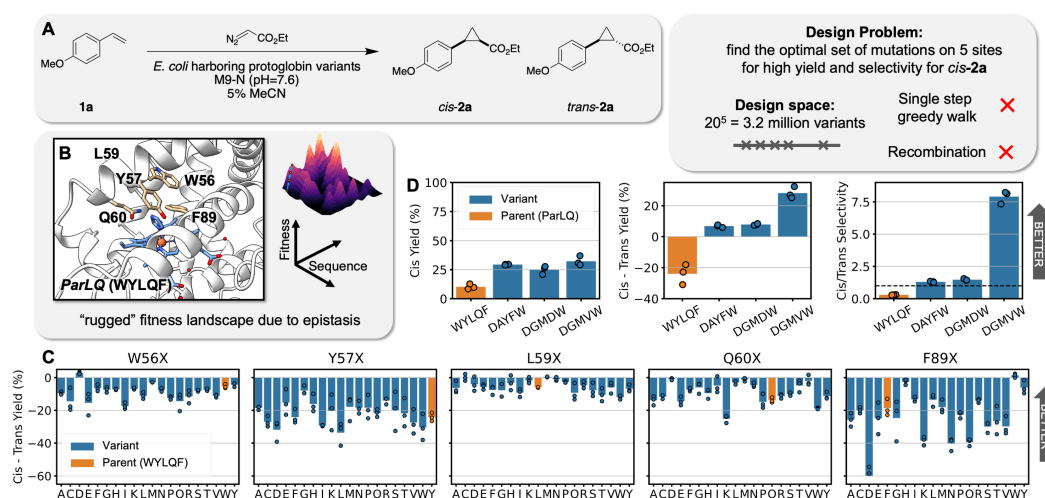


Figure 2.2: A challenging, epistatic protein design space: optimization of five active site residues in *ParPgb*. (A) Our objective was to optimize an enzyme to catalyze the formation of the *cis* product of a cyclopropanation reaction with high yield and high selectivity, which we quantify in a single value as *cis* – *trans* Yield. (B) The parent protein ParLQ is two mutations (W59L and V60Q) away from the wild-type ParPgb sequence. Five residues in the active site of ParLQ which were likely to exhibit epistasis were targeted: W56, Y57, L59, Q60, and F89. (C) The single mutations from parent at the five targeted sites do not offer significant improvements to the objective of *cis* – *trans* Yield. Very few single-mutation variants have the desired selectivity (positive *cis* – *trans* Yield), and it would not be obvious which variant to take forward in a DE campaign. Parent yields vary between runs but consistently show moderate yield and selectivity for the *trans* product. (D) Various recombinations of ideal single mutations are not effective proteins for the desired objective (*cis* – *trans* Yield), and related metrics such as *cis* Yield and *cis/trans* Selectivity. DAYFW, DGMDW, and DGMVW are the ideal combinations of single mutations naively predicted to have the highest *cis* Yield, *cis* – *trans* Yield (objective), and *cis/trans* Selectivity, respectively. Yields were measured in biological triplicate. Overall, these results suggest an optimization problem that is challenging for standard DE methods. Source data are provided as a Source Data file.

known to impact non-native activity (Fig. 2.2B) [146, 148]. Single-site saturation mutagenesis (SSM) was performed at these sites, and variants were screened by gas chromatography for their cyclopropanation products. None of the screened mutants demonstrated a significant, desirable shift in the value of the objective (Fig. 2.2C) or related metrics such as *cis* yield and *cis/trans* selectivity. Given these data, a protein engineer might opt to perform a simple recombination of all positive variants to exploit the typically additive character of mutations [149]. However, in our recombination studies of the single-site mutants with the highest fold-change in *cis* yield (DAYFW), the objective (DGMDW), or the selectivity (DGMVW), respectively, we did not observe a variant which generated *cis*-2a with high yield and selectivity (Fig. 2.2D). Overall, these findings suggest that our design problem is quite challenging for standard DE approaches.

Using ALDE to efficiently optimize ParPgb for a non-native carbene transfer reaction

With the design space confined to five residues and a well-defined objective, we began an ALDE engineering campaign. First, we synthesized an initial library of ParLQ variants which were mutated at all five positions under study (Fig. 2.3A). Mutants in this library were generated through sequential rounds of PCR-based mutagenesis methods utilizing NNK degenerate codons. We elected to use random selection from this library because we did not know if any zero-shot predictors might enrich the starting library with useful variants [50, 53]. In fact, retrospective analysis of the initial library revealed that our objective is not strongly correlated with conventional zero-shot predictors [53, 150], likely because the objective involves non-native chemistry, for which fitness is not sufficiently captured by evolutionary or stability-based metrics alone (Fig. B.1). Four 96-well plates of these random variants were picked and sequenced using the LevSeq long-read pooled sequencing method [151], yielding 216 unique variants without stop codons. Screening revealed that nearly all of the variants had higher cyclopropanation activity than free-heme background activity, likely because *ParLQ* was moderately active to begin with, and its high thermostability allows it to tolerate multiple mutations. The majority of variants displaying improved cyclopropanation yield strongly favored formation of *trans*-2a; however, several of the randomly selected sequences were capable of forming *cis*-2a in much higher yield than any previously tested ParLQ variant (Fig. 2.3B). Notably, the F89Y mutation was particularly important for inverting selectivity to favor the *cis*-2a, but only in the context of certain mutations at positions

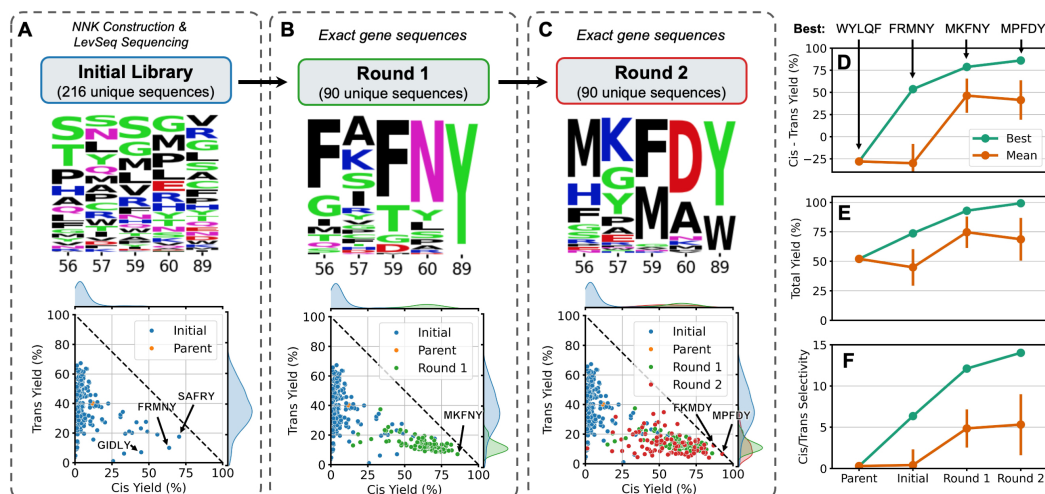


Figure 2.3: ALDE optimization trajectory on the *ParPgb* active site. The optimization campaign started with (A) constructing an initial library with mutations at all five sites under study using NNK degenerate codons, randomly selecting 384 for screening for product formation, and mapping to sequences using LevSeq. This was followed by two rounds of ALDE—(B) Round 1 and (C) Round 2. In Round 1 and Round 2, exact genes were ordered as ENFINIA DNA produced by Elegen Corp. and screened for product formation. For each round, we present the distribution of amino acids sampled at each site and the distribution of yields for the *cis* and *trans* products, with a few of the top-performing variants labeled. Overall improvement in (D) *cis* – *trans* Yield, (E) Total Yield, and (F) *cis/trans* Selectivity over several rounds of ALDE for the best variant in each round and the mean across variants in each round. The best variant in each round, defined by the objective of *cis* – *trans* Yield is labeled. Error bars indicate the standard deviation across variants in the round. Yields were measured in biological triplicate. Source data are provided as a Source Data file.

56, 57, 59, and 60.

The ALDE computational package was used to train a predictive model on sequences and labels in the initial 216-member library and to suggest sequences for testing based on our acquisition function. Based on our extensive computational simulations (described in the following section), we decided to use the DNN ensemble with one-hot encoding of the five targeted residues for model training and Thompson sampling as the acquisition function. Genes encoding the top 90 amino acid sequences, optimized for expression in *E. coli*, were prepared by exact DNA synthesis for screening (Round 1, Fig. 2.3B). Details regarding DNA sequence design are described in the included supplementary materials. Subsequent activity screening of these sequences in triplicate showed that nearly a third of Round 1

sequences met the objective better than the best variant in the initial, randomly selected set (Fig. 2.3B). The best variant in the Round 1 library, MKFNY (W56M Y57K L59F Q60N F89Y), demonstrated a total cyclopropanation yield of 93% and a *cis:trans* selectivity ratio of 12:1.

We then gave the newly collected data back to the ALDE computational algorithm for a second round of active learning. The top 90 predicted sequences were again synthesized and tested exactly as before (Round 2, Fig. 2.3C). Interestingly, the model *explored* the sequence space more in this round, as reflected in the expanded mutational diversity present in Round 2 and the increased variance in the activities of these sequences (both reaction yield and diastereoselectivity) (Fig. 2.3D–F). Impressively, the top-performing variant among these sequences (MPFDY) displayed a total cyclopropane yield of 99% and a 14:1 *cis:trans* selectivity ratio. None of the mutations in MPFDY obviously optimized the objective in the single-site mutagenesis studies (Fig. 2.2C); they work together, however, to deliver an optimal variant. Furthermore, after screening the reaction products of all predicted variants with chiral gas chromatography methods, we found that all of these sequences were generally capable of generating *cis*-**2a** in high enantiopurity.

Having concluded the ALDE-based evolutionary campaign with substrate **1a**, we sought to understand the substrate scope of the sequences explored in this project. We screened eight styrene derivatives for cyclopropanation using the sequences from Round 2 of ALDE (Fig. B.2). The variants show different yields for each of the substrates, even though some of these substrates differed from **1a** only by a single atom. Nevertheless, for every substrate, nearly all of the Round 2 variants were higher yielding and more selective for their respective *cis*- diastereomers than the parent protein, ParLQ. Interestingly, the top-performing variants for each substrate differed in sequence from MPFDY, the top enzyme for **1a** cyclopropanation. For all the predicted variants in Round 1 and 2 of ALDE, sequences were confirmed with LevSeq, and the yield and selectivity of the top variant from each round was validated in vial format, showing good overall consistency.

Computational simulations on combinatorial protein datasets support the utility of ALDE

The design choices used for the wet-lab ALDE campaign were determined by performing computational simulations on two combinatorial landscape datasets for GB1 [152] and TrpB [153]. On these landscapes, fitnesses have been measured

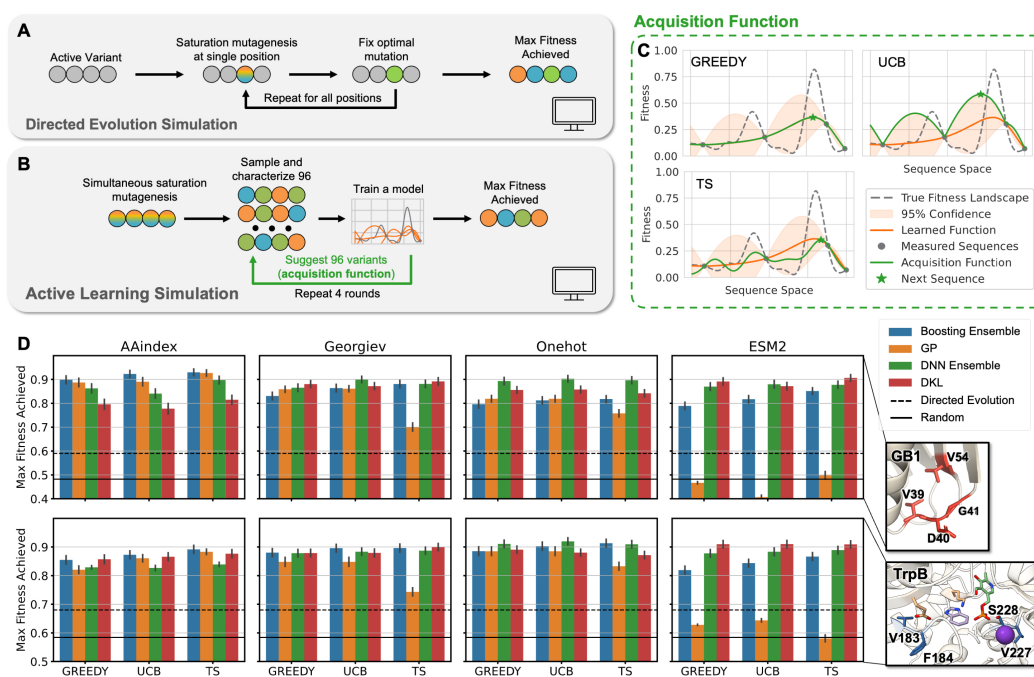


Figure 2.4: Performance of simulated ALDE campaigns on two combinatorially complete protein datasets, GB1 and TrpB. (A) Each DE simulation as a greedy single-step walk on four residues, where each residue is fixed to the optimal mutation until all four residues have been iterated across. DE simulations start from every variant that has some measurable function, with all 24 possible orderings of four residues simulated. (B) Each ALDE simulation starts from a random sample of 96 variants on the 4-site landscape, with four rounds of learning and proposing new sequences to test, each with 96 protein variants. (C) Hypothetical visualization of the three acquisition functions explored in this work: greedy, upper confidence bound (UCB), and Thompson sampling (TS). (D) ALDE for four encodings, four models, and three acquisition functions generally outperforms the average DE simulation and random sampling on the GB1 and TrpB datasets. Performance is quantified as the normalized maximum fitness achieved by the end of the ALDE campaign. Error bars indicate standard deviation across 70 random initializations. Source data are provided as a Source Data file.

experimentally for nearly all of the $20^4 = 160,000$ variants in a library where four amino acid residues were mutated to all possible amino acids. GB1 refers to the B1 domain of protein G, an immunoglobulin binding protein where fitness is measured by binding affinity-based sequence enrichment. The fitness of TrpB, the β -subunit of tryptophan synthase, was measured by coupling growth to the rate of tryptophan formation. Our baseline was DE greedy walk, where one residue was mutated to all possible amino acids, the best mutation was fixed, and the process was repeated at each of the residues under study (Fig. 2.4A). DE simulations were performed from all active variants as starting points, using all 24 possible orders to enumerate the residues under interest.

The ALDE simulation consisted of batch BO, as shown in Fig. 2.4B. In each simulation, a random batch of 96 initial samples was selected, followed by four rounds of 96 samples each, with the surrogate model retrained and proposing new samples (via the acquisition function) in each round. This simulation setup was chosen to closely imitate a real wet-lab active learning campaign. The different parameters explored for ALDE, including encodings, models, and acquisition functions, are summarized in Table 2.1. We expanded the analysis beyond Gaussian process (GP) models, which are the typical surrogate models for BO, to deep kernel learning (DKL) models [140, 141] and frequentist models based on boosting and deep neural network (DNN) ensembles. This was motivated by the rise of high dimensional encodings of protein sequences, such as those from protein language models (i.e. ESM2 [119]), which have shown utility in certain property prediction tasks [154, 155]. Visualizations of the acquisition functions (greedy, upper confidence bound (UCB), and Thompson sampling (TS)) on hypothetical models are given in Fig. 2.4C, with more details in Section A.1.

The performance of each simulated ALDE campaign was quantified as the maximum fitness achieved at the end of the campaign, normalized to the variant with maximum fitness in the design space (Fig. 2.4D). Full optimization trajectories at each iteration of the campaign are provided in Fig. B.3. We conclude that active learning can significantly outperform the average performance of DE and random sampling, and results are consistent across the two different protein datasets. ALDE is competitive with similar methods [133, 134] and also outperforms a single round of MLDE (Fig. B.4). Higher dimensional encodings (Onehot and ESM2) generally work better with deep learning-based models (DNN Ensemble and DKL), while non-deep learning models might learn better from low dimensional AAIndex and Georgiev

Encoding		Dimension per Residue		Description
AAIndex		4		Continuous fixed amino acid descriptors
Georgiev		19		Continuous fixed amino acid descriptors
Onehot		20		Categorical (which amino acid)
ESM2		1280		Learned embedding from a protein language model

Model		Bayesian?	Deep Learning?	Description
Boosting	Ensemble	N	N	An ensemble of 5 boosting models
Gaussian Process (GP)		Y	N	A collection of continuous functions by a posterior
DNN Ensemble		N	Y	An ensemble of 5 multilayer perceptrons (DNNs)
Deep Learning (DKL)	Kernel	Y	Y	A GP on the last layer of a deep neural network

Acquisition Function		Deterministic?	Description
Greedy		Y	Acquires the maximum value of the mean
Upper Confidence Bound (UCB)		Y	Maximum value of a certain confidence interval
Thompson Sampling (TS)		N	Maximum value of a random function sampled

Table 2.1: Summary of Encodings, Models, and Acquisition Functions.

parameters. The simulations further suggest that encodings from protein language models may not offer much benefit, which corroborates previous findings [53] but stands in contrast to other protein properties that can be predicted more effectively by transfer learning from protein language models [59, 154, 155]. We find that ESM2 encodings cannot be used by GPs, likely because they are too high dimensional. Other studies suggest that using the right length-scale priors with GP can enable them to work in these settings [156, 157]; while we did not observe this same effect for our application, further exploration may be interesting here. In our acquisition functions, samples in the batch were sampled independently of each other. We also explored batch expected improvement [158], but this ran extremely slowly without noticeable improvement in performance. Overall, the frequentist ensemble models perform the most consistently across different encodings.

To better understand which models are the most advantageous, we assessed how well calibrated their uncertainties were (Fig. 2.5A). For a calibrated model, an n%

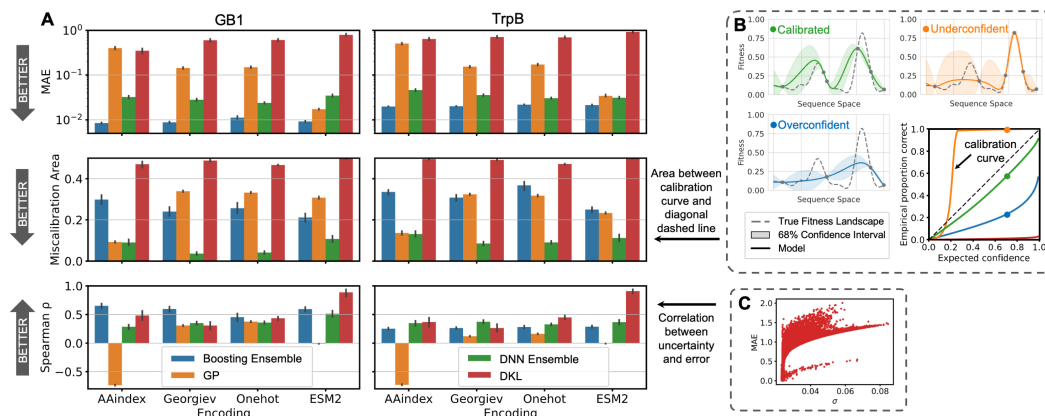


Figure 2.5: Analysis of uncertainty quantification on simulated ALDE campaigns. (A) Metrics used to evaluate how well calibrated each of the four models are for four encodings. Metrics for evaluation are the mean absolute error (MAE), the miscalibration area for the calibration curve, and the Spearman correlation between uncertainty and error. All metrics are calculated based on all measured points in the combinatorial design space. All results are based on the campaigns using UCB as the acquisition function, during the final round of the campaign. Error bars indicate standard deviation across 70 random initializations. (B) Visualizations of three hypothetical models with underconfident, calibrated, and overconfident uncertainties, and their respective calibration curves. (C) Visualization of how the Spearman correlation between uncertainty and error is calculated. Source data are provided as a Source Data file.

confidence interval should contain $n\%$ of true labels across different values of n , which can be evaluated and visualized based on a calibration curve. Hypothetical calibrated, underconfident, and overconfident models are visualized in Fig. 2.5B, with their associated calibration curves. The calibration curves for different encodings and models are given in Fig. B.5. The area between a calibration curve and perfect calibration (dashed line) is defined as its miscalibration area, which should be low. Another way to measure uncertainty calibration is by measuring the Spearman correlation between uncertainty from the model (σ) and the mean absolute error from the model (MAE), which should be high.

Overall, the Boosting and DNN Ensembles have the lowest MAEs, which suggests that they are the most accurate models (Fig. 2.5A). DNN ensembles have the lowest miscalibration areas, suggesting that they are the most calibrated and best models overall. These results are generally consistent across encodings and datasets, with a few outliers. In general, calibrated uncertainty is desirable [159, 160], and it is thought that it is important to understand how calibration shifts when extrapolating

beyond the training set [161, 162]. However, in this study, we find that performance in ALDE simulations (by max fitness achieved) is not necessarily correlated to how calibrated the uncertainties are for each model. For example, DKL performs the best for the ESM2 encoding, but these models have the least calibrated uncertainties and the highest MAEs. The poor calibration of DKL models may result from some mode collapse where out-of-distribution inputs are mapped close to the training representations by the neural network [163]. Because calibration is measured on the entire combinatorial design space, it may not directly correspond to the ability to find an optimal variant.

2.3 Discussion

Overall, ALDE is an effective method for navigating protein fitness landscapes, and it offers several advantages compared to DE. First, ALDE can unlock engineering outcomes not possible with simple DE. By considering multiple interacting positions, ALDE can search for combinations of mutations which may demonstrate desirable epistatic effects [47, 164] and reduce the risk of getting trapped at a local optimum. We demonstrated the advantage of ALDE on *ParPgb* as a particular wet-lab case study – though proof is not possible without testing every DE greedy single-step walk (which is experimentally intractable in the wet lab). Computational simulations of ALDE support this conclusion, as ALDE consistently outperforms MLDE and DE baselines. While ALDE and MLDE [53, 54] are similar conceptually and practically, MLDE only uses greedy acquisition, whereas ALDE can consider model uncertainty, which is potentially useful for exploring larger design spaces. Compared to previous computational studies on BO for protein variants [134, 137], our study examines a more comprehensive range of encodings, models, and acquisition functions, and it introduces analysis of the role of calibrated uncertainty quantification. Interestingly, we found that frequentist ensembles work the best in terms of performance and uncertainty quantification [139, 165], rather than Bayesian approaches such as typical GP models used in BO. Other ways to quantify uncertainty and improve overall performance could be explored in the future [132, 139]. Overall, classical notions of uncertainty quantification seem to play a more limited role than expected in these real-world applications. In a related study, we show that ALDE can be combined with various zero-shot predictors and that our findings here still hold for 16 different protein-fitness landscapes, including those with fewer active variants [61].

In the wet-lab engineering campaign, we were pleased to find that ALDE enabled

access to a compilation of enzymes which, when considered together, demonstrate a broader substrate scope than that of a single enzyme. By contrast, DE is limited because it often “locks” one into high yield for only a single substrate or closely related ones, as the final variant is generally a single optimized sequence. Here, we observe an emergent advantage inherent to ALDE: since sequences that balance *exploration* and *exploitation* for a given task are proposed, they can be serendipitously proficient at related tasks.

ALDE is enabled by several recent advancements in biotechnology. For the initial library constructed using degenerate codons, high-throughput sequencing was necessary to identify the sequences of variants in each well. For this work we utilized LevSeq [151, 166], a method that leverages real-time nanopore sequencing. Furthermore, rapid and reliable access to directly synthesized DNA was instrumental to the speed with which evolution was performed. The ALDE workflow was significantly enhanced with (1) the delivery of exact genes in one week, which shortened time between rounds of evolution, (2) the high fidelity of the delivered gene products meant that no sequencing was required for Rounds 1 and 2 of ALDE, and (3) no over-screening was needed because the exact sequences were arrayed individually. Overall, the time and screening cost of the wet-lab engineering campaign with ALDE was lower than for a greedy walk strategy with DE. A total of six 96-well plates were screened before arriving at a final variant: four plates of random variants, and two plates of predicted sequences within three rounds. By comparison, a greedy walk with DE would have required around five rounds of evolution—typically one mutation is accumulated in each round of a DE greedy walk—with increased screening in the later rounds, which would require greater experimental resources such as chemical reagents and analysis time. We expect that exact gene synthesis will be increasingly important for powering active learning workflows in protein engineering.

In this work, we illustrated ALDE’s power for simultaneously increasing the activity and selectivity of an enzyme for a non-natural reaction, but ALDE is a general workflow that can be used for a broad range of protein engineering applications. Additionally, ALDE could be integrated into robotic systems for automated and efficient protein engineering workflows, and library design could utilize tools such as DeCOIL [167]. While we only engineered on five residues in this study, ALDE should naturally extend to even larger design spaces on more residues, as long as assay-labeled data is collected on variants with mutations spread across those

residues. Determining the residues to target is an open challenge: these residues should tolerate mutations and have the potential to increase the fitness of interest. Initial domain knowledge, evolutionary conservation, or initial mutational screening may be useful here. Library design could also benefit from limiting the number of simultaneous mutations or using zero-shot scores [53, 64]. Despite our *in silico* simulations using combinatorially complete datasets and wet-lab demonstration of ALDE, it remains an open question how generally our findings can be applied to the engineering of other enzymatic systems. Further work is needed to understand how the number of samples and/or rounds required to achieve successful engineering outcomes will increase (linearly or exponentially) with the number of sites explored simultaneously and how epistasis affects this. Future work here may also involve generative modeling if it is not possible to enumerate the acquisition function on the entire design space. Overall, accompanied by a user-friendly codebase, ALDE is a broadly applicable tool that can unlock more efficient and effective protein engineering.

Chapter 3

COMPUTATIONAL CLASSIFICATION AND RETRIEVAL OF ENZYME STARTING POINTS

- (1) Yang, J.; Mora, A.; Liu, S.; Wittmann, B. J.; Anandkumar, A.; Arnold, F. H.; Yue, Y. *Advances in Neural Information Processing Systems* **2024**.

Abstract Enzymes are important proteins that catalyze chemical reactions. In recent years, machine learning methods have emerged to predict enzyme function from sequence; however, there are no standardized benchmarks to evaluate these methods. We introduce a benchmark and dataset suite for the Classification And Retrieval of Enzymes (CARE). CARE centers on two tasks: (1) classification of a protein sequence by its enzyme commission (EC) number and (2) retrieval of an EC number given a chemical reaction. For each task, we design train-test splits to evaluate different kinds of out-of-distribution generalization that are relevant to real use cases. For the classification task, we provide baselines for state-of-the-art methods. Because the retrieval task has not been previously formalized, we propose a method called Contrastive Reaction-Enzyme Pretraining (CREEP) as one of the first baselines for this task and compare it to the recent method, CLIPZyme. CARE is available at <https://github.com/jsunn-y/CARE/>.

3.1 Introduction

Proteins, which are sequences of amino acid building blocks, are not only integral components of all living organisms, but also important for a myriad of commercial applications spanning from the health domain to the bio-economy. Enzymes are a subclass of proteins that can catalyze chemical reactions, and they have many applications in areas such as bioremediation, plastic degradation, gene editing, and drug synthesis [1, 2, 4, 13].

Identifying the specific chemical reactions that an enzyme is capable of performing (*i.e.*, the enzyme’s function) is a key first step for many applications [168]. While hundreds of millions of proteins have been sequenced, less than 1% are annotated with function [66]. For instance, during standard metagenomic analyses, enzyme genes are annotated for their functions, which enables determination of an organism’s critical metabolic pathways and specialization [169–171]. When applied to

chemical synthesis, enzyme annotations are needed to identify catalysts that can replace existing step(s) in drug synthesis procedures—known as retrobiosynthesis [76]. Likewise, enzyme engineering for new-to-nature function [172] involves discovering an enzyme starting point with a desired function before improving activity for that function using protein engineering techniques [16, 50]. Historically, similarity search algorithms, most notably BLAST, have been the most common methods used to assign function to protein sequences [173, 174]. These methods work by finding similar sequences in annotated reference databases, as similar sequences are likely to share function. More recently, searching based on structural similarity (Foldseek) has also demonstrated utility for assigning function [175, 176]. However, up to one third of genes from bacterial genomes cannot be assigned function using existing methods [169]. As such, there is a need for more abundant, high-quality annotations of enzymes and automated workflows to identify enzymes with desired functions.

In recent years, there has been increasing interest in using machine learning (ML) for a broad range of applications related to the functional prediction and design of enzymes [50, 177, 178]. In particular, ML models have emerged to classify protein sequences based on their function, which are reviewed here [179] with a few examples listed here [68, 75, 171, 180–183]. Despite these advances, there is no standard benchmark or dataset for evaluating computational models for enzyme function prediction [184]. A challenge associated with classification is that a given enzyme is often able to perform multiple reactions [10], and many reactions are not annotated. Moreover, complex tasks such as extrapolation to unannotated reactions [185] have yet to be evaluated.

In this work, we present a benchmark suite for the **Classification And Retrieval of Enzymes (CARE)**. Our contributions can be summarized as: **(1)** formalizing model evaluation into two tasks that encompass applications relevant to scientists and engineers: classification of an enzyme by function (Task 1), and retrieval of enzyme sequences based on a reaction (Task 2); **(2)** curating high-quality and easy-to-use datasets; **(3)** providing train-test splits that mimic challenging extrapolations in real-world use cases; and **(4)** benchmarking state-of-the-art models for Task 1, and providing a new method that serves as a baseline for Task 2. Because Task 2 has been minimally explored, we introduce a model called **Contrastive Reaction-Enzyme Pretraining (CREEP)** to serve as a baseline for text, reaction, and sequence integration and compare it to an existing approach (CLIPZyme) that uses sequence and reaction, in addition to other retrieval approaches. CREEP can

perform contrastive learning across three different modalities (protein, reaction, and textual description), and the learned representations are then used for retrieval. Overall, we anticipate that CARE will be a useful and easy-to-use resource for ML researchers to benchmark their enzyme function prediction models.

3.2 Related Work

Datasets. Various databases have emerged to help researchers store, share, and identify functionally annotated enzymes. Protein sequence databases such as UniProt [66] and Pfam [186] are catalogs of annotated protein sequences. While most databases reference protein sequences, increasingly, these sequences can be linked to protein structures, either experimentally validated, as in the Protein Data Bank (PDB), or via structural prediction tools [187] in databases such as the AlphaFold Database [188]. BRENDA [189] is a curated database specific for reaction and enzyme sequence information. Rhea [190] consolidates information from BRENDA, and other sources, such as pathway databases including KEGG [191]. There is ongoing work to compile and clean/standardize reactions from multiple databases, namely ECRReact [192] and EnzymeMap [193]. Related to these databases, there are retrosynthesis planning tools [194–196], and Selenzyme is tool to retrieve enzymes to perform a target reaction [197].

Protein Benchmarks. Our work takes inspiration from existing protein fitness prediction benchmarks, where fitness is a quantification of some function. TAPE [154] and FLIP [198] evaluate representations from protein language models for the prediction of a broad range of general and specific protein properties (stability, secondary structure, binding etc.) ProteinGym considers sequence variant effect prediction by using likelihoods from these language models [199]. While benchmarks for protein fitness prediction tasks are well defined, there are no standardized benchmarks for protein function prediction. Fitness is a numerical quantification of protein function (stability, enzyme activity level, etc.), but function is more qualitative/categorical, e.g. a description of an enzymatic reaction associated with a protein. Here, we focus on proteins that perform chemical reactions, enzymes.

Classification of Enzyme Function. ML models have emerged to predict the outcomes of enzymatic reactions [200–202] and for classification of enzyme function. Enzyme function is usually expressed using enzyme commission (EC) numbers, which is a hierarchical scheme for classifying enzyme function into classes (families) and consists of four levels of descriptions (Fig. 3.1A). Some classification

models are general protein function prediction models, which encompass all proteins, not just enzymes, such as ProtCNN/ENN [68] and ProteInfer [182]. Many models utilize representations from protein language models [119, 155, 203], and others incorporate protein structure as information, such as DeepFRI [181] and BioCLIP [204]. Methods related to supervised contrastive learning [205] have been particularly useful here, such as CLEAN, HiFi-NN, and Enzhier [75, 171, 206], likely by reducing imbalances in the number of sequences representing each EC number. Recently, enhanced approaches have enabled function prediction in ProtEx [183], PhiGnet [207], and others [208–210]. Other retrieval tools enable more sensitive detection of homologs such as DHR [211], ProtTrek [120], and using structure [176, 212, 213]. Finally, there are related models that predict substrates for enzymes [214, 215] and that aim to learn connections between chemical space and protein space [216].

Large Language Models. Recently, there has been an explosion in pretrained models in the biological and chemical domains, particularly large language models (LLMs) [217, 218]. For example, ChatGPT is capable of answering questions related to general scientific knowledge. These language models can be further finetuned for applications such as answering questions about protein sequences (including Pika [219], InstructProtein [220], InstructBioMol [221], ProteinGPT [222], and ProteinChat [223]) and for reaction synthesis planning (ChemCrow [224]), among others. LLMs present important benchmarks for enzyme functional classification and retrieval, given the widespread adoption of LLMs as science facilitators and their ease of use (in particular webserver-based approaches), compared to domain-specific methods [225].

Multimodal Contrastive Learning. Contrastive learning is an efficient and effective pretraining paradigm that aligns positive pairs and contrasts negative pairs simultaneously. The design of these pairs depends on the specific tasks, such as using data augmentations of the same image [226] or considering the topology and geometry of molecules [227]. More recently, contrastive learning has shown success in aligning the representation space of different biological and chemical modalities, *e.g.*, text and chemical structure alignment in MoleculeSTM [228], text and protein sequence alignment in ProteinDT [229] and ProteinCLIP [230], reaction structure and protein structure alignment in CLIPZyme [200], and protein sequence and structure alignment in BioCLIP [204] and with text in ProTrek [120], among others [217, 231]. Cross-modal alignment in the representation space has been shown to

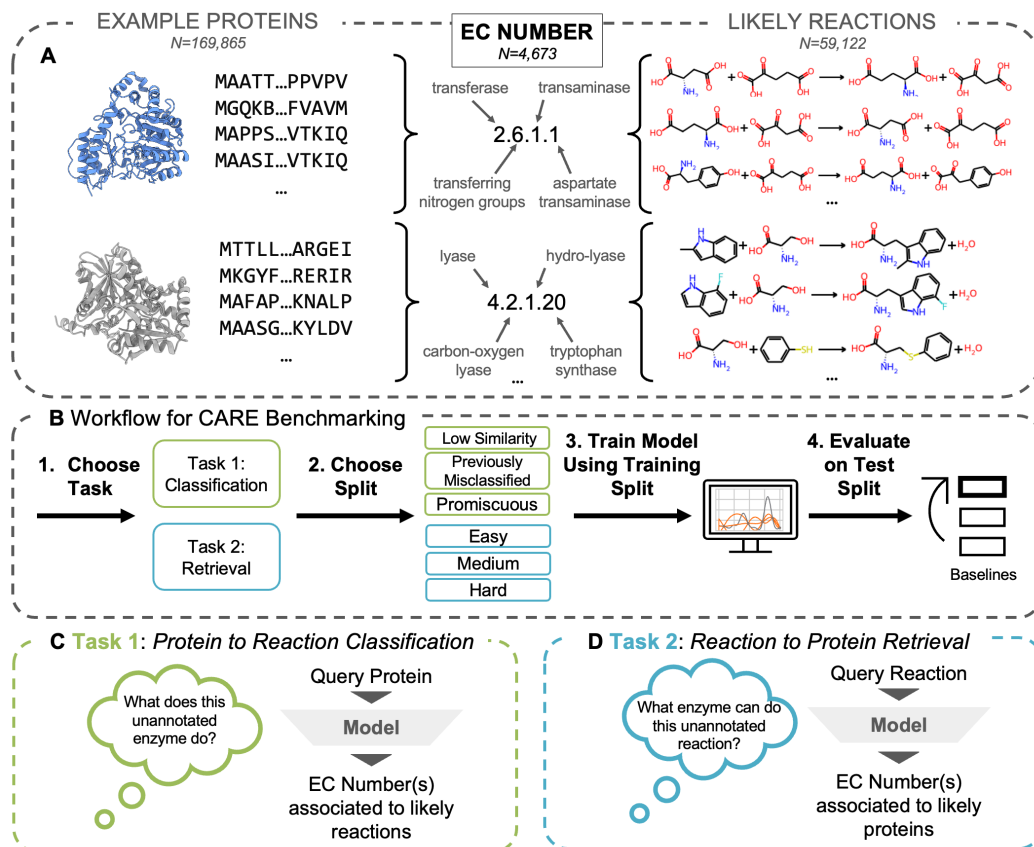


Figure 3.1: **Overview of CARE.** (A) Dataset format for CARE, showing examples of enzymes and their associated reactions. The EC number acts as a bridge between a protein sequence and the reactions it is likely to perform. The EC number is a hierarchical classification scheme for enzyme function with four levels of description, with increasing specificity from left to right. (B) General workflow for CARE benchmarking. (C) Task 1 is a real-world use case for enzyme classification based on a protein sequence. (D) Task 2 is a real-world use case for enzyme retrieval based on a reaction.

improve generalizability and improve performance on challenging tasks, such as out-of-distribution learning, zero-shot learning, and text-guided molecule design and optimization [229]. Consideration of multiple modalities may be especially important for the prediction of qualitative functions.

3.3 Overview of CARE

Though there are many studies using ML models to perform enzyme classification based on EC numbers (Fig. 3.1A), there is no standardized benchmark to evaluate how well these models generalize to unseen protein sequences. To address this need, we present a benchmark suite for the classification and retrieval of enzymes (CARE,

Fig. 3.1B). CARE formalizes classification of an enzyme sequence by EC number as "Task 1" (Fig. 3.1C). For this task, we design train-test splits of protein sequences to test out-of-domain generalizations that are relevant to real-world use cases. In addition, CARE addresses another key limitation of current studies: classification is limited to EC numbers, which is a closed vocabulary of functions (reactions), so existing models cannot generalize to unannotated reactions. Thus, we introduce an entirely new task, retrieval of an EC number given a reaction, which we call "Task 2" (Fig. 3.1D). For this task, we design train-test splits to evaluate how well models can generalize to out-of-domain reactions, ensuring that the splits pose different levels of difficulty.

To streamline benchmarking, we curate a dataset of enzymes, reactions, and their associated EC numbers for CARE. At a high level, we build two datasets, one that links protein sequence to EC and one that links reaction to EC (Fig. 3.1A). The former is processed from Swiss-Prot, the validated portion of UniProt [66] and filtered to protein sequences between length 100 and 1024 with annotated EC number(s). The latter is formed as a combination of EnzymeMap [193] and ECReact [192], where ECReact is only used to supplement EC numbers that are missing in EnzymeMap. Our workflow for generating the datasets used in this work is explained in detail in Section A.2 and shown visually in Fig. A.1.

The overall workflow for benchmarking using CARE is shown in Fig. 3.1B. For each task, domain-specific train-test splits are provided from the processed datasets. Model training can use any of the data in the train split, and each model is evaluated on the associated test split. In the rest of this study, we explain the specific design choices used to generate train-test splits and analyze benchmarking results of state-of-the-art methods on these splits. The curated datasets and splits used in CARE can be accessed at <https://github.com/jsunn-y/CARE/>.

3.4 Task 1: Enzyme Classification

Task 1, classification of an enzyme sequence, tests the ability of a model to extrapolate to unseen protein sequences. Task 1 is a fairly well studied task [179], but model evaluation has not been previously standardized as there are many factors to consider, such as the distribution of sequences and functions in the test sets. Task 1 applies to use cases where a scientist is given an unannotated enzyme sequence and seeks to understand the enzymatic function associated with that sequence (Fig. 3.1C), for example for metagenomic analysis or finding new enzymes for retrobiosynthesis.

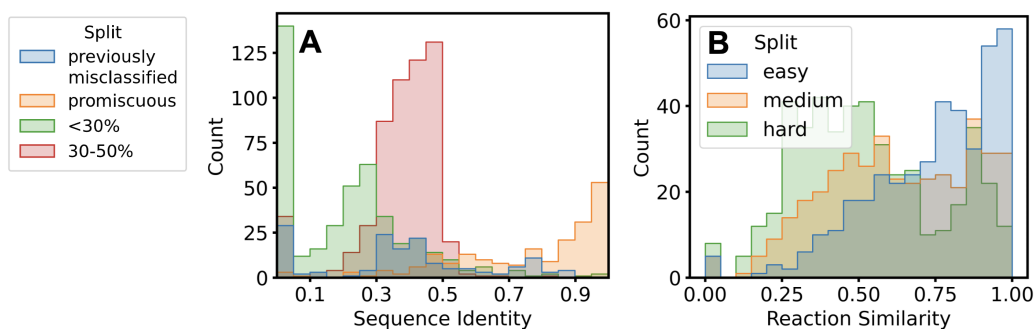


Figure 3.2: Distribution of similarities between samples in each test set and the corresponding train set. (A) Protein sequence identity (Task 1) was measured to the closest hit in the train set using BLASTp. Sequence identity can be thought of as normalized Levenshtein distance. **(B)** Reaction similarity (Task 2) was measured as cosine similarity with each reaction represented using DRFP [232]. Reaction similarity was measured from the test set to each train-set cluster center of reactions belonging to an EC number.

Table 3.1: Summary of train-test splits used in Task 1. For Task 1, certain protein sequences are held out. Train "samples" refers to the number of unique protein-EC data pairs, and test "samples" refers to the number of protein sequences. All splits for Task 1 share the same protein train set.

Split name	Description	Train samples	Test samples
<30% Identity	Held-out sequences with approximately < 30% identity to the training set	184,529	432
30-50% Identity	Held-out sequences with approximately 30-50% identity to the training set	184,529	560
Previously Misclassified (Price)	Previously misclassified enzyme sequences from Price et al.	184,529	148
Promiscuous	Held-out sequences with multiple EC numbers	184,529	209

With the emergence of conditional generative models for protein sequences, it is also important to have high-throughput computational methods that can predict the function of generated sequences [89, 90, 93, 94, 125]. For this task, a query protein is passed through a trained model to predict an EC number. The EC number is then associated with likely reactions that the protein will be able to perform.

Splits for Task 1. Task 1 can be framed as evaluating how models generalize toward unseen sequences with different types of difficulty, visualized in Appendix Fig. A.3A. The train-test splits for Task 1 are summarized in Table 3.1.

- *<30% and 30-50% identity* splits: these two test splits contain sequences with sequence identities falling in the respective range, to sequences in the training set by using clustering (Section A.2). Sequence identity is related to the normalized Levenshtein distance between two sequences. Natural sequences with high (>40%) sequence identity at the protein sequence level are likely to share function [233]. It is expected that the lower the sequence identity, the more difficult it is to assign functional annotation.
- *previously misclassified (Price et al.)* [169] split: challenging to assign because some have low sequence identity to other proteins, and many may lie near "activity cliffs" (Fig. A.3), a region where function can change sharply in sequence space.
- *promiscuous* split: in this study, we define promiscuous enzymes as those mapping to multiple EC numbers (thus lying on multiple activity peaks). These enzymes are particularly interesting for enzyme engineers, as new-to-nature activity can be found in between activity peaks [50].

For Task 1, the training split is constructed by holding out all of the pooled sequences in the test splits. The classification output vocabulary (EC numbers) is closed rather than open, so EC numbers in the test sets are present in the train sets. We verify that sequences in the test sets generally fall within the expected sequence identities, relative to the training set (Fig. 3.2A). Notably, the *Price* test set has a wide distribution of sequence identities, while the *promiscuous* test set has high sequence similarity to the training set. The sequences in the test splits are distributed generally evenly across all different EC numbers (Fig. A.4 in Appendix). More details can be found in Appendix Section A.2.

Task 1 benchmarking results. Benchmarking results for Task 1 are summarized in Table 3.2. We start with two methods as baselines, classification using a random order of EC numbers (Random), and Diamond BLAST at the protein sequence level, herein referred to as BLASTp, which is a workhorse bioinformatics tool that performs local-alignment to determine the most similar sequence(s) given a target

Table 3.2: **Performance of various methods on Task 1.** Performance is measured as k=1 classification accuracy (%). However, for the promiscuous split, we use k=(number of true ECs) for Random and CLEAN and report the average accuracy across all true EC numbers. For this split, we did not evaluate ChatGPT and Pika, as we prompted for only a single EC in the response. For BLAST and Foldseek, we use all EC numbers associated to the top single hit. More details are provided in [Section A.2](#). CLEAN is a state-of-the-art method at the time of publication. Other methods such as ProteInfer, HiFi-NN, Enzhier, ProtEx, PhiGnet, etc. could also be benchmarked here.

Split	Method	Level 4 Accuracy (X.X.X.X)	Level 3 Accuracy (X.X.X.-)	Level 2 Accuracy (X.X.-.-)	Level 1 Accuracy (X.-.-.-)
<30% Identity	Random	0.0	1.2	3.2	19.4
	BLASTp	51.4	60.0	62.5	65.7
	Foldseek	54.9	68.3	72.9	80.6
	ChatGPT	0.0	0.0	1.6	28.9
	Pika	20.6	37.7	46.1	61.6
	CLEAN	55.1	68.8	74.8	84.5
30-50% identity	Random	0.0	0.7	3.6	22.5
	BLASTp	81.1	87.9	90.7	92.3
	Foldseek	79.8	87.1	90.5	95.0
	ChatGPT	0.0	1.4	3.0	34.8
	Pika	37.7	50.2	60.0	73.8
	CLEAN	80.2	88.0	91.6	95.5
Previously Misclassified (Price)	Random	0.0	0.7	4.7	22.3
	BLASTp	35.1	70.9	78.4	78.4
	Foldseek	41.2	82.4	93.2	96.6
	ChatGPT	0.0	9.5	17.6	37.2
	Pika	4.1	50.7	64.9	82.4
	CLEAN	31.8	74.3	81.8	85.8
Promiscuous	Random	0.5	4.1	9.0	41.1
	BLASTp	93.7	94.8	95.2	95.9
	Foldseek	88.0	90.9	92.4	94.4
	CLEAN	69.4	77.9	81.5	87.0

query and a database [173]. Additionally, we include search by structural similarity using Foldseek [175].

While there exist many ML tools to directly perform EC number classification, CLEAN [75] and several others [171, 183, 206, 207, 210] are a few that seem to report the current state-of-the-art with comparable performance. Some of the

latter methods were not yet publicly available at the time of this study, and other recent retrieval methods would also be promising approaches to test, but we opted to focus on benchmarking on CLEAN here. CLEAN generally performs the best for enzymes with low sequence identity to known sequences, but the BLASTp and Foldseek baselines perform similarly. ChatGPT was also tested, but it appeared to often hallucinate as it was forced to provide an answer, with results similar to the random baseline. Pika seems to bridge the gap between standard LLMs and enzyme classification, but its performance is not as good as standard tools like BLASTp [219].

Overall, these results suggest that there is still room to improve classification of protein sequences with low sequence identity (<30%) and that lie near multiple activity peaks (*Price*). In fact, Foldseek has the best performance on the *Price* split, suggesting that structure can be conserved, even in scenarios where sequence is misleading. The *promiscuous* split is not easy for all methods; even though the test sequences have high sequence identity to the train set, CLEAN sometimes misses EC numbers. Interestingly, BLASTp and Foldseek perform close to the state-of-the-art, even when compared to more complex ML models. Future models could take advantage of this finding to augment training, as ProtEx does [183]. Additional details on the implementation of each method can be found in Appendix Section A.2.

3.5 Task 2: Enzyme Retrieval

Task 2, enzyme retrieval from a query reaction, tests the ability of a model to extrapolate to unseen reactions. Task 2 has not been formalized or explored in previous studies, but it is equally as important as Task 1, as it applies to a use case where a scientist or engineer is seeking to identify a previously characterized enzyme sequence that can perform a novel (unannotated) reaction (Fig. 3.1D). Typical applications include: an environmental engineer looking for an enzyme to degrade a toxic pollutant [234], an enzyme engineer looking for an enzyme to catalyze a selective reaction for drug synthesis [235, 236], or a gene annotator identifying the gene for an "orphan" enzyme with known function but unknown sequence [237]. For Task 2, a query reaction is passed through a trained model to perform retrieval to an EC number and its associated proteins that are likely to be able to perform that reaction.

Splits for Task 2. Task 2 aims to evaluate how well a model generalizes to unseen reactions with different levels of difficulty. The train-test splits for Task 2

Table 3.3: **Summary of train-test splits used in Task 2.** Certain reactions are held out, and "samples" refers to the number of reaction-EC pairs.

Split name	Description	Train ECs	Train samples	Test samples
Easy	Certain reactions are held out, sampled uniformly across ECs, but no EC numbers are held out. The test set is the same as the holdout set.	4,960	61,373	393
Medium	All reactions corresponding to certain ECs are held out, at EC level 4 (X.X.X.X). Test set reactions are sampled uniformly across ECs from the holdout set.	4,748	57,691	393
Hard	All reactions corresponding to certain ECs are held out, at EC level 3 (X.X.X.-). Test set reactions are sampled uniformly across ECs from the holdout set.	3,052	35,252	460

(*easy*, *medium*, and *hard*) are summarized in Table 3.3 and visualized in Appendix Fig. A.3B. We equate greater difficulty with a more challenging train-test split; in a harder set, the test reactions are less similar to reactions in the corresponding train set. We decide similarity based on the amount of overlap in EC number (e.g., a reaction from 4.2.1.20 is considered more similar to another one in 4.2.1.20 than one from 4.2.1.1).

- *easy* split: EC numbers are randomly sampled at EC level 4 (X.X.X.X) and then randomly mapped to reactions, which are held out as the test set.
- *medium* split: the same reactions are used for testing as the easy set, but all other reaction-EC pairs which share the same EC level 4 (X.X.X.X) are held out from training.
- *hard* split: random EC numbers are sampled at EC level 3 (X.X.X.-) and all reactions under that EC3 are held out from training, while a subset of reactions from the held-out EC numbers are used for testing.

The sequences in the test splits are distributed generally evenly across different EC numbers (Appendix Fig. A.4). From *easy* to *medium* to *hard*, the test set

reactions also become more dissimilar to their respective training sets (Fig. 3.2B). Reaction similarity was quantified by DRFP, which is a reaction representation that uses set differences between product and reactant fingerprints and has demonstrated solid performance without requiring model training [232]. Overall, Task 2 is more complex than Task 1, because unlike Task 1, entire EC numbers are held out from the training set (i.e., the classification output vocabulary is open rather than closed), so multiple modalities must be considered to link the unseen reactions to their respective EC numbers. More details on splitting can be found in Appendix Section A.2.

CREEP baseline method. There is a lack of models that have been tested for their ability to generalize beyond annotated reactions and none that have considered reaction, text and sequence together, so we develop a contrastive learning method for this task, called **Contrastive Reaction-Enzyme Pretraining (CREEP)** (Fig. 3.3). Our approach is related to CLIPZyme [200], which uses contrastive alignment of reactions represented as 2D graphs and protein structures represented as 3D graphs. Somewhat differently, CREEP leverages finetuning of pretrained language models, rxnfp [238] and ProtT5 [203] to learn aligned representations of reactions and proteins, respectively. Rxnfp is a BERT-style [239] language model that was trained on reactions represented as SMILES/SMARTS strings and has demonstrated state-of-the-art performance on reaction type classification [238]. ProtT5 is a T5 protein language model that has been used for prediction of various protein properties and demonstrates similar performance to ESM [119, 203]. We used these language models for simplicity and their ease of finetuning. Uniquely, CREEP can learn on a third modality (CREEP w/ Text): textual description of the EC number based on gene ontology [240], which is encoded using SciBERT [241]. The use of text has recently been shown to boost function prediction tasks [242]. More details on CREEP training can be found in Appendix Section A.2.

Task 2 benchmarking results. Benchmarking results for Task 2 are summarized in Table 3.4. We start with two methods as baselines: randomly guessing a ranking of EC numbers (Random) and finding the most similar reaction in the train set to the test query (Similarity Baseline). For the Similarity Baseline, we used DRFP to represent chemical reactions [232]. Details on the downstream multimodal retrieval process can be found in Appendix Section A.2 and Fig. A.4. Methods like CREEP and CLIPZyme retrieve reference protein sequences based on a query reaction, and those proteins are mapped to a ranking of retrieved EC numbers; thus, they can

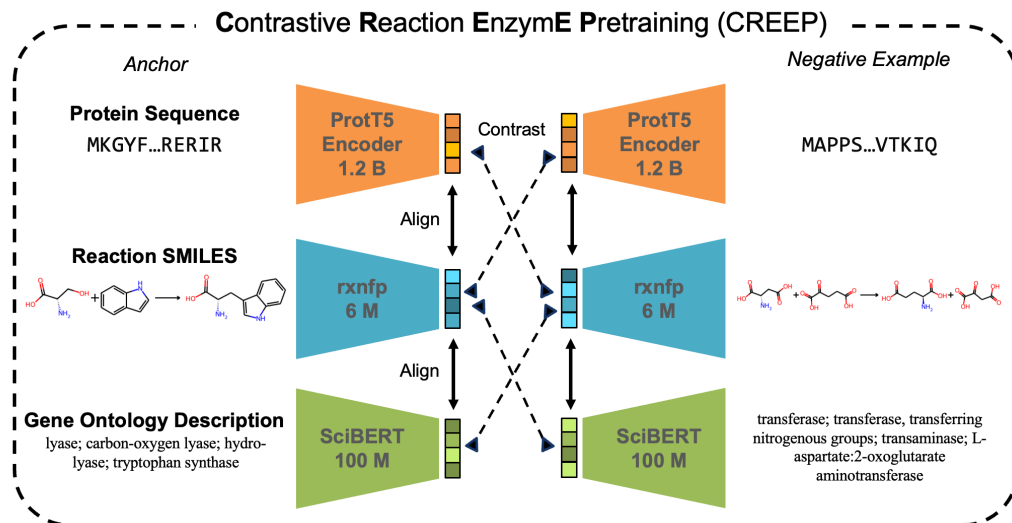


Figure 3.3: **Model architecture for CREEP.** CREEP aligns reaction and protein sequence using contrastive learning to perform downstream retrieval between domains. Optionally, CREEP training can be augmented by using textual description as a third modality to bridge the other two. Model sizes by number of parameters are also listed.

generalize to EC numbers that are not linked to reactions in (i.e. missing from) the training set.

Overall, the Similarity Baseline is quite strong and performs much better than Random. ChatGPT was prompted with reactions written using compound IUPAC names rather than SMILES strings, both with (ChatGPT w/ Text) and without (ChatGPT) textual descriptions from gene ontology [240]. We only show ChatGPT performance for the easy split, as ChatGPT is not a good measure of generalization on the medium and hard splits. These both require "leaving out" entire EC numbers before training to ensure there is no data leakage.

On the more difficult test sets, CREEP offers an advantage compared to the Similarity Baseline, and particularly when combined with textual description: CREEP (w/ Text). This suggests that utilizing protein sequence information as a modality is useful. Still, performance on the harder splits is weak across the board, which suggests that there is significant room for future improvement, although retrieval of EC class may not be the ideal metric for out-of-distribution reaction classification. We anticipate that contrastive alignment with textual descriptions will play an increasingly important role in enzyme retrieval [217, 229, 230], and there is opportunity for better curation of these descriptions. Alternatively, CLIPZyme [200]

Table 3.4: **Performance of various methods on Task 2.** Performance is measured as k=1 retrieval accuracy (%). *denotes that there may be data leakage causing performance to be inflated. Bolded accuracy is the best model.

Split	Method	Level 4 Accuracy (X.X.X.X)	Level 3 Accuracy (X.X.X.-)	Level 2 Accuracy (X.X.-.-)	Level 1 Accuracy (X.-.-.-)
Easy	Random	0.0	1.0	4.6	22.9
	Similarity Baseline	59.3	77.1	85.2	90.6
	ChatGPT*	4.8	22.6	43.5	71.0
	ChatGPT (w/ Text)*	13.7	56.7	81.4	98.5
	CLIPZyme	12.2	39.9	61.8	79.9
	CREEP	39.4	66.4	79.9	92.9
	CREEP (w/ Text)	60.3	89.3	93.9	96.7
Medium	Random	0.0	0.5	4.1	17.0
	Similarity Baseline	0.0	40.2	55.7	73.3
	CLIPZyme	2.0	26.0	46.6	69.0
	CREEP	4.1	44.3	63.1	86.5
	CREEP (w/ Text)	7.4	59.5	75.6	92.1
Hard	Random	0.0	0.9	1.5	18.3
	Similarity Baseline	0.0	0.0	13.5	42.0
	CLIPZyme	1.1	4.1	13.5	46.7
	CREEP	1.3	4.8	18.7	57.6
	CREEP (w/ Text)	1.3	9.8	22.2	57.2

represents reactions and proteins as graphs, but performance is lower, potentially due to some sample loss from some data being unable to be processed to graphs during training and inference. Here, performance could likely be improved by optimizing training hyperparameters. Additional details on the implementation of each method are in Appendix [Section A.2](#), and additional benchmarking results are presented in [Fig. B.6](#).

3.6 Discussion

We made an important design choice in the CARE benchmarks: to perform classification and retrieval at the coarse-grained level of EC numbers, which could be a limitation. Enzymes can often perform many reactions, meaning it is an acceptable assumption that enzymes belonging to the same EC number will share the capacity to perform similar reactions, even if they are not directly annotated for all reactions. The ultimate task in this domain is to perform direct reaction to protein sequence retrieval and vice-versa (as in CLIPZyme [200] and Reactzyme

[243]). However, currently the data for validation is limited, and we believe that many negative examples in those datasets may actually be feasible protein-reaction pairs. As experimentalists obtain higher resolution annotations of proteins and their associated reactions, this will become more realistic. Furthermore, our goal was to provide benchmarks for a broad range of functions, but model evaluation could also be performed on more specific classes of enzymes with direct protein-reaction pairs, as explored by EnzymeCAGE [244]. Other ways to hold out proteins with difficult-to-predict functions could also be explored. Some EC numbers are also incorrectly annotated, which are discussed in more detail here [214].

In the future, the proposed train-test splits could be refined for both tasks. For the Task 1 splits, there is some leakage in the sequence identity, with some sequences in test sets lying outside of the enforced sequence identity ranges, likely due to the different sequence similarity algorithms used (MMseqs2 vs Diamond BLAST). MMseqs2 utilizes cascaded clustering which pre-filters based on initial clusters in the target set [245], while BLAST attempts to identify the closest sequence within the specified set based on local similarity. Another limitation is that while there are no duplicate reactions present in both the train and test sets, some of the test reactions are very similar to reactions in the training set despite having different ECs. It would be beneficial to do a more detailed analysis of reaction similarities and explore other representations of reactions to understand which reactions can be considered equivalent. Many EC classes also involve multi-complex enzymes; in other words, certain subunits of these enzyme are not actually performing catalytic activity. Future work could filter out some of the non catalytically active subunits or act to specifically predict the catalytic subunit, or the entire complex based on a reaction. Over time, the train-test splits should be updated as additional functional annotations are acquired and compiled in databases. For example, while over 36 million sequences in BRENDA/UniProt have EC numbers associated with them, these are often detected using homology based models, which may incorrectly assign EC numbers. A more detailed analysis of the specific failure modes of these ML models could be valuable future work [185].

We opted to assess performance using accuracy due to its simplicity and interpretability, but other retrieval and virtual screening metrics such as BEDROC and enrichment could also be explored. For the promiscuous enzymes with multiple EC numbers, we reported accuracy averaged across all of the true labels, but other classification metrics such as precision and recall should be considered in future

work. The number of retrieved ECs could also be chosen using strategies like max separation and p-value as implemented in CLEAN [75].

There is also significant opportunity to use other modalities such as textual description and protein structure for more effective representations of enzymes and reactions, or to improve the text-based annotations. Gene context is also useful for annotating protein function and could be incorporated into future benchmarks [246]. The textual annotations used in this work are direct textual references to the EC class, meaning that the models in task 2 that incorporate text (e.g. CREEP), are effectively learning the relationship between the EC as a word and the EC as a number, in addition to the similarity of the reaction. Other tools such as Pika which use multiple types of textual description extend this to learn relations about the enzyme, and could be used in future iterations of CREEP. Future work will involve incorporating structure and graph based representations [247] into CREEP, similar to those used in CLIPZyme [200]. We also plan to do a more detailed analysis of the representations learned by CREEP. The addition of textual description in CREEP potentially introduces indirect data leakage between the train and test sets, so future iterations of CARE will need to consider this. Future work should also consider how to include protein function prediction models that go beyond enzymes—or models that would like to use additional data/modalities [120, 248]—into the CARE evaluation framework. LAB-Bench provides a framework for evaluating scientific reasoning using language, which will increasingly intersect with CARE [225]. A major bottleneck we encountered was that many models, including language models, were not available or difficult to use, limiting our ability to include them as benchmarks. There is also room to improve the prompt engineering of language models which could be further explored to enhance the performance of these models.

Conclusion

Predicting the functions of enzymes is important for many applications ranging from gene annotation to enzyme engineering. While many models exist to classify enzyme function via EC numbers, there are no standardized benchmarks for evaluation of these models. Furthermore, no existing models have been tested for generalization beyond annotated reactions. To address this need, we introduce CARE, which is a benchmarking suite to formalize model evaluation for these two tasks. We also present CREEP, a model which uses multimodal contrastive learning and is one of the first models that can perform the latter task. We encourage developers to

integrate their current and future methods or benchmarking results into the CARE Github repository (<https://github.com/jsunn-y/CARE/>). Overall, CARE is an important tool for encouraging progress in enzyme functional annotation. We believe that we are just seeing the beginning of the widespread adoption of multi-modal models for protein functional prediction, and we expect that many researchers will find CARE useful for formulating and evaluating their models.

Chapter 4

STEERING GENERATIVE MODELS WITH EXPERIMENTAL DATA FOR PROTEIN FITNESS OPTIMIZATION

- (1) Yang, J.; Chu, W.; Khalil, D.; Astudillo, R.; Wittmann, B. J.; Arnold, F. H.; Yue, Y. *Advances in Neural Information Processing Systems* **2025**.

Abstract Protein fitness optimization involves finding a protein sequence that maximizes desired quantitative properties in a combinatorially large design space of possible sequences. Recent advances in steering protein generative models (e.g., diffusion models and language models) with labeled data offer a promising approach. However, most previous studies have optimized surrogate rewards and/or utilized large amounts of labeled data for steering, making it unclear how well existing methods perform and compare to each other in real-world optimization campaigns where fitness is measured through low-throughput wet-lab assays. In this study, we explore fitness optimization using small amounts (hundreds) of labeled sequence-fitness pairs and comprehensively evaluate strategies such as classifier guidance and posterior sampling for guiding generation from different discrete diffusion models of protein sequences. We also demonstrate how guidance can be integrated into adaptive sequence selection akin to Thompson sampling in Bayesian optimization, showing that plug-and-play guidance strategies offer advantages over alternatives such as reinforcement learning with protein language models. Overall, we provide practical insights into how to effectively steer modern generative models for next-generation protein fitness optimization.

4.1 Introduction

Proteins, sequences of amino acids, can be optimized for useful properties such as binding affinity, catalytic activity, or stability, numerically quantified as “fitness.” However, protein optimization is challenging: the design space of proteins is enormous, as a protein of length M can be constructed in 20^M different ways, of which only a negligible fraction are functional [16]. Moreover, most wet-lab assays only provide $10^2 - 10^3$ fitness labels. Consequently, researchers often rely on directed evolution, an iterative process aiming to incrementally improve protein fitness [43] through multiple rounds of mutation and experimental screening. In each round,

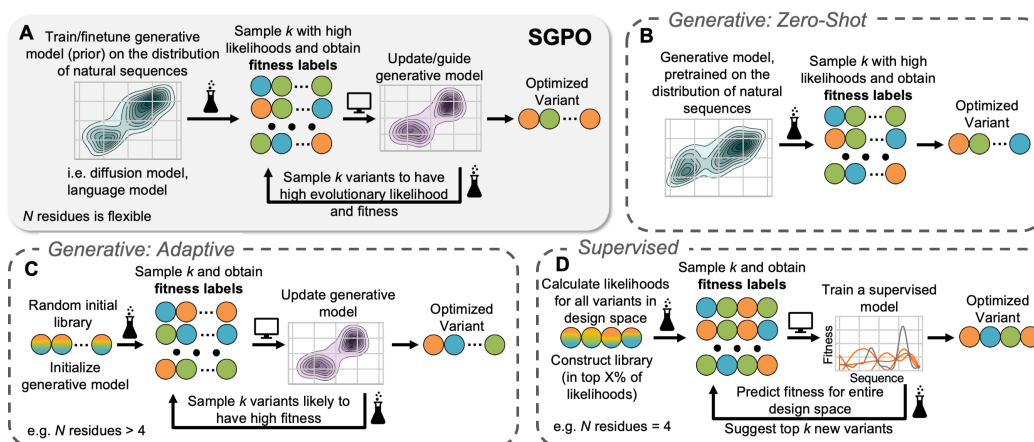


Figure 4.1: Comparison of steered generation for protein optimization (SGPO) to other ML-assisted workflows for protein engineering. (A) SGPO involves initializing a generative prior model to sample sequences with high natural likelihoods and steering that model with assay-labeled fitness data. Optimization is difficult because the design space is vast, and the throughput of wet-lab fitness assays (Erlenmeyer flask icon) is low, so adaptive learning across multiple iterations is beneficial. Previous methods have utilized generative models such as (B) fully zero-shot methods that sample highly natural sequences but do not utilize labeled fitness data or (C) those that only utilize labeled fitness. (D) Alternatively, supervised approaches involve enumerating to calculate fitness predictions for all variants in a design space, limiting them to optimizing few residues (i.e., $N < 9$).

a protein is mutated, the variants' fitnesses are measured, and the most beneficial variant is selected for the next iteration. However, this approach can be slow, often accumulating only one mutation per round, and inefficient, as it performs a local search limited to closely related protein sequences.

In recent years, there has been a growing interest in developing machine learning (ML)-assisted methods to optimize protein fitness more efficiently [49, 50, 129, 131, 249]. Many recent studies have focused on generative approaches combining unlabeled and labeled data for protein design. Broadly, these methods achieve *conditional generation* by steering generative priors of natural protein sequences [128] using fitness data, thereby enabling incorporation of the steered models into adaptive optimization cycles [131]. We refer to this class of methods as **Steered Generation for Protein Optimization (SGPO)**. These methods address the individual limitations of previous approaches (Fig. 4.1A, Table 4.1). First, SGPO leverages labeled data, which is essential for fitness goals that deviate from natural functions (e.g., engineering enzymes for non-native activities [1, 250]), unlike zero-shot methods relying solely on generative priors of natural sequences (*Generative: Zero-Shot*,

Fig. 4.1B) [57, 251–254]. Second, generative priors [80, 87] sample sequences with high evolutionary likelihoods and potentially higher fitness, giving these methods a significant advantage over approaches relying exclusively on labeled data (*Generative: Adaptive*, Fig. 4.1C) [131, 255–261]. Finally, SGPO scales to larger design spaces, unlike most supervised ML-assisted directed evolution (MLDE) approaches, which require enumerating and scoring all variants in the design space (*Supervised*, Fig. 4.1D) [52–55, 58, 59, 61, 64, 250, 262–264].

Despite these advantages, SGPO methods still face practical limitations in real-world fitness optimization, particularly across two major classes of approaches: guiding discrete diffusion models [114, 265–269] and finetuning models such as protein language models (PLMs) through reinforcement learning (RL) [86, 270–273]. The limitations of prior work are summarized as follows: (1) Few previous studies have explored steering with few ($10^2 - 10^3$) labeled sequences [114, 271] for protein optimization based on real fitness data, e.g. activity or fluorescence, rather than computational surrogates [114, 272]. (2) Most studies only evaluate one type of generative prior and steering strategy, so it is unclear how different combinations perform in practice. (3) There is room to incorporate principles from adaptive optimization, such as uncertainty-aware exploration (e.g., Bayesian optimization), which have shown clear benefits in protein engineering [64, 250].

In this study, we aim to understand the best practices for integrating SGPO into real-world engineering workflows. We focus here on modern generative models (i.e., discrete diffusion, language models) but acknowledge that other related methods are relevant, such as those based on variational autoencoders [274, 276] and other adaptive search strategies [277–279]. We explore the following questions: Which steering strategies perform best, and with which types of models? How can we utilize uncertainty to better explore the design space when performing guidance?

Overall, we make the following key contributions:

1. We motivate SGPO as a useful, general framework and contextualize existing methods for protein optimization under this umbrella.
2. We comprehensively evaluate design decisions for SGPO, including different generative models for sequences and steering strategies (Fig. 4.2 & Fig. 4.3, Section 4.2), offering best practices for protein optimization with few fitness labels.

Table 4.1: **SGPO is a general approach for protein fitness optimization that does not face the individual limitations of other strategies.** Namely, SGPO utilizes zero-shot knowledge from the natural distribution of proteins, can be guided by assay-labeled fitness data, and can optimize many residues (N) simultaneously. Beyond those listed here, there are many other studies that combine different elements of these approaches.

Approach	Prior Information Used?	Assay Fitness Used?	Scales to large N ?	Protein Examples (non-exhaustive)
SGPO	✓	✓	✓	Lisanza et al. [114], Nisonoff et al. [265], Goel et al. [269], Widatalla et al. [270], Stocco et al. [271], Blalock et al. [272], Brookes et al. [274], and Huang et al. [275]
Generative: ✓ Zero-Shot		×	✓	Hie et al. [57], Sumida et al. [251], Fei et al. [252], Seki et al. [253], and Lambert et al. [254]
Generative: × Adaptive		✓	✓	Song and Li [255], Stanton et al. [256], Brookes and Listgarten [258], Jain et al. [259], and Angermueller et al. [261]
Supervised ✓		✓	×	Wittmann et al. [53], Ding et al. [55], Hawkins-Hooker et al. [262], Zhao et al. [263], and Sun et al. [264]

3. We introduce ideas from adaptive optimization into SGPO by proposing a method that ensembles multiple plug-and-play fitness predictors and leverages their predictive uncertainty to enable more efficient exploration.
4. We are the first to adapt *decoupled annealing posterior sampling* [280] for SGPO, and this type of plug-and-play guidance has the strongest performance overall.

On the TrpB, CreiLOV, and GB1 protein fitness datasets, we find that SGPO methods can consistently identify high-fitness protein variants. In particular, our results highlight the advantages of plug-and-play guidance with diffusion models over finetuned language models—offering greater steerability and lower computational cost. To support future research and real-world adoption, our extensive, user-friendly code is available at <https://github.com/jsunn-y/SGPO>.

4.2 Related Work

Generative models for discrete sequences. The most widely adopted generative models for natural protein sequences are PLMs, such as autoregressive transformers [281] and masked language models [155]. Increasingly, various diffusion model [282] architectures have shown efficacy for modeling discrete data ($\mathbf{x}$) [283], such as protein sequences [89, 95], leveraging many similar learning techniques such as masking or autoregressive decoding [284–287] (Fig. 4.2). These generative prior models $p(\mathbf{x})$ can be categorized broadly into two types: those that perform diffusion in a continuous latent space [288–290] and those that diffuse directly over discrete space (Fig. 4.2). In the protein domain, it has also been shown that latent diffusion over embeddings from PLMs can be more effective [291–293]. Alternatively, models performing diffusion in discrete space use a transition matrix to update all discrete states in each timestep (D3PM) [294], which has later been formulated as continuous-time Markov chains [110, 285, 295, 296]. Two common ways to add noise to discrete sequences are to use uniform noise matrices or absorbing state (masking) matrices (Fig. 4.2). These have been followed by simplified frameworks showing some of the highest performance for modeling natural language, such as masked diffusion language models (MDLMs) [284, 287, 297] and a variation that uses uniform noise called uniform diffusion language models (UDLMs) [296]. We elaborate more on these methods in Section A.3.

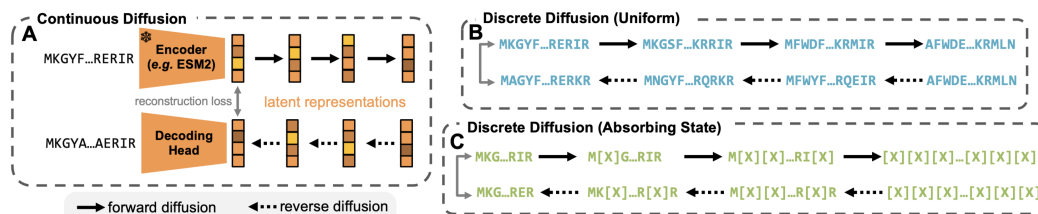


Figure 4.2: **Overview of different approaches to train diffusion models over discrete state spaces.** During inference, a noised latent representation or sequence is decoded into a reasonable sequence (bottom track for each method). [X] refers to a masked token.

Plug-and-play guidance strategies. An advantage of diffusion models is the ability to perform plug-and-play guidance based on fitness labels ($\mathbf{y}$) without finetuning the generative prior model weights, resulting in reduced training costs and potentially strong signal despite having few ($\sim 10^2$) labels. Guiding a continuous diffusion model often involves skewing the learned score function using gradients from a supervised value function that can predict labels $\mathbf{y}$ from data $\mathbf{x}$ [298–300].

These methods are often referred to as posterior sampling, as they aim to sample from the posterior distribution, $p(\mathbf{x}|\mathbf{y})$. Recent works extend this idea to guiding discrete diffusion models. Classifier guidance (CG) [265] skews the rate matrix of the reverse time Markov chain of discrete diffusion models using a time-dependent value function, $p(\mathbf{y}|\mathbf{x}_t, t)$; variable splitting methods (DAPS) [280, 301] use discrete diffusion models as denoisers and only require a value function of clean data, $p(\mathbf{y}|\mathbf{x}_0)$; diffusion optimized sampling (NOS) [268] trains a value function on continuous embeddings of discrete tokens and optimizes the embedding for higher fitness; sequential Monte Carlo methods (SMC) [302–306] evolve multiple particles from a series of distributions to approximate the posterior distribution in limit. We explain these methods in more detail in Section A.3, along with other variations on the guidance process. In this study, we focus on CG, DAPS, and NOS as guidance techniques (Fig. 4.3). Future work could also consider guidance techniques for autoregressive language models, such as future discriminators for generation (FUDGE) [307], plug and play language models (PPLM) [308], and twisted SMC [309, 310]. Additionally Xiong et al. [311], demonstrate how guidance generalizes to masked language models and order-agnostic autoregressive models.

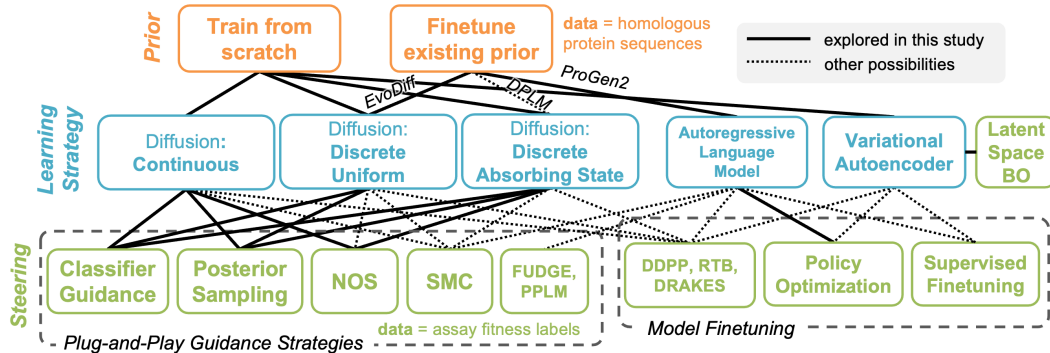


Figure 4.3: **Methods design space for SGPO: a non-exhaustive landscape of generative models for protein sequences and methods to steer them with labeled data.** Three major types of diffusion models for sequences include those that perform diffusion over continuous space and those that perform diffusion over discrete space with a uniform or absorbing state (masking) noising process. Various types of guidance strategies are compatible with certain models, in green (NOS: diffusion optimization sampling, SMC: sequential monte carlo, FUDGE: future discriminators for generation, PPLM: plug and play language models, DDPP: discrete denoising posterior prediction, RTB: relative trajectory balance, DPLM: diffusion protein language model, BO: Bayesian optimization). Differently, language models and variational autoencoders can be aligned with labeled data via reinforcement learning such as policy optimization or supervised finetuning.

Reinforcement learning via model finetuning. We consider RL broadly here as techniques that achieve conditional generation by finetuning generative models with labeled data, thus pushing those models to produce more favorable generations. There are emerging RL techniques applied to discrete diffusion models, including discrete denoising posterior prediction (DDPP) [312], relative trajectory balance (RTB) [313–315], and direct reward backpropagation with gumbel softmax trick (DRAKES) [316]. While the above strategies are specific to discrete diffusion models, supervised fine-tuning (SFT) and policy optimization are two important techniques used in RL that can be broadly applied to generative models such as language models (Fig. 4.3). Policy optimization has generally shown better performance than SFT [271, 272]; in particular, direct preference optimization (DPO) is often used for its algorithmic simplicity and ease of training [317] (details in Section A.3). RL has demonstrated utility for aligning generative models of proteins (language models, inverse folding models, variational autoencoders) with properties like stability [270–272, 318], but these methods can have high computational costs of finetuning and may require large amounts of labels ($> 10^3$) to effectively steer generations. We include DPO with an autoregressive PLM (finetuned ProGen2 [281]) as a baseline.

Adaptive optimization. Protein engineering is commonly conducted through adaptive workflows such as directed evolution [43] or ML-based approaches such as Bayesian optimization [256, 319]. These methods follow an iterative loop: labeled data is collected via expensive wet-lab assays, a surrogate model $p(\mathbf{y}|\mathbf{x})$ is trained or updated, an acquisition function implied by the surrogate is used to propose new sequences to evaluate, and the cycle repeats [64, 131, 250]. The surrogate model, often a Gaussian process or a deep ensemble, provides uncertainty estimates, which are used by an acquisition function (e.g., expected improvement, Thompson sampling) to balance exploration and exploitation of the design space. In this study, we adapt these ideas to guide diffusion models for protein sequence generation, as described in Section 4.4. A closely related line of work is latent space Bayesian optimization [256, 276, 320–323], which searches for optimal sequences within a latent space—typically learned by an autoencoder, which can implicitly capture a prior on natural protein sequences. In this work, we compare against APEXGo [276], a method that performs trust-region Bayesian optimization in the latent space of a variational autoencoder trained over protein sequences. There are also related methods that involve conditional sampling from a prior [274]. However, we note

that SGPO offers greater flexibility by avoiding reliance on an explicit latent space, which enables the use of modern, more powerful generative models such as diffusion models and protein language models that are not easily accommodated by traditional latent Bayesian optimization pipelines.

4.3 Problem Setup

We focus on evaluating methods that fall under SGPO, where the primary downstream task entails starting from a known sequence with some level of fitness for a target objective (i.e. activity, stability, fluorescence, binding, etc.) and identifying a modified sequence with maximized fitness, where real-world fitness can only be measured for 10^2 to 10^3 sequences. Our goal is to sample sequences with maximum fitness $\mathbf{y}$ from the *generative prior* $p(\mathbf{x})$, which is trained on the multiple sequence alignment (MSA) of homologous protein sequences that are evolutionarily related to a known protein with some level of desired fitness (details in [Section A.3](#)). This model can be thought of as capturing the distribution of sequences with high likelihood from a given protein family.

During inference, sequences can be sampled *unconditionally* from $p(\mathbf{x})$, or sampling can be *guided* using a supervised model of the form $p(\mathbf{y}|\mathbf{x}) \propto \exp(f(\mathbf{x})/\beta)$, where $f(\cdot)$ is a learned fitness predictor—also referred to as the *classifier* or *value function*. This predictor is trained on a small number of labeled sequence-fitness pairs (typically in the hundreds) to reflect practical data limitations. The goal of guided sampling is to generate protein sequences from the posterior distribution, $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{x}) \exp(f(\mathbf{x})/\beta)$. We use a computational *oracle* to acquire and evaluate fitness labels $\mathbf{y}$, to simulate how fitness would be measured in a real-world campaign. Details on training and guidance with the value function are provided in [Section A.3](#) and [Table A.2](#). As an alternative steering method to guidance, we finetune the generative prior with labeled data using an autoregressive language model (ARLM) and DPO, which serves as a baseline. We further compare to a baseline of latent space Bayesian optimization. The strength of steering is tuned by method-specific hyperparameters.

4.4 Results

We study three proteins, the TrpB enzyme [[153](#)], the CreiLOV fluorescent protein [[324](#)], and the GB1 binding protein [[325](#)] due to the availability of fitness data across many residues ([Table 4.2](#)). We focus protein fitness optimization to a design space of 15 residues in TrpB (only these positions are allowed to vary) and all 119 and 56

Table 4.2: **Summary of datasets used in this work.** Train and test fitness refer to the number of fitness labels used for training and testing the oracle. We focus on TrpB and CreiLOV, with some of the GB1 results moved to the Appendix. While the TrpB dataset has a lot more training labels, it may be more difficult to learn due to relatively high amounts of epistatic effects between residues (non-additivity of mutation effects).

Dataset	Length	Targeted Residues	Design Space	MSA Size	Train Fitness	Test Fitness
TrpB Enzyme Activity	389	117, 118, 119, 162, 166, 182, 183, 184, 185, 186, 227, 228, 230, 231, 301	$N=15$	$5.7e4$	75,618	23,313
CreiLOV Fluorescence	119	All	$N=119$	$3.7e5$	6,842	2,401
GB1 Binding	56	All	$N=56$	126	$3.9e6$	$9.6e4$

residues in CreiLOV and GB1, respectively. For each protein’s variants, we evaluate fitness by approximating it via a supervised oracle trained on a large amount of real data (Section A.3).

Model pretraining captures the distribution of evolutionarily related protein sequences and enables sampling sequences with high fitness

Based on the methods explained in Section A.3, we trained generative priors on natural sequences from the MSA, focusing on continuous diffusion models (*Continuous*), discrete diffusion models with uniform (*D3PM*, *UDLM*) and absorbing state noising processes (*MDLM*), and autoregressive language models (*ARLM*) (Table 4.3). Overall, the trained models capture the natural distribution of protein sequences, with the D3PM models seeming to match the distribution the most closely while also generating sequences with high diversity (Fig. 4.4, Fig. B.7). The two different diffusion models over continuous space show comparatively lower performance, and diffusing over the latent space of ESM embeddings does not boost performance on this task. The UDLM model has low performance due to mode collapse (Fig. B.7, Fig. B.9). Future work could finetune the pretrained diffusion protein language model (DPLM) as an MDLM [95].

Overall, we found that pretrained priors sample protein variants that have higher mean fitness, which corroborates previous studies finding that sequences with higher

Table 4.3: **Summary of generative priors evaluated in this work.** Each generative prior was trained on an MSA of homologous natural sequences. All denoising processes were modeled using a transformer architecture (Section A.3). *Italicized* models were further explored in downstream guidance experiments.

Model	Type	Noise	# Params	Notes
<i>Continuous</i>	Continuous Diffusion	Gaussian	27.9 M	
Continuous-ESM			25.5 M	diffusion over ESM embeddings
D3PM-Baseline	Discrete Diffusion	Uniform	37.9 M	
<i>D3PM</i>			37.9 M	finetuned from EvoDiff 38M-Uniform
UDLM			28.6 M	uniform diffusion language model
<i>MDLM</i>	Discrete Diffusion	Absorbing	28.6 M	masked diffusion language model
<i>ARLM</i>	Language Model	n/a	151 M	autoregressive language model finetuned from ProGen2-small

evolutionary likelihood are also likely to have higher fitness [57, 61]. Based on these results, we proceeded to perform remaining experiments with one model from each category of model type, namely the *Continuous*, *D3PM*, *MDLM*, and *ARLM* models.

Evaluating SGPO design choices

Impressively, steering with modest amounts of labeled data (200 sequence-fitness pairs) enables most models and methods to generate sequences with even higher fitness, while sacrificing some generation diversity (Fig. 4.5). In this low data regime, guidance with diffusion models outperforms DPO with language models; the latter does not enable as much steerability. CG and DAPS enable the strongest steerability overall, but DAPS outperforms CG for the continuous models (Fig. 4.5A, D). In general, guidance seems to work similarly for uniform diffusion (D3PM) and to absorbing state diffusion (MDLM). Overall, the continuous diffusion models do not perform as well as other models, as the prior does not capture the distribution of natural sequences with high fitness as well (Fig. 4.5A). NOS does not seem to allow for as much steerability, despite an extensive hyperparameter scan (Table A.3). Finally, we conducted a closer analysis of the number of unique sequences generated by the

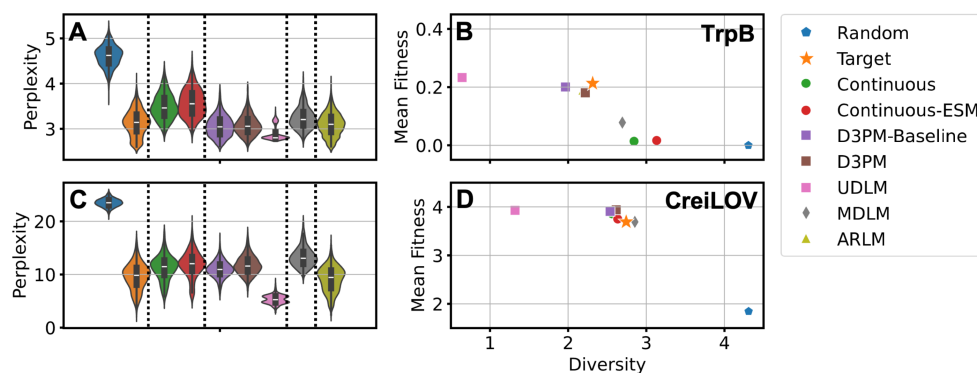


Figure 4.4: Pretrained generative priors capture the target distribution of naturally occurring sequences that are homologous to TrpB (A-B) and CreiLOV (C-D), respectively. Lower perplexity corresponds to higher likelihood in the model. The diversity of sequences was computed as the average Shannon entropy of mutated positions with mean fitness corresponding to the oracle predictions. While the various models largely achieve comparable performance, the D3PM models capture the target distribution with the highest fidelity, whereas the UDLM model is prone to mode collapse. For each model, 1000 sequences were sampled and repeats were allowed to approximate the distribution. To approximate the target distribution, 1000 sequences were sampled from the MSA used for pretraining. Perplexity was calculated by passing generated sequences through the 764 M parameter ProGen2-base model. More details on model training can be found in [Table 4.3](#) and [Section A.3](#), and GB1 results are provided in [Fig. B.8](#).

steered models and confirmed that most models produce entirely novel sequences, suggesting that they are not over-steering ([Fig. B.10](#)).

“Thompson sampling” using an ensemble of classifiers is effective for adaptive optimization

Next, we performed adaptive optimization experiments, which mimic real-world protein engineering scenarios and follow a setup similar to batch Bayesian optimization: in each round, a batch of sequences is sampled, evaluated for fitness, and used to retrain a supervised value function that guides sampling from the pre-trained prior. We focused on the MDLM models with the CG and DAPS guidance strategies, as these combinations achieved the best performance in our earlier set of experiments ([Fig. 4.5](#)). Based on findings from these previous experiments, we selected the ideal guidance strength hyperparameter to balance fitness and diversity—ensuring high predicted fitness without significantly compromising sequence diversity ([Table A.3](#)). For both guidance strategies, we employed an algorithm akin to Thompson sampling [326, 327], drawing a different value function from a frequentist ensemble of neural network regressors to guide the generation of each

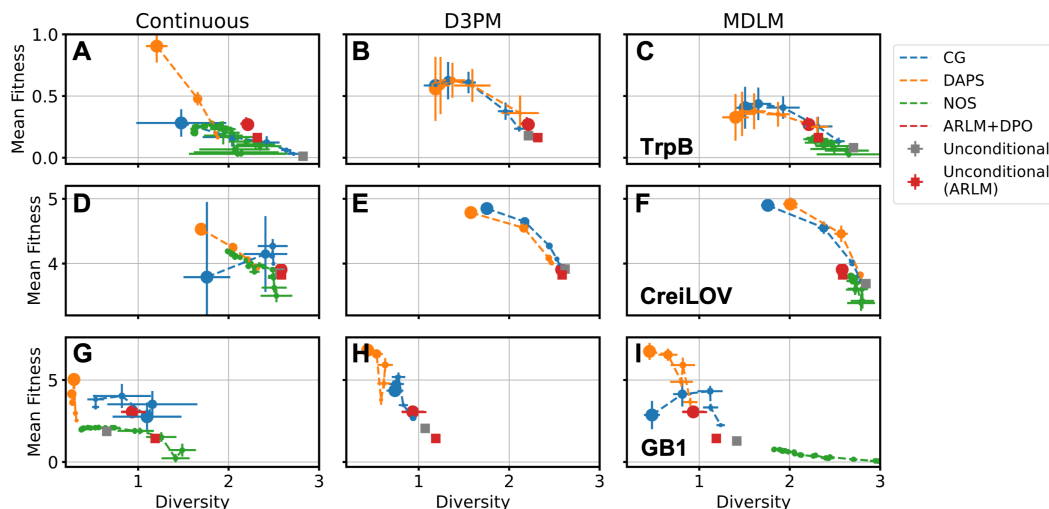


Figure 4.5: **Pareto boundaries demonstrate the trade-off between generating sequences with high fitness and high diversity** for TrpB (A-C), CreiLOV (D-F), and GB1 (G-I). Sequences sampled from the generative models (Continuous, D3PM, and MDLM), after guidance with labeled fitness data, are enriched in high-fitness protein variants, and most methods show higher performance than the ARLM+DPO baseline. Larger circle indicates a stronger guidance strength hyperparameter (excluding NOS), specified in Table A.3. Each experiment was repeated using 10 different standardized sets of 200 unique sequences used for steering, each drawn from the D3PM prior, and error bars show standard deviation. Mean fitness and diversity were calculated based on 200 generated samples, with diversity calculated as the average Shannon entropy of amino acids at mutated positions. Unconditional refers to sequences sampled from the prior with no guidance.

new sample [250]. Pseudocode for our adaptive optimization algorithm is provided in Section A.3.

Plug-and-play guidance strategies outperform baselines such as DPO with an ARLM, sampling just from the unconditional generative prior, and latent space Bayesian optimization with APEXGo (Fig. 4.6): Sampled sequences achieve higher values of mean and maximum fitness. Furthermore, campaigns using an ensemble of value functions and “Thompson sampling” achieve higher maximum fitness than those using only a single value function for guidance (Table B.1), which may be because these models enable more exploration of sequence space (Fig. B.11). However, it is difficult to ascertain whether CG or DAPS works better as a guidance strategy, as the performance is highly dependent on the guidance strength hyperparameter, and the optimal hyperparameter will not typically be known in a real-world campaign. Because the oracle may not capture the true nature of the protein fitness landscape, we also suggest making relative comparisons here rather than absolute comparisons

between model performance.

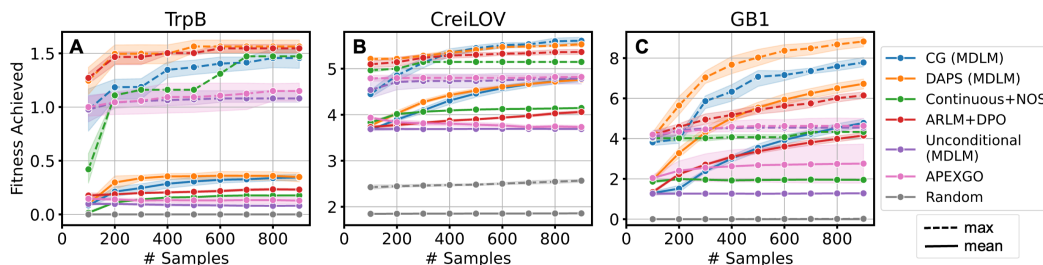


Figure 4.6: **Maximum/mean fitness achieved improves over multiple iterations of steering in an adaptive setting similar to batch Bayesian optimization** for TrpB (A), CreiLOV (B), and GB1 (C). 100 sequences were sampled in each round. Within each round, an ensemble of 10 value functions (classifiers) was trained on fitness data from all previously queried samples, and each new sample was generated by the MDLM model guided with a value function sampled from the ensemble (akin to Thompson sampling). Only unique, novel samples were acquired. Guidance strength parameter is provided in Table A.3. Error bars show standard deviation between 5 different random initializations.

4.5 Discussion

In this work, we conduct a comprehensive study of SGPO methods and demonstrate that it is an effective approach for protein fitness optimization, by capturing the distribution of natural protein sequences with a generative prior and then steering the generations with labeled data. We find that DAPS with discrete diffusion models has the highest performance overall, and plug-and-play guidance-based strategies are generally more effective than finetuning language models; the latter can be difficult when only few fitness labels are available. SGPO approaches also outperform latent space Bayesian optimization (namely APEXGo), which we attribute to the difficulty in calibrating the trust region in very low-data regimes with limited rounds of optimization and the fact that latent space Bayesian optimization relies heavily on the structure of the latent space learned during generation model training, which can limit extrapolation to high-fitness but unnatural variants.

Using plug-and-play guidance approaches has other advantages. First, only one hyperparameter (guidance strength) needs to be tuned. In real-world engineering scenarios, even in the absence of ground truth fitness labels, one practical approach to selecting the guidance strength is to scan over values and choose the highest setting for which n generated sequences remain unique and novel relative to previously measured sequences, where n corresponds to the screening throughput available for

the next round. By contrast, for DPO, various hyperparameters need to be tuned, and the training process has to be monitored closely. Even for NOS, different parameters such as the step size, the number of steps, and the stability coefficient must be tuned together. A further advantage of guidance is the low computational cost required, as the prior model weights are not updated during guidance. Pretraining/finetuning to obtain each initial prior was achieved on a single H100 GPU in less than one hour while each individual guidance experiment took minutes; pretraining language models took several hours on a single GPU.

Still, there are certain limitations of our work. We focused on proteins with fitness as mostly native function, but it would be interesting to test SGPO on other protein fitness optimization tasks where the pretrained prior may not provide as much utility. We also focused on protein optimization where only $\approx 10^2$ fitness labels were available; different methods, such as RL, may perform better for applications where larger amounts of fitness data are available [131, 272]. We focused on guidance strategies and did not test DPO or model finetuning-based methods with discrete diffusion models, but future work could adapt these methods for discrete diffusion [328]. Furthermore, for TrpB and for language models, we manually mapped sequences back into the design space after generation (Section A.3), but explicitly building this into sampling techniques, such as inpainting in masked models [269, 272] may lead to improved performance. We did not consider insertions or deletions, but variable-length sequence generation could be considered in the future. Finally, we did not directly compare to existing approaches for protein engineering such as directed evolution for reasons explained in Section A.3.

There are several promising directions for future work to improve and extend SGPO methods. For instance, we experimented with guiding generation using value functions sampled from a Gaussian process posterior, enabling principled Thompson sampling from a fully Bayesian perspective. However, the Gaussian process struggled to model high-dimensional protein representations, leading to poor performance. This limitation could potentially be addressed with better kernel choices [141, 157, 250]. Recent work has also begun to incorporate multi-objective optimization [329–332] and uncertainty quantification [333] when guiding diffusion models. Simultaneously, alternatives to acquisition-function-based approaches are being developed to enable Bayesian optimization in large design spaces where enumeration is infeasible [136]. Other emerging approaches—closer in spirit to flow matching—are being proposed for discrete data and may offer new opportunities for

exploration [266, 334, 335]. Finally, for masked diffusion models, strategies such as remasking or scheduling could be explored to improve inference, particularly to enhance model amenability to guidance [336–339]. It will also be interesting to further explore guidance in other discrete domains such as natural language and small molecules [296].

In summary, guiding generative models with labeled data offers a powerful, flexible, and principled framework for protein fitness optimization, as it effectively leverages both the evolutionary information encoded in natural protein sequences and task-specific fitness objectives. At the same time, we recognize the potential dual-use risks: such methods could, in principle, be misused to design harmful proteins, underscoring the importance of appropriate safeguards [340, 341]. In short, our work has examined multiple effective SGPO strategies and offered insights on best-practices for real-world protein fitness optimization, laying the groundwork for further exploration and wet-lab validation.

CONCLUSION AND OUTLOOK

- (1) Yang, J.; Li, F.-Z.; Long, Y.; Arnold, F. H. *Cell Systems* **2026**, *17*, 101372.

5.1 Toward General, Autonomous Protein Engineering

In [Chapter 4](#), we proposed a new paradigm for generative [\[80, 86, 87, 274\]](#) AI-driven protein engineering that unifies the previously separate tasks of protein discovery/design ([Chapter 3](#)) and protein optimization ([Chapter 2](#)). Building on this, we argue that the future of protein engineering will utilize *a unified generative model that understands the rules of natural proteins while also integrating assay-labeled data* ([Fig. 5.1](#)). Whereas directed evolution is limited to slow, local searches, the proposed model could leverage multiple data sources to travel to semi-distant regions of sequence space—regions unexplored by evolution but still viable for maximizing a desired function. Such a model would encompass many existing tasks in protein design, such as function-guided design and inverse folding, and lay the foundation for many more [\[65\]](#).

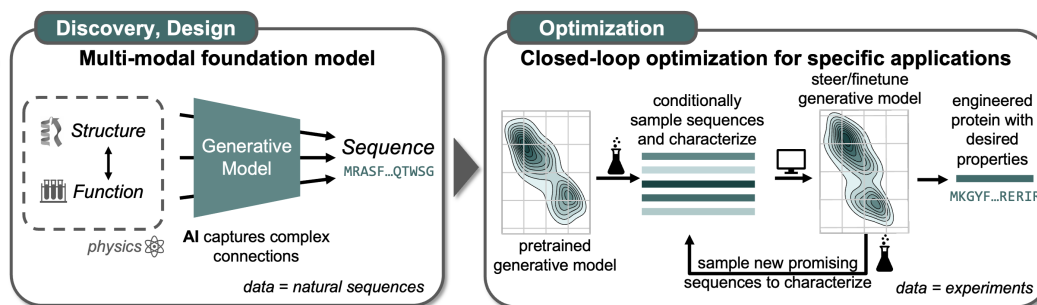


Figure 5.1: Vision for all-in-one enzyme (protein) sequence engineering with a generative model that learns the distribution of viable proteins, conditioned on properties such as fitness, function, and/or structure. Such a model would be trained on publicly available data on natural proteins (left), and be updated/steered with data from experimental measurements (right). For more specific tasks, weak connections between modalities would benefit the most from steering. The thickness of the arrow shows the relative strength of the connection given existing data and/or tools. Many existing tasks in protein discovery and design fall under this framework, e.g. protein fitness optimization, inverse folding, natural language-guided functional protein generation, etc.

Crucially, such a generative model would need to be conditioned on relevant information to enable controllable generation [85, 342]. Two aspects of the conditions are essential: (1) “function,” which qualitatively describes the reaction(s) that an enzyme can perform, and (2) “fitness,” which quantitatively measures performance for a particular function and can be a combination of properties such as catalytic rate, stability, and protein expression level (Fig. 1.1). Some preliminary work has explored “function”-based conditional generation using language models [123, 125] and diffusion models [124, 343]. Function could be conveyed as reaction representations, natural language descriptions, or protein structural features—particularly for binder design, where current methods largely focus on structure generation and inverse folding. Nevertheless, these efforts remain largely fragmented, highlighting the critical need for a unified multimodal framework that integrates diverse forms of knowledge, enabling transfer learning across tasks and generalization to unseen functions (Fig. 5.1)—just as a chemist, structural biologist, and bioinformatician would collaborate to understand or predict a protein’s function.

Efforts in this direction have included conditioning protein language models [79] on protein structures [91, 116, 344] or functional descriptors [91, 123, 125], discrete diffusion models for protein sequence generation [343], and the integration of natural language, protein structure, and protein sequence into unified representation learning frameworks [120, 121, 124, 200]. However, current AI models do not have a generalized understanding of enzyme functions and reaction mechanisms and thus cannot design enzymes with novel functions. To solve this, new ML architectures are needed to enable conditional sampling from probabilistic distributions and bridge disparate modalities spanning natural language, sequences, graphs, 3D coordinates, reaction mechanisms, tabular data, and more [345]. This will be a challenging task, because most strategies used in computer vision and natural language processing are not perfectly suited for biological data, and different modalities are annotated for different proteins. Utilizing inductive biases from physics such as information about electronics and transition states may be crucial to improving the generalization ability of such models.

More difficult applications, such as optimizing an enzyme for a new-to-nature reaction that is not captured in the natural distribution of enzyme function will require guiding or finetuning the generative model based on expensive (low-throughput) real-world fitness measurements (Fig. 5.1). Developing new ML techniques for steering generative models as part of active learning, or iterative cycles of experi-

mentation and computational modeling, will be essential. For example, this could involve guidance with diffusion models [89, 95, 114, 256, 265, 346–348] or reinforcement learning with language models [63, 271, 272, 349]. In essence, these methods would shift the distribution of the generative model to *conditionally* sample proteins with desired properties (Fig. 5.1) [128, 131]. Even with the help of inductive biases from physics, some new-to-nature reactions will require such a large extrapolative leap that activity may be impossible to attain without first conditioning the model on related but more readily attainable protein-reaction data, analogous to the “substrate walking” approach often used in directed evolution [350].

In the not-too-distant future, fully automated “cloud” labs, or autonomous laboratories, could perform experiments continuously and reproducibly, and this would be accessible to diverse users regardless of their discipline or expertise. An AI agent could design experiments, execute, analyze, and improve its model, and repeat this process in iterative cycles of active learning. This kind of self-improving AI-driven experimentation is already beginning to emerge, with increasing capabilities for autonomous hypothesis generation and testing [135, 351–358].

As part of long-term active learning, the development of a standardized enzyme sequence-fitness database will be indispensable. While the pretraining step can utilize public data sources such as the Protein Data Bank (sequence-structure), reactions in BRENDA (sequence-function), and natural language descriptions from SwissProt and literature articles (sequence-function), there is a critical need for a sequence-fitness database comprising more specific fitness measurements such as data from enzymatic activity, stability, and expression assays, including environmental conditions, kinetic behaviors, substrate specificities, and reaction mechanisms. For this to become a reality, an open-source, collaborative initiative could be instrumental, where research institutions and industry players contribute to and benefit from the growth of the database [359]. Standardized benchmarks and/or competitions could further accelerate the pace of advancement and foster a more integrated research community [360]. With these datasets, automated enzyme engineering will enable us to create new genetically encoded chemistry much more efficiently than human-driven processes alone. As ML models continue to advance, they will progressively improve by integrating knowledge across different modalities.

5.2 Closing Thoughts

Nature has been exploring the universe of enzymes long before any of us were born, albeit slowly and with its own objective: survival and reproduction. With directed evolution, we can steer enzymes toward functions useful to humans, but we are still limited to a slow and rather naïve optimization process. But now we are armed with AI, a powerful tool that can help us traverse sequence space to uncover useful new enzymes. The outlook for protein design with AI is particularly promising for several reasons. (1) Proteins have a track record of reliable optimization for many functions by nature and by humans using nature’s original optimization strategy, evolution. By contrast, there is no guarantee that other forms of matter—chemicals, materials, etc.—can be so readily improved. (2) Proteins have been explored by nature long before we came along, and a wealth of data is accumulating on these natural designs that can inform future designs. These literal life lessons are being collected in databases to inform future engineering. In comparison, we have limited data on other materials, most of which have only been constructed by humans since the industrial revolution. (3) Finally, and critically, we have the machinery to build nearly any desired protein sequence, encoded in DNA and produced by a cell factory. For other materials, synthesis remains a major bottleneck [361]. With advancements in biological sequencing [151, 166], synthesis [167, 362, 363], and assay technology [364], our knowledge of and ability to manipulate protein space will only become more powerful.

In the era of artificial intelligence, we may wonder how effective ML approaches will be. Existing methods can design proteins with novel forms, but future work is needed to design proteins that can access novel functions, such as new-to-nature reactions (Fig. 5.2). To what extent can learning from natural proteins extrapolate to new sequences or even new structures? Is there enough density of natural functional enzymes to infer sequences that might lead to novel functions? AI workflows for new-to-nature enzyme activity will surely need assay-labeled data in addition to knowledge of natural proteins, but how many experiments will be necessary? John Maynard Smith articulated that evolution is possible because there is a high enough density of functional proteins in single-mutant neighbors, a realization that made it possible to access new functional proteins by directed evolution. Unfortunately, we do not have precedent to predict how useful existing data on natural proteins will be for creating functional novelty. Nonetheless, exploiting all of the data we have will undoubtedly get us to our goals faster.

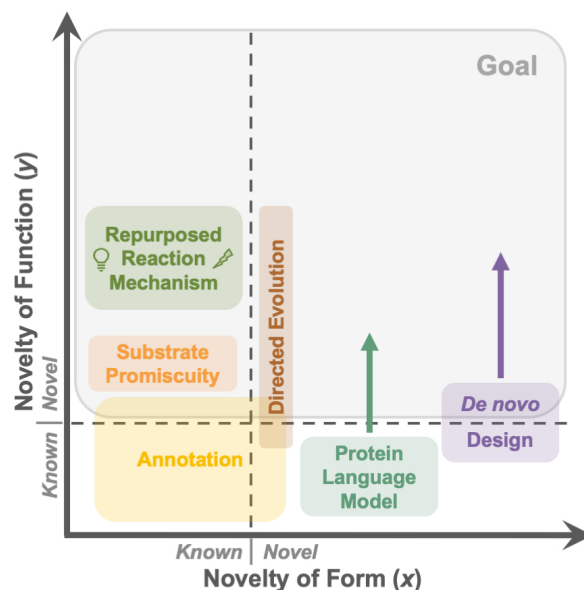


Figure 5.2: **Most existing ML methods can generate proteins with novel sequences/structures (forms), but designing proteins with novel enzymatic functions is more difficult.** Novel enzyme functions can be achieved by discovering sequences via functional annotation, taking advantage of promiscuity with respect to other substrates, exploiting novel chemical mechanisms of action, or modifying enzyme sequences through directed evolution. Developing better ML methods to unlock novel functions that are inaccessible from existing enzymes would be highly impactful.

Overall, AI is learning to compose across multiple modalities, accelerating engineering toward functional proteins, and filling in the enzymatic universe. While we have focused here on enzyme engineering, many of the ideas and approaches outlined here generalize to other protein engineering applications where unlabeled and labeled data can be utilized together. With regards to biosecurity, the generation of new toxins, resistance mechanisms, or pathogens does not lie far from our brave new world of enzymatic chemistry, so there will need to be monitoring of such tasks [340, 341, 365]. One day, we may replace inefficient, wasteful human chemistry with the highly selective, efficient chemistry performed by enzymes. Let us use AI to safely propel us toward that dream.

BIBLIOGRAPHY

- (1) Arnold, F. H. *Angewandte Chemie International Edition* **2018**, 57, 4143–4148.
- (2) Buller, R.; Lutz, S.; Kazlauskas, R. J.; Snajdrova, R.; Moore, J. C.; Bornscheuer, U. T. *Science* **2023**, 382, eadh8615.
- (3) Pyser, J. B.; Chakrabarty, S.; Romero, E. O.; Narayan, A. R. H. *ACS Central Science* **2021**, 7, 1105–1116.
- (4) Reisenbauer, J. C.; Sicinski, K. M.; Arnold, F. H. *Current Opinion in Chemical Biology* **2024**, 83, 102536.
- (5) Wilson, D. B. *Current Opinion in Biotechnology* **2009**, 20, 295–299.
- (6) Gomes, E. S.; Schuch, V.; Lemos, E. G. D. M. *Brazilian Journal of Microbiology* **2013**, 44, 1007–1034.
- (7) Arnold, F. H. *Microbe Magazine* **2011**, 6, 316–318.
- (8) Dennett, D. C., *Darwin's Dangerous Idea: Evolution and the Meanings of Life*; Simon & Schuster: 1995.
- (9) Smith, J. M. *Nature* **1970**, 225, 563–564.
- (10) Tawfik, O. K. a. D. S. *Annual Review of Biochemistry* **2010**, 79, 471–505.
- (11) Bayer, T.; Wu, S.; Snajdrova, R.; Baldenius, K.; Bornscheuer, U. *Angewandte Chemie International Edition* **2025**, e202505976.
- (12) Bell, E. L.; Finnigan, W.; France, S. P.; Green, A. P.; Hayes, M. A.; Hepworth, L. J.; Lovelock, S. L.; Niikura, H.; Osuna, S.; Romero, E.; Ryan, K. S.; Turner, N. J.; Flitsch, S. L. *Nature Reviews Methods Primers* **2021**, 1, 46.
- (13) Chen, K.; Arnold, F. H. *Nature Catalysis* **2020**, 3, 203–213.
- (14) Kan, S. B. J.; Lewis, R. D.; Chen, K.; Arnold, F. H. *Science* **2016**, 354, 1048–1051.
- (15) Sarai, N. S.; Fulton, T. J.; O'Meara, R. L.; Johnston, K. E.; Brinkmann-Chen, S.; Maar, R. R.; Tecklenburg, R. E.; Roberts, J. M.; Reddel, J. C. T.; Katsoulis, D. E.; Arnold, F. H. *Science* **2024**, 383, 438–443.
- (16) Romero, P. A.; Arnold, F. H. **2009**, 10, 866–876.
- (17) Pierce, N. A.; Winfree, E. *Protein Engineering, Design and Selection* **2002**, 15, 779–782.
- (18) Bornscheuer, U. T.; Huisman, G. W.; Kazlauskas, R. J.; Lutz, S.; Moore, J. C.; Robins, K. *Nature* **2012**, 485, 185–194.

- (19) Leveson-Gower, R. B.; Mayer, C.; Roelfes, G. *Nature Reviews Chemistry* **2019**, *3*, 687–705.
- (20) Miller, D. C.; Athavale, S. V.; Arnold, F. H. *Nature Synthesis* **2022**, *1*, 18–23.
- (21) Knight, A. M.; Kan, S. B. J.; Lewis, R. D.; Brandenberg, O. F.; Chen, K.; Arnold, F. H. *ACS Central Science* **2018**, *4*, 372–377.
- (22) Bedbrook, C. N.; Rice, A. J.; Yang, K. K.; Ding, X.; Chen, S.; LeProust, E. M.; Gradinaru, V.; Arnold, F. H. *Proceedings of the National Academy of Sciences* **2017**, *114*, E2624–E2633.
- (23) Merkl, R.; Sterner, R. *Biological Chemistry* **2016**, *397*, 1–21.
- (24) Voigt, C. A.; Martinez, C.; Wang, Z.-G.; Mayo, S. L.; Arnold, F. H. *Nature Structural Biology* **2002**, *9*, 553–558.
- (25) Alford, R. F. et al. *Journal of Chemical Theory and Computation* **2017**, *13*, 3031–3048.
- (26) Goldenzweig, A. et al. *Molecular Cell* **2016**, *63*, 337–346.
- (27) Khersonsky, O.; Lipsh, R.; Avizemer, Z.; Ashani, Y.; Goldsmith, M.; Leader, H.; Dym, O.; Rogotner, S.; Trudeau, D. L.; Prilusky, J.; Amengual-Rigo, P.; Guallar, V.; Tawfik, D. S.; Fleishman, S. J. *Molecular Cell* **2018**, *72*, 178–186.e5.
- (28) Leman, J. K. et al. *Nature Methods* **2020**, *17*, 665–680.
- (29) Schymkowitz, J.; Borg, J.; Stricher, F.; Nys, R.; Rousseau, F.; Serrano, L. *Nucleic Acids Research* **2005**, *33*, W382–W388.
- (30) Weinstein, J. J.; Goldenzweig, A.; Hoch, S.; Fleishman, S. J. *Bioinformatics* **2021**, *37*, ed. by Ponty, Y., 123–125.
- (31) Huang, P.-S.; Boyken, S. E.; Baker, D. *Nature* **2016**, *537*, 320–327.
- (32) Kalvet, I.; Ortmayer, M.; Zhao, J.; Crawshaw, R.; Ennist, N. M.; Levy, C.; Roy, A.; Green, A. P.; Baker, D. *Journal of the American Chemical Society* **2023**, *145*, 14307–14315.
- (33) Siegel, J. B.; Zanghellini, A.; Lovick, H. M.; Kiss, G.; Lambert, A. R.; St.Clair, J. L.; Gallaher, J. L.; Hilvert, D.; Gelb, M. H.; Stoddard, B. L.; Houk, K. N.; Michael, F. E.; Baker, D. *Science* **2010**, *329*, 309–313.
- (34) Cao, L. et al. *Nature* **2022**, DOI: [10.1038/s41586-022-04654-9](https://doi.org/10.1038/s41586-022-04654-9).
- (35) Pacesa, M. et al. BindCraft: One-Shot Design of Functional Protein Binders, 2024.
- (36) Siddiqui, S. A.; Stuyver, T.; Shaik, S.; Dubey, K. D. *JACS Au* **2023**, *3*, 3259–3269.

- (37) Fersht, A., *Structure and Mechanism in Protein Science: A Guide to Enzyme Catalysis and Protein Folding*. W. H. Freeman and Company: New York, 1999.
- (38) Olsson, M. H. M.; Parson, W. W.; Warshel, A. *Chemical Reviews* **2006**, *106*, 1737–1756.
- (39) Tokuriki, N.; Stricher, F.; Serrano, L.; Tawfik, D. S. *PLoS Computational Biology* **2008**, *4*, ed. by Eisenberg, D., e1000002.
- (40) Tokuriki, N.; Tawfik, D. S. *Science* **2009**, *324*, 203–207.
- (41) Chu, A. E. *Nature Biotechnology* **2024**, *42*, 203–215.
- (42) Röthlisberger, D.; Khersonsky, O.; Wollacott, A. M.; Jiang, L.; DeChancie, J.; Betker, J.; Gallaher, J. L.; Althoff, E. A.; Zanghellini, A.; Dym, O.; Albeck, S.; Houk, K. N.; Tawfik, D. S.; Baker, D. *Nature* **2008**, *453*, 190–195.
- (43) Packer, M. S.; Liu, D. R. *Nature Reviews Genetics* **2015**, *16*, 379–394.
- (44) Wang, Y.; Xue, P.; Cao, M.; Yu, T.; Lane, S. T.; Zhao, H. *Chemical Reviews* **2021**, *121*, 12384–12444.
- (45) Drummond, D. A.; Silberg, J. J.; Meyer, M. M.; Wilke, C. O.; Arnold, F. H. *Proceedings of the National Academy of Sciences* **2005**, *102*, 5380–5385.
- (46) Zhao, H.; Giver, L.; Shao, Z.; Affholter, J. A.; Arnold, F. H. *Nature Biotechnology* **1998**, *16*, 258–261.
- (47) Fröhlich, C.; Bunzel, H. A.; Buda, K.; Mulholland, A. J.; Van Der Kamp, M. W.; Johnsen, P. J.; Leiros, H.-K. S.; Tokuriki, N. *Nature Catalysis* **2024**, *7*, 499–509.
- (48) Miton, C. M.; Buda, K.; Tokuriki, N. *Current Opinion in Structural Biology* **2021**, *69*, 160–168.
- (49) Yang, K. K.; Wu, Z.; Arnold, F. H. *Nature Methods* **2019**, *16*, 687–694.
- (50) Yang, J.; Li, F.-Z.; Arnold, F. H. *ACS Central Science* **2024**, *10*, 226–241.
- (51) Romero, P. A.; Krause, A.; Arnold, F. H. **2012**, *110*, E193–E201.
- (52) Thomas, N. et al. *Cell Systems* **2025**, *16*, 101236.
- (53) Wittmann, B. J.; Yue, Y.; Arnold, F. H. *Cell Systems* **2021**, *12*, 1026–1045.e7.
- (54) Wu, Z.; Kan, S. B. J.; Lewis, R. D.; Wittmann, B. J.; Arnold, F. H. *Proceedings of the National Academy of Sciences* **2019**, *116*, 8852–8858.
- (55) Ding, K.; Chin, M.; Zhao, Y.; Huang, W.; Mai, B. K.; Wang, H.; Liu, P.; Yang, Y.; Luo, Y. *Nature Communications* **2024**, *15*, 6392.
- (56) Fram, B. et al. *Nature Communications* **2024**, *15*, 5141.

- (57) Hie, B. L.; Shanker, V. R.; Xu, D.; Bruun, T. U. J.; Weidenbacher, P. A.; Tang, S.; Wu, W.; Pak, J. E.; Kim, P. S. *Nature Biotechnology* **2023**, DOI: [10.1038/s41587-023-01763-2](https://doi.org/10.1038/s41587-023-01763-2).
- (58) Hsu, C.; Nisonoff, H.; Fannjiang, C.; Listgarten, J. *Nature Biotechnology* **2022**, *40*, Bandiera_abtest: a Cg_type: Nature Research Journals Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Machine learning;Protein design Subject_term_id: machine-learning;protein-design, 1114–1122.
- (59) Jiang, K.; Yan, Z.; Di Bernardo, M.; Sgrizzi, S. R.; Villiger, L.; Kayabolen, A.; Kim, B. J.; Carscadden, J. K.; Hiraizumi, M.; Nishimasu, H.; Gootenberg, J. S.; Abudayyeh, O. O. *Science* **2024**, *387*, eadr6006.
- (60) Ding, D.; Shaw, A. Y.; Sinai, S.; Rollins, N.; Prywes, N.; Savage, D. F.; Laub, M. T.; Marks, D. S. *Nature Communications* **2024**, *15*, 1639.
- (61) Li, F.-Z.; Yang, J.; Johnston, K. E.; Guersoy, E.; Yue, Y.; Arnold, F. H. *bioRxiv* **2024**, DOI: [10.1101/2024.10.24.619774](https://doi.org/10.1101/2024.10.24.619774).
- (62) Landwehr, G. M.; Bogart, J. W.; Magalhaes, C.; Hammarlund, E. G.; Karim, A. S.; Jewett, M. C. *Nature Communications* **2025**, *16*, 865.
- (63) Sun, H.; He, L.; Deng, P.; Liu, G.; Liu, H.; Cao, C.; Ju, F.; Wu, L.; Qin, T.; Liu, T.-Y. *bioRxiv* **2025**, DOI: [10.1101/2023.11.16.565910](https://doi.org/10.1101/2023.11.16.565910).
- (64) Vornholt, T.; Mutný, M.; Schmidt, G. W.; Schellhaas, C.; Tachibana, R.; Panke, S.; Ward, T. R.; Krause, A.; Jeschek, M. *ACS Central Science* **2024**, *10*, 1357–1370.
- (65) Albanese, K. I.; Barbe, S.; Tagami, S.; Woolfson, D. N.; Schiex, T. *Nature Reviews Methods Primers* **2025**, *5*, 1–28.
- (66) The UniProt Consortium et al. *Nucleic Acids Research* **2023**, *51*, D523–D531.
- (67) Mak, W. S.; Tran, S.; Marcheschi, R.; Bertolani, S.; Thompson, J.; Baker, D.; Liao, J. C.; Siegel, J. B. *Nature Communications* **2015**, *6*, 10005.
- (68) Bileschi, M. L.; Belanger, D.; Bryant, D. H.; Sanderson, T.; Carter, B.; Sculley, D.; Bateman, A.; DePristo, M. A.; Colwell, L. J. *Nature Biotechnology* **2022**, *40*, 932–937.
- (69) Buton, N.; Coste, F.; Le Cunff, Y. *Bioinformatics* **2023**, *39*, ed. by Valencia, A., btad620.
- (70) Derry, A.; Altman, R. B. *Explainable Protein Function Annotation Using Local Structure Embeddings*; Preprint; Bioinformatics, 2023.
- (71) Dickson, A. M.; Mofrad, M. R. K. *Fine-Tuning Protein Embeddings for Generalizable Annotation Propagation*; Preprint; Bioinformatics, 2023.

- (72) Feehan, R.; Franklin, M. W.; Slusky, J. S. G. *Nature Communications* **2021**, *12*, 3712.
- (73) Li, F.; Yuan, L.; Lu, H.; Li, G.; Chen, Y.; Engqvist, M. K. M.; Kerkhoven, E. J.; Nielsen, J. *Nature Catalysis* **2022**, *5*, 662–672.
- (74) Thurimella, K.; Mohamed, A. M. T.; Graham, D. B.; Owens, R. M.; La Rosa, S. L.; Plichta, D. R.; Bacallado, S.; Xavier, R. J. *Protein Language Models Uncover Carbohydrate-Active Enzyme Function in Metagenomics*; Preprint; Bioinformatics, 2023.
- (75) Yu, T.; Cui, H.; Li, J. C.; Luo, Y.; Jiang, G.; Zhao, H. **2023**, 379, 1358–1363.
- (76) Yu, T.; Boob, A. G.; Volk, M. J.; Liu, X.; Cui, H.; Zhao, H. *Nature Catalysis* **2023**, *6*, 137–151.
- (77) Zheng, L.; Shi, S.; Fang, P.; Zhang, H.; Pan, Z.; Huang, S.; Xia, W.; Li, H.; Zeng, Z.; Zhang, S.; Chen, Y.; Lu, M.; Li, Z.; Zhu, F. *AnnoPRO: An Innovative Strategy for Protein Function Annotation Based on Image-like Protein Representation and Multimodal Deep Learning*; Preprint; Bioinformatics, 2023.
- (78) Visani, G. M.; Hughes, M. C.; Hassoun, S. *Bioinformatics* **2021**, *37*, ed. by Martelli, P. L., 2017–2024.
- (79) Fan, W.; Zhou, Y.; Wang, S.; Yan, Y.; Liu, H.; Zhao, Q.; Song, L.; Li, Q. **2025**, DOI: [10.48550/arXiv.2501.10282](https://doi.org/10.48550/arXiv.2501.10282).
- (80) Hsu, C.; Fannjiang, C.; Listgarten, J. *Nature Biotechnology* **2024**, *42*, 196–199.
- (81) Kyro, G. W.; Qiu, T.; Batista, V. S. *arXiv* **2025**, DOI: [10.48550/arXiv.2502.19173](https://doi.org/10.48550/arXiv.2502.19173).
- (82) Wen, S.; Zheng, W.; Bornscheuer, U. T.; Wu, S. *Current Opinion in Green and Sustainable Chemistry* **2025**, *52*, 101010.
- (83) Winnifrith, A.; Outeiral, C.; Hie, B. L. *Current Opinion in Structural Biology* **2024**, *86*, 102794.
- (84) Wu, W. et al. *arXiv* **2025**, DOI: [10.48550/arXiv.2503.13522](https://doi.org/10.48550/arXiv.2503.13522).
- (85) Ferruz, N.; Höcker, B. *Nature Machine Intelligence* **2022**, *4*, 521–532.
- (86) Ruffolo, J. A.; Madani, A. *Nature Biotechnology* **2024**, *42*, 200–202.
- (87) Wu, Z.; Johnston, K. E.; Arnold, F. H.; Yang, K. K. *Current Opinion in Chemical Biology* **2021**, *65*, 18–27.
- (88) Russ, W. P.; Figliuzzi, M.; Stocker, C.; Barrat-Charlaix, P.; Socolich, M.; Kast, P.; Hilvert, D.; Monasson, R.; Cocco, S.; Weigt, M.; Ranganathan, R. *Science* **2020**, *369*, 440–445.

- (89) Alamdari, S.; Thakkar, N.; van den Berg, R.; Tenenholtz, N.; Strome, B.; Moses, A. M.; Lu, A. X.; Fusi, N.; Amini, A. P.; Yang, K. K. *bioRxiv* **2024**, Preprint; version 2 posted Nov 4, 2024, DOI: [10.1101/2023.09.11.556673](https://doi.org/10.1101/2023.09.11.556673).
- (90) Ferruz, N.; Schmidt, S.; Höcker, B. *Nature Communications* **2022**, *13*, 4348.
- (91) Hayes, T. et al. *Science* **2025**, *387*, 850–858.
- (92) Hesslow, D.; Zanichelli, N.; Notin, P.; Poli, I.; Marks, D. *arXiv* **2022**, DOI: [10.48550/arXiv.2205.05789](https://doi.org/10.48550/arXiv.2205.05789).
- (93) Madani, A.; Krause, B.; Greene, E. R.; Subramanian, S.; Mohr, B. P.; Holton, J. M.; Olmos, J. L.; Xiong, C.; Sun, Z. Z.; Socher, R.; Fraser, J. S.; Naik, N. *Nature Biotechnology* **2023**, DOI: [10.1038/s41587-022-01618-2](https://doi.org/10.1038/s41587-022-01618-2).
- (94) Munsamy, G.; Illanes-Vicioso, R.; Funcillo, S.; Lindner, S.; Ayres, G.; Sheehan, L. S.; Moss, S.; Eckhard, U.; Lorenz, P.; Ferruz, N. *bioRxiv* **2024**, DOI: [10.1101/2024.05.03.592223](https://doi.org/10.1101/2024.05.03.592223).
- (95) Wang, X.; Zheng, Z.; Ye, F.; Xue, D.; Huang, S.; Gu, Q. In *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp 52309–52333.
- (96) Ruffolo, J. A.; Nayfach, S.; Gallagher, J.; Bhatnagar, A.; Beazer, J.; Hussain, R.; Yip, J.; Hill, E.; Pacesa, M.; Meeske, A. J.; Cameron, P.; Madani, A. **2024**, DOI: [10.1101/2024.04.22.590591](https://doi.org/10.1101/2024.04.22.590591).
- (97) Buller, R.; Damborsky, J.; Hilvert, D.; Bornscheuer, U. T. *Angewandte Chemie International Edition* **2024**, *64*, e202421686.
- (98) Yeh, A. H.-W. et al. *Nature* **2023**, *614*, 774–780.
- (99) Lauko, A. et al. *Science* **2025**, *388*, eadu2454.
- (100) Anishchenko, I.; Kipnis, Y.; Kalvet, I.; Zhou, G.; Krishna, R.; Pellock, S. J.; Lauko, A.; Lee, G. R.; An, L.; Dauparas, J.; DiMaio, F.; Baker, D. *bioRxiv* **2024**, DOI: [10.1101/2024.09.25.614868](https://doi.org/10.1101/2024.09.25.614868).
- (101) Ingraham, J. B. et al. *Nature* **2023**, DOI: [10.1038/s41586-023-06728-8](https://doi.org/10.1038/s41586-023-06728-8).
- (102) Watson, J. L. et al. *Nature* **2023**, *620*, 1089–1100.
- (103) Wang, Q. et al. *Research* **2024**, *7*, 0413.
- (104) Zheng, Z.; Zhang, B.; Didi, K.; Yang, K. K.; Yim, J.; Watson, J. L.; Chen, H.-F.; Trippe, B. L. **2025**, DOI: [10.48550/arXiv.2502.12479](https://doi.org/10.48550/arXiv.2502.12479).
- (105) Hua, C.; Liu, Y.; Zhang, D.; Zhang, O.; Luan, S.; Yang, K. K.; Wolf, G.; Precup, D.; Zheng, S. *arXiv* **2024**, DOI: [10.48550/arXiv.2410.00327](https://doi.org/10.48550/arXiv.2410.00327).
- (106) Hua, C.; Lu, J.; Liu, Y.; Zhang, O.; Tang, J.; Ying, R.; Jin, W.; Wolf, G.; Precup, D.; Zheng, S. *arXiv* **2024**, DOI: [10.48550/arXiv.2411.16694](https://doi.org/10.48550/arXiv.2411.16694).

- (107) Dauparas, J. et al. *Science* **2022**, 378, 49–56.
- (108) Hsu, C.; Verkuil, R.; Liu, J.; Lin, Z.; Hie, B.; Sercu, T.; Lerer, A.; Rives, A. In *Proceedings of the 39th International Conference on Machine Learning*, 2022; Vol. 162, pp 8946–8970.
- (109) Ahern, W.; Yim, J.; Tischer, D.; Salike, S.; Woodbury, S.; Kim, D.; Kalvet, I.; Kipnis, Y.; Coventry, B.; Altae-Tran, H.; Bauer, M.; Barzilay, R.; Jaakkola, T.; Krishna, R.; Baker, D. A. *bioRxiv* **2025**, DOI: [10.1101/2025.04.09.648075](https://doi.org/10.1101/2025.04.09.648075).
- (110) Campbell, A.; Yim, J.; Barzilay, R.; Rainforth, T.; Jaakkola, T. In *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp 5453–5512.
- (111) Chu, A. E.; Kim, J.; Cheng, L.; El Nesr, G.; Xu, M.; Shuai, R. W.; Huang, P.-S. *Proceedings of the National Academy of Sciences* **2024**, 121, e2311500121.
- (112) Geffner, T.; Didi, K.; Cao, Z.; Reidenbach, D.; Zhang, Z.; Dallago, C.; Kucukbenli, E.; Kreis, K.; Vahdat, A. *arXiv* **2025**, DOI: [10.48550/arXiv.2507.09466](https://doi.org/10.48550/arXiv.2507.09466).
- (113) Huguet, G.; Vuckovic, J.; Fatras, K.; Thibodeau-Laufer, E.; Lemos, P.; Islam, R.; Liu, C.-H.; Rector-Brooks, J.; Akhound-Sadegh, T.; Bronstein, M.; Tong, A.; Bose, A. J. In *Advances in Neural Information Processing Systems*, 2024.
- (114) Lisanza, S. L. et al. *Nature Biotechnology* **2024**, DOI: [10.1038/s41587-024-02395-w](https://doi.org/10.1038/s41587-024-02395-w).
- (115) Lu, A. X.; Yan, W.; Robinson, S. A.; Yang, K. K.; Gligorijevic, V.; Cho, K.; Bonneau, R.; Abbeel, P.; Frey, N. *bioRxiv* **2025**, DOI: [10.1101/2024.12.02.626353](https://doi.org/10.1101/2024.12.02.626353).
- (116) Wang, X.; Zheng, Z.; Ye, F.; Xue, D.; Huang, S.; Gu, Q. In *13th International Conference on Learning Representations*, 2025.
- (117) Wang, C.; Alamdari, S.; Domingo-Enrich, C.; Amini, A.; Yang, K. K. *Current Opinion in Structural Biology* **2025**, 91, 103018.
- (118) Jumper, J. et al. *Nature* **2021**, 596, 583–589.
- (119) Lin, Z.; Akin, H.; Rao, R.; Hie, B.; Zhu, Z.; Lu, W.; Smetanin, N.; Verkuil, R.; Kabeli, O.; Shmueli, Y.; Fazel-Zarandi, M.; Sercu, T.; Candido, S.; Rives, A. *Science* **2023**, 379, 1123–1130.
- (120) Su, J.; Zhou, X.; Zhang, X.; Yuan, F. *bioRxiv* **2024**, DOI: [10.1101/2024.05.30.596740](https://doi.org/10.1101/2024.05.30.596740).
- (121) Yang, J.; Mora, A.; Liu, S.; Wittmann, B. J.; Anandkumar, A.; Arnold, F. H.; Yue, Y. In *Advances in Neural Information Processing Systems*, 2024.
- (122) Zhou, X.; Han, C.; Zhang, Y.; Su, J.; Zhuang, K.; Jiang, S.; Yuan, Z.; Zheng, W.; Dai, F.; Zhou, Y.; Tao, Y.; Wu, D.; Yuan, F. *bioRxiv* **2025**, DOI: [10.1101/2025.01.05.630192](https://doi.org/10.1101/2025.01.05.630192).

- (123) Dai, F.; Fan, Y.; Su, J.; Wang, C.; Han, C.; Zhou, X.; Liu, J.; Qian, H.; Wang, S.; Zeng, A.; Wang, Y.; Yuan, F. *bioRxiv* **2025**, DOI: [10.1101/2024.08.01.606258](https://doi.org/10.1101/2024.08.01.606258).
- (124) Liu, S.; Zhu, Y.; Lu, J.; Xu, Z.; Nie, W.; Gitter, A.; Xiao, C.; Tang, J.; Guo, H.; Anandkumar, A. *Nature Machine Intelligence* **2025**, 7, 580–591.
- (125) Yang, J.; Bhatnagar, A.; Ruffolo, J. A.; Madani, A. *arXiv* **2025**, DOI: [10.48550/arXiv.2410.03634](https://doi.org/10.48550/arXiv.2410.03634).
- (126) Notin, P.; Rollins, N.; Gal, Y.; Sander, C.; Marks, D. *Nature Biotechnology* **2024**, 42, 216–228.
- (127) Aghazadeh, A.; Nisonoff, H.; Ocal, O.; Brookes, D. H.; Huang, Y.; Koyluoglu, O. O.; Listgarten, J.; Ramchandran, K. *Nature Communications* **2021**, 12, 5225.
- (128) Freschlin, C. R.; Fahlberg, S. A.; Romero, P. A. *Current Opinion in Biotechnology* **2022**, 75, 102713.
- (129) Wittmann, B. J.; Johnston, K. E.; Wu, Z.; Arnold, F. H. *Current Opinion in Structural Biology* **2021**, 69, Publisher: Elsevier Ltd, 11–18.
- (130) Greenman, K. P.; Amini, A. P.; Yang, K. K. **2023**.
- (131) Hie, B. L.; Yang, K. K. *Current Opinion in Structural Biology* **2022**, 72, 145–152.
- (132) Hie, B.; Bryson, B. D.; Berger, B. *Cell Systems* **2020**, 11, 461–477.e9.
- (133) Qiu, Y.; Wei, G.-W. *Journal of Chemical Information and Modeling* **2022**, 62, 4629–4641.
- (134) Qiu, Y.; Hu, J.; Wei, G.-W. *Nature Computational Science* **2021**, 1, 809–818.
- (135) Rapp, J. T.; Bremer, B. J.; Romero, P. A. *Nature Chemical Engineering* **2024**, 1, 97–107.
- (136) Bal, M. I.; Sessa, P. G.; Mutny, M.; Krause, A. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*; Poster, Singapore, 2025.
- (137) Hu, R.; Fu, L.; Chen, Y.; Chen, J.; Qiao, Y.; Si, T. *Briefings in Bioinformatics* **2023**, 24, bbac570.
- (138) Gantz, M. et al. *bioRxiv* **2024**, DOI: [10.1101/2024.04.08.588565](https://doi.org/10.1101/2024.04.08.588565).
- (139) Abe, T.; Buchanan, E. K.; Pleiss, G.; Zemel, R.; Cunningham, J. P. **2022**.
- (140) Bowden, J.; Yeh, C. T.; Astudillo, R.; Song, J.; Chen, Y.; Desautels, T. A.; Yue, Y. *NeurIPS* **2024**.
- (141) Wilson, A. G.; Hu, Z.; Salakhutdinov, R.; Xing, E. P. **2015**.

- (142) Eriksson, D.; Pearce, M.; Gardner, J. R.; Turner, R.; Poloczec, M. **2020**.
- (143) A. Elnaggar; M. Heinzinger; C. Dallago; G. Rehawi; Y. Wang; L. Jones; T. Gibbs; T. Feher; C. Angerer; M. Steinegger; D. Bhowmik; B. Rost *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2022**, *44*, 7112–7127.
- (144) Coelho, P. S.; Brustad, E. M.; Kannan, A.; Arnold, F. H. *Science* **2013**, *339*, 307–310.
- (145) Pesce, A.; Bolognesi, M.; Nardini, M. In *Advances in Microbial Physiology*; Elsevier: 2013; Vol. 63, pp 79–96.
- (146) Gao, S.; Das, A.; Alfonzo, E.; Sicinski, K. M.; Rieger, D.; Arnold, F. H. *Journal of the American Chemical Society* **2023**, *145*, 20196–20201.
- (147) Hanley, D.; Li, Z.-Q.; Gao, S.; Virgil, S. C.; Arnold, F. H.; Alfonzo, E. *Journal of the American Chemical Society* **2024**, *146*, 19160–19167.
- (148) Porter, N. J.; Danelius, E.; Gonen, T.; Arnold, F. H. *Journal of the American Chemical Society* **2022**, *144*, 8892–8896.
- (149) Park, Y.; Metzger, B. P. H.; Thornton, J. W. *Science* **2022**, *376*, 823–830.
- (150) Hopf, T. A.; Ingraham, J. B.; Poelwijk, F. J.; Schärfe, C. P. I.; Springer, M.; Sander, C.; Marks, D. S. *Nature Biotechnology* **2017**, *35*, 128–135.
- (151) Long, Y.; Mora, A.; Li, F.-Z.; Gürsoy, E.; Johnston, K. E.; Arnold, F. H. *ACS Synthetic Biology* **2025**, *14*, 230–238.
- (152) Wu, N. C.; Dai, L.; Olson, C. A.; Lloyd-Smith, J. O.; Sun, R. *eLife* **2016**, *5*, e16965.
- (153) Johnston, K. E.; Almhjell, P. J.; Watkins-Dulaney, E. J.; Liu, G.; Porter, N. J.; Yang, J.; Arnold, F. H. *Proceedings of the National Academy of Sciences* **2024**, *121*, e2400439121.
- (154) Rao, R.; Bhattacharya, N.; Thomas, N.; Duan, Y.; Chen, X.; Canny, J.; Abbeel, P.; Song, Y. S. **2019**.
- (155) Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C. L.; Ma, J.; Fergus, R. *Proceedings of the National Academy of Sciences* **2021**, *118*, DOI: [10.1073/pnas.2016239118](https://doi.org/10.1073/pnas.2016239118).
- (156) Hvarfner, C.; Hellsten, E. O.; Nardi, L. **2024**.
- (157) Michael, R.; Kæstel-Hansen, J.; Mørch Groth, P.; Bartels, S.; Salomon, J.; Tian, P.; Hatzakis, N. S.; Boomsma, W. *PLOS Computational Biology* **2024**, *20*, ed. by Fariselli, P., e1012061.
- (158) Letham, B.; Karrer, B.; Ottoni, G.; Bakshy, E. **2018**.
- (159) Luo, Y.; Liu, Y.; Peng, J. *Nature Machine Intelligence* **2023**.

- (160) Stanton, S.; Maddox, W.; Wilson, A. G., 206, 959–986.
- (161) Fannjiang, C.; Bates, S.; Angelopoulos, A. N.; Listgarten, J.; Jordan, M. I. *Proceedings of the National Academy of Sciences* **2022**, *119*, e2204569119.
- (162) Fannjiang, C.; Listgarten, J. *Cold Spring Harbor Perspectives in Biology* **2024**, *16*, a041469.
- (163) Ober, S. W.; Rasmussen, C. E.; van der Wilk, M. *arXiv* **2021**.
- (164) Hollmann, F.; Sanchis Martinez, J.; Reetz, M. T. *Angewandte Chemie International Edition* **2024**, e202404880.
- (165) Wilson, A. G.; Izmailov, P. **2022**.
- (166) Wittmann, B. J.; Johnston, K. E.; Almhjell, P. J.; Arnold, F. H. *ACS Synthetic Biology* **2022**, *11*, 1313–1324.
- (167) Yang, J.; Ducharme, J.; Johnston, K. E.; Li, F.-Z.; Yue, Y.; Arnold, F. H. *ACS Synthetic Biology* **2023**, acssynbio.3c00301.
- (168) Radivojac, P. et al. *Nature Methods* **2013**, *10*, 221–227.
- (169) Price, M. N. et al. *Nature* **2018**, *557*, 503–509.
- (170) Pavlopoulos, G. A. et al. *Nature* **2023**, *622*, 594–602.
- (171) Ayres, G.; Munsamy, G.; Heinzinger, M.; Ferruz, N.; Yang, K.; Lorenz, P. **2024**.
- (172) Jain, S.; Ospina, F.; Hammer, S. C. *JACS Au* **2024**, jacsau.4c00247.
- (173) Altschul, S. F.; Gish, W.; Miller, W.; Myers, E. W.; Lipman, D. J. *Journal of Molecular Biology* **1990**.
- (174) Finn, R. D.; Clements, J.; Arndt, W.; Miller, B. L.; Wheeler, T. J.; Schreiber, F.; Bateman, A.; Eddy, S. R. *Nucleic Acids Research* **2015**, *43*, W30–W38.
- (175) Van Kempen, M.; Kim, S. S.; Tumescheit, C.; Mirdita, M.; Lee, J.; Gilchrist, C. L.; Söding, J.; Steinegger, M. *Nature Biotechnology* **2023**, *42*, 243–246.
- (176) Gilchrist, C. L. M.; Mirdita, M.; Steinegger, M. **2024**.
- (177) Kouba, P.; Kohout, P.; Haddadi, F.; Bushuiev, A.; Samusevich, R.; Sedlar, J.; Damborsky, J.; Pluskal, T.; Sivic, J.; Mazurenko, S. *ACS Catalysis* **2023**, *13*, 13863–13895.
- (178) Johnston, K. E.; Fannjiang, C.; Wittmann, B. J.; Hie, B. L.; Yang, K. K.; Wu, Z. Machine Learning for Protein Engineering, arXiv:2305.16634 [q-bio], 2023.
- (179) Ramakrishnan, P.; Bromberg, Y. Functional profiling of the sequence stockpile: a review and assessment of in silico prediction tools, en, 2023.
- (180) Ryu, J. Y.; Kim, H. U.; Lee, S. Y. *Proceedings of the National Academy of Sciences* **2019**, *116*, 13996–14001.

- (181) Gligorijević, V.; Renfrew, P. D.; Kosciolk, T.; Leman, J. K.; Berenberg, D.; Vatanen, T.; Chandler, C.; Taylor, B. C.; Fisk, I. M.; Vlamakis, H.; Xavier, R. J.; Knight, R.; Cho, K.; Bonneau, R. *Nature Communications* **2021**, *12*, 1–14.
- (182) Sanderson, T.; Bileschi, M. L.; Belanger, D.; Colwell, L. J. *eLife* **2023**, *12*, e80942.
- (183) Shaw, P.; Gurram, B.; Belanger, D.; Gane, A.; Bileschi, M. L.; Colwell, L. J.; Toutanova, K.; Parikh, A. P. **2024**.
- (184) Bennett, J.; Blumenthal, D. B.; Grimm, D. G.; Haselbeck, F.; Joeres, R.; Kalinina, O. V.; List, M. *Nature Methods* **2024**, *21*, 1444–1453.
- (185) De Crecy-Lagard, V.; Dias, R.; Friedberg, I.; Yuan, Y.; Swairjo, M. *bioRxiv* **2024**, DOI: [10.1101/2024.07.01.601547](https://doi.org/10.1101/2024.07.01.601547).
- (186) Mistry, J.; Chuguransky, S.; Williams, L.; Qureshi, M.; Salazar, G. A.; Sonnhammer, E. L. L.; Tosatto, S. C. E.; Paladin, L.; Raj, S.; Richardson, L. J.; Finn, R. D.; Bateman, A. *Nucleic Acids Research* **2021**, *49*, D412–D419.
- (187) Abramson, J. et al. *Nature* **2024**, DOI: [10.1038/s41586-024-07487-w](https://doi.org/10.1038/s41586-024-07487-w).
- (188) Varadi, M. et al. *Nucleic Acids Research* **2024**, *52*, D368–D375.
- (189) Chang, A.; Jeske, L.; Ulbrich, S.; Hofmann, J.; Koblitz, J.; Schomburg, I.; Neumann-Schaal, M.; Jahn, D.; Schomburg, D. *Nucleic Acids Research* **2021**, *49*, D498–D508.
- (190) Bansal, P.; Morgat, A.; Axelsen, K. B.; Muthukrishnan, V.; Coudert, E.; Aimo, L.; Hyka-Nouspikel, N.; Gasteiger, E.; Kerhornou, A.; Neto, T. B.; Pozzato, M.; Blatter, M.-C.; Ignatchenko, A.; Redaschi, N.; Bridge, A. *Nucleic Acids Research* **2022**, *50*, D693–D700.
- (191) Kanehisa, M.; Furumichi, M.; Sato, Y.; Kawashima, M.; Ishiguro-Watanabe, M. *Nucleic Acids Research* **2023**, *51*, D587–D592.
- (192) Probst, D.; Manica, M.; Nana Teukam, Y. G.; Castrogiovanni, A.; Paratore, F.; Laino, T. *Nature Communications* **2022**, *13*, 964.
- (193) Heid, E.; Probst, D.; Green, W. H.; Madsen, G. K. H. *Chemical Science* **2023**, *14*, 14229–14242.
- (194) Finnigan, W.; Hepworth, L. J.; Flitsch, S. L.; Turner, N. J. *Nature Catalysis* **2021**, *4*, 98–104.
- (195) Finnigan, W.; Lubberink, M.; Hepworth, L. J.; Citoler, J.; Matthey, A. P.; Ford, G. J.; Sangster, J.; Cosgrove, S. C.; Da Costa, B. Z.; Heath, R. S.; Thorpe, T. W.; Yu, Y.; Flitsch, S. L.; Turner, N. J. *ACS Catalysis* **2023**, *13*, 11771–11780.

- (196) Levin, I.; Liu, M.; Voigt, C. A.; Coley, C. W. *Nature Communications* **2022**, *13*, 7747.
- (197) Carbonell, P.; Wong, J.; Swainston, N.; Takano, E.; Turner, N. J.; Scrutton, N. S.; Kell, D. B.; Breitling, R.; Faulon, J.-L. *Bioinformatics* **2018**, *34*, ed. by Stegle, O., 2153–2154.
- (198) Dallago, C.; Mou, J.; Johnston, K. E.; Wittmann, B. J.; Bhattacharya, N.; Goldman, S.; Madani, A.; Yang, K. K. *FLIP: Benchmark tasks in fitness landscape inference for proteins*; tech. rep.; 2021.
- (199) Notin, P.; Kollasch, A. W.; Ritter, D.; van Niekerk, L.; Paul, S.; Spinner, H.; Rollins, N.; Shaw, A.; Weitzman, R.; Frazer, J.; Dias, M.; Franceschi, D.; Orenbuch, R.; Gal, Y.; Marks, D. S. **2023**.
- (200) Mikhael, P. G.; Chinn, I.; Barzilay, R. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.
- (201) Schwaller, P.; Laino, T.; Gaudin, T.; Bolgar, P.; Hunter, C. A.; Bekas, C.; Lee, A. A. *ACS Central Science* **2019**, *5*, 1572–1583.
- (202) Kreutter, D.; Schwaller, P.; Reymond, J.-L. *Chemical Science* **2021**, *12*, 8648–8659.
- (203) Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rehawi, G.; Wang, Y.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; Bhowmik, D.; Rost, B. **2021**, *14*, 29.
- (204) Robinson, L. C. B.; Atkinson, T.; Copoiu, L.; Bordes, P.; Pierrot, T.; Barrett, T. *Contrasting Sequence with Structure: Pre-training Graph Representations with PLMs*; preprint; Bioinformatics, 2023.
- (205) Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised Contrastive Learning, en, arXiv:2004.11362 [cs, stat], 2021.
- (206) Duan, H.; Li, Z.; Wu, Y.; Chen, W.; Xia, L. C. **2024**.
- (207) Jang, Y. J. *Nature Communications* **2024**, *15*, 6601.
- (208) Song, Y.; Yuan, Q.; Chen, S.; Zeng, Y.; Zhao, H.; Yang, Y. *Nature Communications* **2024**, *15*, 8180.
- (209) Zheng, L.; Li, B.; Xu, S.; Chen, J.; Liang, G. *bioRxiv* **2024**.
- (210) Boger, R. S.; Chithrananda, S.; Angelopoulos, A. N.; Yoon, P. H.; Jordan, M. I.; Doudna, J. A. **2024**, DOI: [10.1101/2024.06.27.601042](https://doi.org/10.1101/2024.06.27.601042).
- (211) Hong, L.; Hu, Z.; Sun, S.; Tang, X.; Wang, J.; Tan, Q.; Zheng, L.; Wang, S.; Xu, S.; King, I.; Gerstein, M.; Li, Y. *Nature Biotechnology* **2024**, DOI: [10.1038/s41587-024-02353-6](https://doi.org/10.1038/s41587-024-02353-6).
- (212) Hamamsy, T.; Barot, M.; Morton, J. T.; Steinegger, M.; Bonneau, R.; Cho, K. **2023**, DOI: [10.1101/2023.11.26.568742](https://doi.org/10.1101/2023.11.26.568742).

- (213) Hamamsy, T.; Morton, J. T.; Blackwell, R.; Berenberg, D.; Carriero, N.; Gligorijevic, V.; Strauss, C. E. M.; Leman, J. K.; Cho, K.; Bonneau, R. *Nature Biotechnology* **2023**, DOI: [10.1038/s41587-023-01917-2](https://doi.org/10.1038/s41587-023-01917-2).
- (214) Campbell, M. J. **2024**, DOI: [10.1101/2024.06.21.599972](https://doi.org/10.1101/2024.06.21.599972).
- (215) Kroll, A.; Ranjan, S.; Engqvist, M. K. M.; Lercher, M. J. *Nature Communications* **2023**, *14*, 2787.
- (216) Paton, A.; Boiko, D.; Perkins, J.; Cemalovic, N.; Reschützegger, T.; Gomes, G.; Narayan, A. *chemRxiv* **2024**, DOI: [10.26434/chemrxiv-2024-w4dtr](https://doi.org/10.26434/chemrxiv-2024-w4dtr).
- (217) Zhang, Q. et al. Scientific Large Language Models: A Survey on Biological & Chemical Domains, en, arXiv:2401.14656 [cs], 2024.
- (218) Ramos, M. C.; Collison, C. J.; White, A. D. A Review of Large Language Models and Autonomous Agents in Chemistry, en, arXiv:2407.01603 [physics], 2024.
- (219) Carrami, E. M.; Sharifzadeh, S. PQA: Zero-shot Protein Question Answering for Free-form Scientific Enquiry with Large Language Models, en, arXiv:2402.13653 [cs], 2024.
- (220) Wang, Z.; Zhang, Q.; Ding, K.; Qin, M.; Zhuang, X.; Li, X.; Chen, H. InstructProtein: Aligning Human and Protein Language via Knowledge Instruction, en, arXiv:2310.03269 [cs, q-bio], 2023.
- (221) Zhuang, X.; Ding, K.; Lyu, T.; Jiang, Y.; Li, X.; Xiang, Z.; Wang, Z.; Qin, M.; Feng, K.; Wang, J.; Zhang, Q.; Chen, H. *bioRxiv* **2024**, arXiv:2410.07919 [cs, q-bio].
- (222) Xiao, Y.; Sun, E.; Jin, Y.; Wang, Q.; Wang, W. *arXiv* **2024**, arXiv:2408.11363 [cs, q-bio].
- (223) Huo, M.; Guo, H.; Cheng, X.; Singh, D.; Rahmani, H.; Li, S.; Gerlof, P.; Ideker, T.; Grotjahn, D. A.; Villa, E.; Song, L.; Xie, P. *arXiv* **2024**.
- (224) Bran, A. M.; Cox, S.; Schilter, O.; Baldassari, C.; White, A. D.; Schwaller, P. ChemCrow: Augmenting large-language models with chemistry tools, en, arXiv:2304.05376 [physics, stat], 2023.
- (225) Laurent, J. M.; Janizek, J. D.; Ruzo, M.; Hinks, M. M.; Hammerling, M. J.; Narayanan, S.; Ponnampati, M.; White, A. D.; Rodriques, S. G. LAB-Bench: Measuring Capabilities of Language Models for Biology Research, en, arXiv:2407.10362 [cs], 2024.
- (226) Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. *arXiv:2002.05709 [cs, stat]* **2020**, arXiv: 2002.05709.
- (227) Liu, S.; Du, W.; Ma, Z.; Guo, H.; Tang, J. **2023**.
- (228) Liu, S.; Nie, W.; Wang, C.; Lu, J.; Qiao, Z.; Liu, L.; Tang, J.; Xiao, C.; Anandkumar, A. *Nature Machine Intelligence* **2023**, *5*, 1447–1457.

- (229) Liu, S.; Li, Y.; Li, Z.; Gitter, A.; Zhu, Y.; Lu, J.; Xu, Z.; Nie, W.; Ramanathan, A.; Xiao, C.; Tang, J.; Guo, H.; Anandkumar, A. A Text-guided Protein Design Framework, en, arXiv preprint arXiv:2302.04611, 2023.
- (230) Wu, K. E.; Chang, H.; Zou, J. **2024**.
- (231) Singh, R.; Sledzieski, S.; Bryson, B.; Cowen, L.; Berger, B. *Proceedings of the National Academy of Sciences* **2023**, *120*, e2220778120.
- (232) Probst, D.; Schwaller, P.; Reymond, J.-L. *Digital Discovery* **2022**, *1*, 91–97.
- (233) Rost, B. *Protein Engineering, Design and Selection* **1999**, *12*, 85–94.
- (234) Richman, T.; Arnold, E.; Williams, A. J. *Scientific Data* **2022**, *9*, 180.
- (235) Samusevich, R.; Hebra, T.; Bushuiev, R.; Bushuiev, A.; Čalounová, T.; Smrčková, H.; Chatpatanasiri, R.; Kulhánek, J.; Perković, M.; Engst, M.; Tajovská, A.; Sivic, J.; Pluskal, T. Highly accurate discovery of terpene synthases powered by machine learning reveals functional terpene cyclization in Archaea, en, 2024.
- (236) Samusevich, R.; Hebra, T.; Bushuiev, R.; Bushuiev, A.; Kulhánek, J.; Čalounová, T.; Perković, M.; Tajovská, A.; Sivic, J.; Pluskal, T. **2024**.
- (237) Hirota, K.; Salim, F.; Yamada, T. **2024**.
- (238) Schwaller, P.; Probst, D.; Vaucher, A. C.; Nair, V. H.; Kreutter, D.; Laino, T.; Reymond, J.-L. *Nature Machine Intelligence* **2021**, *3*, 144–152.
- (239) Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. *arXiv:1810.04805 [cs]* **2019**, arXiv: 1810.04805.
- (240) Ashburner, M. et al. *Nature Genetics* **2000**, *25*, 25–29.
- (241) Beltagy, I.; Lo, K.; Cohan, A. SciBERT: A Pretrained Language Model for Scientific Text, en, arXiv:1903.10676 [cs], 2019.
- (242) Duan, H.; Skreta, M.; Cotta, L.; Rajaonson, E. M.; Dhawan, N.; Aspuru-Guzik, A.; Maddison, C. J. **2024**.
- (243) Hua, C.; Zhong, B.; Luan, S.; Hong, L.; Wolf, G.; Precup, D.; Zheng, S. *NeurIPS* **2024**.
- (244) Liu, Y.; Hua, C.; Zeng, T.; Rao, J.; Zhang, Z.; Wu, R.; Coley, C. W.; Zheng, S. *bioRxiv* **2024**.
- (245) Steinegger, M.; Söding, J. *Nature Biotechnology* **2017**, *35*, Number: 11 Publisher: Nature Publishing Group, 1026–1028.
- (246) Merchant, A. T.; King, S. H.; Nguyen, E.; Hie, B. L. *bioRxiv* **2024**.
- (247) Tavakoli, M.; Shmakov, A.; Ceccarelli, F.; Baldi, P. Rxn Hypergraph: a Hypergraph Attention Model for Chemical Reaction Representation, en, arXiv:2201.01196 [physics], 2022.

- (248) Char, S.; Corley, N.; Alamdari, S.; Yang, K. K.; Amini, A. P. *bioRxiv* **2024**.
- (249) Yang, J.; Li, F.-Z.; Long, Y.; Arnold, F. H. *Cell Systems* **2026**, *17*, 101372.
- (250) Yang, J.; Lal, R. G.; Bowden, J. C.; Astudillo, R.; Hameedi, M. A.; Kaur, S.; Hill, M.; Yue, Y.; Arnold, F. H. *Nature Communications* **2025**, *16*, 714.
- (251) Sumida, K. H. et al. *Journal of the American Chemical Society* **2024**, *146*, 2054–2061.
- (252) Fei, H.; Li, Y.; Liu, Y.; Wei, J.; Chen, A.; Gao, C. *Cell* **2025**, *188*, 4674–4692.e19.
- (253) Seki, K.; Guo, A. B.; Akpinaroglu, D.; Kortemme, T. *bioRxiv* **2025**, Preprint; version 1 posted Aug 3, 2025, 2025.08.03.668353.
- (254) Lambert, T.; Tavakoli, A.; Dharuman, G.; Yang, J.; Bhethanabotla, V.; Kaur, S.; Hill, M.; Ramanathan, A.; Anandkumar, A.; Arnold, F. H. *bioRxiv* **2025**, Preprint; version 1 posted Aug 30, 2025, DOI: [10.1101/2025.08.30.673177](https://doi.org/10.1101/2025.08.30.673177).
- (255) Song, Z.; Li, L. In *Proceedings of the 40th International Conference on Machine Learning*, ed. by Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; Scarlett, J.; ICML 2023, PMLR: Honolulu, Hawaii, USA, 2023; Vol. 202, pp 32349–32364.
- (256) Stanton, S.; Maddox, W.; Gruver, N.; Maffettone, P.; Delaney, E.; Greenside, P.; Wilson, A. G. In *Proceedings of the 39th International Conference on Machine Learning*, PMLR: 2022; Vol. 162, pp 20459–20478.
- (257) Gupta, A.; Zou, J. *Nature Machine Intelligence* **2019**, *1*, Number: 2 Publisher: Nature Publishing Group, 105–111.
- (258) Brookes, D. H.; Listgarten, J. Design by adaptive sampling, en, arXiv:1810.03714 [cs, q-bio, stat], 2020.
- (259) Jain, M.; Bengio, E.; Hernandez-Garcia, A.; Rector-Brooks, J.; Dossou, B. F. P.; Ekbote, C.; Fu, J.; Zhang, T.; Kilgour, M.; Zhang, D.; Simine, L.; Das, P.; Bengio, Y. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, PMLR: 2022; Vol. 162, pp 9786–9801.
- (260) Kim, H.; Kim, M.; Yun, T.; Choi, S.; Bengio, E.; Hernández-García, A.; Park, J. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*; ICML 2025 (poster), to appear, Vancouver, Canada, 2025.
- (261) Angermueller, C.; Dohan, D.; Belanger, D.; Deshpande, R.; Murphy, K.; Colwell, L. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*; ICLR 2020, 2020.
- (262) Hawkins-Hooker, A.; Kmec, J.; Bent, O.; Duckworth, P. In *ICML 2024 Workshop on Machine Learning for Life and Material Science: From Theory to Industry Applications (ML4LMS)*; Workshop poster; non-archival, Vienna, Austria, 2024.

- (263) Zhao, J.; Zhang, C.; Luo, Y. *Contrastive Fitness Learning: Reprogramming Protein Language Models for Low-N Learning of Protein Fitness Landscape*; preprint; Bioinformatics, 2024.
- (264) Sun, H.; He, L.; Deng, P.; Liu, G.; Zhao, Z.; Jiang, Y.; Cao, C.; Ju, F.; Wu, L.; Liu, H.; Qin, T.; Liu, T.-Y. *Nature Machine Intelligence* **2025**, DOI: [10.1038/s42256-025-01103-w](https://doi.org/10.1038/s42256-025-01103-w).
- (265) Nisonoff, H.; Xiong, J.; Allenspach, S.; Listgarten, J. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*; ICLR 2025 — Poster, Singapore, 2025.
- (266) Stark, H.; Jing, B.; Wang, C.; Corso, G.; Berger, B.; Barzilay, R.; Jaakkola, T. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, PMLR: Vienna, Austria, 2024; Vol. 235, pp 46495–46513.
- (267) Klarner, L.; Rudner, T. G. J.; Morris, G. M.; Deane, C. M.; Teh, Y. W. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, PMLR: 2024; Vol. 235, pp 24770–24807.
- (268) Gruver, N.; Stanton, S.; Frey, N. C.; Rudner, T. G. J.; Hotzel, I.; Lafrance-Vanasse, J.; Rajpal, A.; Cho, K.; Wilson, A. G. In *Advances in Neural Information Processing Systems*; NeurIPS 2023, 2023; Vol. 36.
- (269) Goel, S.; Thoutam, V.; Marroquin, E. M.; Gokaslan, A.; Firouzbakht, A.; Vincoff, S.; Kuleshov, V.; Kratochvil, H. T.; Chatterjee, P. In *ICLR 2025 Workshop on Learning Meaningful Representations of Life (LMRL)*; Workshop poster; non-archival, Singapore, 2025.
- (270) Widatalla, T.; Rafailov, R.; Hie, B. *bioRxiv* **2024**, DOI: [10.1101/2024.05.20.595026](https://doi.org/10.1101/2024.05.20.595026).
- (271) Stocco, F.; Artigues-Lleixà, M.; Hunklinger, A.; Widatalla, T.; Güell, M.; Ferruz, N. *arXiv* **2025**, DOI: [10.48550/arXiv.2412.12979](https://doi.org/10.48550/arXiv.2412.12979).
- (272) Blalock, N.; Seshadri, S.; Babbar, A.; Fahlberg, S. A.; Kulkarni, A.; Romero, P. A. *bioRxiv* **2025**, Preprint, DOI: [10.1101/2025.05.02.651993](https://doi.org/10.1101/2025.05.02.651993).
- (273) Wang, Z.; Fan, J.; Guo, R.; Nguyen, T.; Ji, H.; Liu, G. ProteinZero: Self-Improving Protein Generation via Online Reinforcement Learning, *arXiv preprint*, 2025.
- (274) Brookes, D. H.; Park, H.; Listgarten, J. In *International Conference on Machine Learning*, 2019.
- (275) Huang, L.-K.; Zhu, R.; He, B.; Yao, J. In *Proceedings of the 42nd International Conference on Machine Learning*, ed. by Singh, A.; Fazel, M.; Hsu, D.; Lacoste-Julien, S.; Berkenkamp, F.; Maharaj, T.; Wagstaff, K.; Zhu, J., PMLR: 2025; Vol. 267, pp 26247–26260.
- (276) Torres, M. D. T.; Zeng, Y.; Wan, F.; Maus, N.; Gardner, J.; de la Fuente-Nunez, C. *bioRxiv* **2024**, preprint, DOI: [10.1101/2024.11.27.625757](https://doi.org/10.1101/2024.11.27.625757).

- (277) Kirjner, A.; Yim, J.; Samusevich, R.; Bracha, S.; Jaakkola, T.; Barzilay, R.; Fiete, I. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.
- (278) Sinai, S.; Wang, R.; Whatley, A.; Slocum, S.; Locane, E.; Kelsic, E. D. AdaLead: A simple and robust adaptive greedy search algorithm for sequence design, 2020.
- (279) Ren, Z.; Li, J.; Ding, F.; Zhou, Y.; Ma, J.; Peng, J. In *Proceedings of the 39th International Conference on Machine Learning*, ed. by Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; Sabato, S., PMLR: 2022; Vol. 162, pp 18520–18536.
- (280) Zhang, B.; Chu, W.; Berner, J.; Meng, C.; Anandkumar, A.; Song, Y. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; CVPR 2025; arXiv:2407.01521, IEEE: 2025.
- (281) Nijkamp, E.; Ruffolo, J. A.; Weinstein, E. N.; Naik, N.; Madani, A. *Cell Systems* **2023**, *14*, 968–978.e3.
- (282) Ho, J.; Jain, A.; Abbeel, P. In *Advances in Neural Information Processing Systems (NeurIPS)*; NeurIPS 2020, 2020; Vol. 33.
- (283) Li, T.; Chen, M.; Guo, B.; Shen, Z. *arXiv preprint arXiv:2508.10875* **2025**, Preprint, DOI: [10.48550/arXiv.2508.10875](https://doi.org/10.48550/arXiv.2508.10875).
- (284) Sahoo, S. S.; Arriola, M.; Schiff, Y.; Gokaslan, A.; Marroquin, E.; Chiu, J. T.; Rush, A.; Kuleshov, V. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2024; Vol. 37.
- (285) Lou, A.; Meng, C.; Ermon, S. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, PMLR: Vienna, Austria, 2024; Vol. 235, pp 32819–32848.
- (286) Nie, S.; Zhu, F.; You, Z.; Zhang, X.; Ou, J.; Hu, J.; Zhou, J.; Lin, Y.; Wen, J.-R.; Li, C. In *Advances in Neural Information Processing Systems (NeurIPS)*, San Diego, CA, USA, 2025.
- (287) Shi, J.; Han, K.; Wang, Z.; Doucet, A.; Titsias, M. K. In 2024.
- (288) Li, X. L.; Thickstun, J.; Gulrajani, I.; Liang, P.; Hashimoto, T. B. In *Advances in Neural Information Processing Systems (NeurIPS)*; NeurIPS 2022, 2022; Vol. 35.
- (289) Chen, T.; Zhang, R.; Hinton, G. E. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*; ICLR 2023, 2023.
- (290) Dieleman, S.; Sartran, L.; Roshannai, A.; Savinov, N.; Ganin, Y.; Richemond, P. H.; Doucet, A.; Strudel, R.; Dyer, C.; Durkan, C.; Hawthorne, C.; Leblond, R.; Grathwohl, W.; Adler, J. *arXiv preprint arXiv:2211.15089* **2022**, Preprint, DOI: [10.48550/arXiv.2211.15089](https://doi.org/10.48550/arXiv.2211.15089).

- (291) Meshchaninov, V.; Strashnov, P.; Shevtsov, A.; Nikolaev, F.; Ivanisenko, N.; Kardymon, O.; Vetrov, D. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*; Poster; to appear, Vancouver, Canada, 2025.
- (292) Chen, T.; Vure, P.; Pulugurta, R.; Chatterjee, P. In *NeurIPS 2023 Workshop on Generative AI and Biology (GenBio)*; Workshop poster; non-archival, New Orleans, LA, USA, 2023.
- (293) Torres, M. D. T.; Chen, T.; Wan, F.; Chatterjee, P.; de la Fuente, C. *Cell Bio-materials* **2025**, *1*, Published version of bioRxiv preprint 10.1101/2025.01.31.636003; online early Sept 2, 2025, 100183.
- (294) Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; van den Berg, R. In *Advances in Neural Information Processing Systems*; NeurIPS 2021, 2021; Vol. 34.
- (295) Campbell, A.; Benton, J.; De Bortoli, V.; Rainforth, T.; Deligiannidis, G.; Doucet, A. In *Advances in Neural Information Processing Systems (NeurIPS)*; NeurIPS 2022, Curran Associates, Inc.: New Orleans, LA, USA, 2022; Vol. 35, pp 28266–28279.
- (296) Schiff, Y.; Sahoo, S. S.; Phung, H.; Wang, G.; Boshar, S.; Dalla-torre, H.; Almeida, B. P. d.; Rush, A.; Pierrot, T.; Kuleshov, V. Simple Guidance Mechanisms for Discrete Diffusion Models, arXiv:2412.10193 [cs], 2024.
- (297) Hoogeboom, E.; Gritsenko, A. A.; Bastings, J.; Poole, B.; van den Berg, R.; Salimans, T. In *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*; ICLR 2022 (poster), 2022.
- (298) Chung, H.; Kim, J.; Mccann, M. T.; Klasky, M. L.; Ye, J. C. In *The Eleventh International Conference on Learning Representations*, 2023.
- (299) Zheng, H.; Chu, W.; Zhang, B.; Wu, Z.; Wang, A.; Feng, B. T.; Zou, C.; Sun, Y.; Kovachki, N. B.; Ross, Z. E.; Bouman, K. L.; Yue, Y. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)*; ICLR 2025 Spotlight, OpenReview: Singapore, 2025.
- (300) Soares, D.; Hetzel, L.; Szymczak, P.; Theis, F.; Günnemann, S.; Szczurek, E. Targeted AMP generation through controlled diffusion with efficient embeddings, Preprint, 2025.
- (301) Chu, W.; Wu, Z.; Chen, Y.; Song, Y.; Yue, Y. In *Advances in Neural Information Processing Systems (NeurIPS)*; Poster; to appear, 2025.
- (302) Li, X.; Zhao, Y.; Wang, C.; Scalia, G.; Eraslan, G.; Nair, S.; Biancalani, T.; Ji, S.; Regev, A.; Levine, S.; Uehara, M. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*; Workshop poster; non-archival, New Orleans, LA, USA, 2024.
- (303) Uehara, M.; Su, X.; Zhao, Y.; Li, X.; Regev, A.; Ji, S.; Levine, S.; Biancalani, T. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*; ICML 2025 (poster), PMLR: Vancouver, Canada, 2025.

- (304) Wu, L.; Trippe, B. L.; Naesseth, C. A.; Blei, D. M.; Cunningham, J. P. Practical and Asymptotically Exact Conditional Sampling in Diffusion Models, 2024.
- (305) Lee, C. K.; Jeha, P.; Frellsen, J.; Liò, P.; Albergo, M. S.; Vargas, F. In *ICLR 2025 Workshop on Frontiers in Probabilistic Inference: Learning Meets Sampling*; Oral; non-archival, Singapore, 2025.
- (306) Singhal, R.; Horvitz, Z.; Teehan, R.; Ren, M.; Yu, Z.; McKeown, K.; Ranganath, R. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*; ICML 2025 (poster), Vancouver, Canada, 2025.
- (307) Yang, K.; Klein, D. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; arXiv:2104.05218 [cs], 2021, pp 3511–3535.
- (308) Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; Liu, R. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*; ICLR 2020, 2020.
- (309) Zhao, S.; Brekelmans, R.; Makhzani, A.; Grosse, R. B. In *Proceedings of the 41st International Conference on Machine Learning*, ed. by Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; Berkenkamp, F., PMLR: 2024; Vol. 235, pp 60704–60748.
- (310) Amin, A. N.; Gruver, N.; Kuang, Y.; Li, Y. L.; Elliott, H.; McCarter, C.; Raghu, A.; Greenside, P.; Wilson, A. G. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*; Spotlight, Singapore, 2025.
- (311) Xiong, J.; Nisonoff, H.; Gaur, I.; Lukarska, M. E.; Oltrogge, L. M.; Savage, D. F.; Listgarten, J. Guide your favorite protein sequence generative model, arXiv:2505.04823, 2025.
- (312) Rector-Brooks, J.; Hasan, M.; Peng, Z.; Quinn, Z.; Liu, C.; Mittal, S.; Dziri, N.; Bronstein, M.; Bengio, Y.; Chatterjee, P.; Tong, A.; Bose, A. J. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*; Poster, Singapore, 2025.
- (313) Venkatraman, S.; Jain, M.; Scimeca, L.; Kim, M.; Sendera, M.; Hasan, M.; Rowe, L.; Mittal, S.; Lemos, P.; Bengio, E.; Adam, A.; Rector-Brooks, J.; Bengio, Y.; Berseth, G.; Malkin, N. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*; NeurIPS 2024, Vancouver, Canada, 2024.
- (314) Bartoldson, B. R.; Venkatraman, S.; Diffenderfer, J.; Jain, M.; Ben-Nun, T.; Lee, S.; Kim, M.; Obando-Ceron, J.; Bengio, Y.; Kailkhura, B. In *Advances in Neural Information Processing Systems (NeurIPS)*; Poster; to appear, 2025.

- (315) Venkatraman, S.; Hasan, M.; Kim, M.; Scimeca, L.; Sendera, M.; Bengio, Y.; Berseth, G.; Malkin, N. In *Proceedings of the 42nd International Conference on Machine Learning; ICML 2025* (poster). Also available as arXiv:2502.06999, PMLR: Vancouver, BC, Canada, 2025; Vol. 267, pp 1–28.
- (316) Wang, C.; Uehara, M.; He, Y.; Wang, A.; Biancalani, T.; Lal, A.; Jaakkola, T.; Levine, S.; Wang, H.; Regev, A. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)*; ICLR 2025 Poster, Singapore, 2025.
- (317) Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D.; Finn, C. In *Advances in Neural Information Processing Systems (NeurIPS)*; NeurIPS 2023, 2023; Vol. 36.
- (318) Lim, H.; Lee, G.-H.; No, K. T. *arXiv preprint arXiv:2505.01277* **2025**, Preprint, DOI: [10.48550/arXiv.2505.01277](https://doi.org/10.48550/arXiv.2505.01277).
- (319) Frazier, P. I. In *Recent Advances in Optimization and Modeling of Contemporary Problems*; INFORMS TutORials in Operations Research; INFORMS: Catonsville, MD, 2018, pp 255–278.
- (320) Maus, N.; Jones, H.; Moore, J.; Kusner, M. J.; Bradshaw, J.; Gardner, J. In *Advances in Neural Information Processing Systems*, ed. by Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; Oh, A., Curran Associates, Inc.: 2022; Vol. 35, pp 34505–34518.
- (321) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. *ACS Central Science* **2018**, *4*, 268–276.
- (322) Castro, E.; Godavarthi, A.; Rubinfién, J.; Givechian, K. B.; Bhaskar, D.; Krishnaswamy, S. *Nature Machine Intelligence* **2022**, *4*, 840–851.
- (323) Lee, S.; Park, J.; Chu, J.; Yoon, M.; Kim, H. J. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*; ICLR 2025 (Oral), Singapore, 2025.
- (324) Chen, Y.; Hu, R.; Li, K.; Zhang, Y.; Fu, L.; Zhang, J.; Si, T. *ACS Synthetic Biology* **2023**, *12*, 1461–1473.
- (325) Olson, C. A.; Wu, N. C.; Sun, R. *Current Biology* **2014**, *24*, 2643–2651.
- (326) Kandasamy, K.; Krishnamurthy, A.; Schneider, J.; Póczos, B. In *International conference on artificial intelligence and statistics*, 2018, pp 133–142.
- (327) Russo, D. J.; Van Roy, B.; Kazerouni, A.; Osband, I.; Wen, Z., et al. *Foundations and Trends® in Machine Learning* **2018**, *11*, 1–96.
- (328) Borso, U.; Paglieri, D.; Wells, J.; Rocktäschel, T. In *ICLR 2025 Workshop on Bidirectional Human–AI Alignment (Bi-Align)*; Workshop paper, Singapore, 2025.

- (329) Annadani, Y.; Belakaria, S.; Ermon, S.; Bauer, S.; Engelhardt, B. E. In *Advances in Neural Information Processing Systems (NeurIPS)*; Poster; to appear, 2025.
- (330) Tang, S.; Zhang, Y.; Chatterjee, P. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*; Poster; ICML 2025, Vancouver, Canada, 2025.
- (331) Li, Z.; Yuan, H.; Huang, K.; Ni, C.; Ye, Y.; Chen, M.; Wang, M. *arXiv preprint arXiv:2403.13219* **2024**, Preprint, DOI: [10.48550/arXiv.2403.13219](https://doi.org/10.48550/arXiv.2403.13219).
- (332) Chen, T.; Zhang, Y.; Chatterjee, P. AReURDi: Annealed Rectified Updates for Refining Discrete Flows with Multi-Objective Guidance, 2025.
- (333) Wu, D.; Kuang, N. L.; Niu, R.; Ma, Y.-A.; Yu, R. Diffusion-BBO: Diffusion-Based Inverse Modeling for Online Black-Box Optimization, arXiv preprint; also presented as a poster at NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty, 2025.
- (334) Davis, O.; Kessler, S.; Petrache, M.; Ceylan, I. I.; Bronstein, M.; Bose, A. J. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024; Vol. 37.
- (335) Tang, S.; Zhang, Y.; Tong, A.; Chatterjee, P. In *ICLR 2025 Workshop on Integrating Generative and Experimental Platforms for Biomolecular Design (GEM)*; Workshop poster (non-archival); also available as arXiv:2503.17361, 2025.
- (336) Wang, G.; Schiff, Y.; Sahoo, S. S.; Kuleshov, V. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*; Poster, 2025.
- (337) Peng, F. Z.; Bezemek, Z.; Patel, S.; Rector-Brooks, J.; Yao, S.; Tong, A.; Chatterjee, P. In *ICLR 2025 DeLTa Workshop*; Workshop poster; non-archival, Singapore, 2025.
- (338) Liu, S.; Nam, J.; Campbell, A.; Stärk, H.; Xu, Y.; Jaakkola, T.; Gómez-Bombarelli, R. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*; Poster, Singapore, 2025.
- (339) Amin, A. N.; Gruver, N.; Wilson, A. G. In *Advances in Neural Information Processing Systems (NeurIPS)*; NeurIPS 2025 (poster), to appear, 2025.
- (340) Baker, D.; Church, G. *Science* **2024**, 383, 349–349.
- (341) Wittmann, B.; Alexanian, T.; Bartling, C.; Beal, J.; Clore, A.; Diggans, J.; Flyangolts, K.; Gemler, B. T.; Mitchell, T.; Murphy, S. T.; Wheeler, N. E.; Horvitz, E. *bioRxiv* **2024**, DOI: [10.1101/2024.12.02.626439](https://doi.org/10.1101/2024.12.02.626439).
- (342) Ferruz, N.; Heinzinger, M.; Akdel, M.; Goncarenco, A.; Naef, L.; Dallago, C. *Computational and Structural Biotechnology Journal* **2023**, 21, 238–250.

- (343) Praljak, N.; Yeh, H.; Moore, M.; Socolich, M.; Ranganathan, R.; Ferguson, A. L. *bioRxiv* **2024**, DOI: [10.1101/2024.11.11.622734](https://doi.org/10.1101/2024.11.11.622734).
- (344) Ruffolo, J. A.; Bhatnagar, A.; Beazer, J.; Nayfach, S.; Russ, J.; Hill, E.; Hussain, R.; Madani, A. *bioRxiv* **2024**, DOI: [10.1101/2024.08.03.606485](https://doi.org/10.1101/2024.08.03.606485).
- (345) Guloglu, B.; Bragança, M.; Graves, A.; Cameron, S.; Atkinson, T.; Copoiu, L.; Laterre, A.; Barrett, T. D. *bioRxiv* **2025**, 2025.04.29.651170.
- (346) Frey, N. C. et al. *bioRxiv* **2025**, 2025.02.19.639050.
- (347) Gruver, N.; Stanton, S.; Frey, N. C.; Rudner, T. G. J.; Hotzel, I.; Lafrance-Vanasse, J.; Rajpal, A.; Cho, K.; Wilson, A. G. In *Advances in Neural Information Processing Systems* 36, 2023.
- (348) Yang, J.; Chu, W.; Khalil, D.; Astudillo, R.; Wittmann, B. J.; Arnold, F. H.; Yue, Y. In *Advances in Neural Information Processing Systems*, 2025.
- (349) Widatalla, T.; Rafailov, R.; Hie, B. *bioRxiv* **2024**, DOI: [10.1101/2024.05.20.595026](https://doi.org/10.1101/2024.05.20.595026).
- (350) Almhjell, P. J.; Johnston, K. E.; Porter, N. J.; Kennemur, J. L.; Bhethanabotla, V. C.; Ducharme, J.; Arnold, F. H. *Nature Chemical Biology* **2024**, 20, 1086–1093.
- (351) Boiko, D. A.; MacKnight, R.; Kline, B.; Gomes, G. *Nature* **2023**, 624, 570–578.
- (352) Gao, S.; Fang, A.; Huang, Y.; Giunchiglia, V.; Noori, A.; Schwarz, J. R.; Ektefaie, Y.; Kondic, J.; Zitnik, M. *Cell* **2024**, 187, 6125–6151.
- (353) Gottweis, J. et al. *arXiv* **2025**, DOI: [10.48550/arXiv.2502.18864](https://doi.org/10.48550/arXiv.2502.18864).
- (354) Huang, K. et al. *bioRxiv* **2025**, DOI: [10.1101/2025.05.30.656746](https://doi.org/10.1101/2025.05.30.656746).
- (355) Roohani, Y.; Lee, A.; Huang, Q.; Vora, J.; Steinhart, Z.; Huang, K.; Marson, A.; Liang, P.; Leskovec, J. In *13th International Conference on Learning Representations*, 2025.
- (356) Simon, E.; Swanson, K.; Zou, J. *Nature Methods* **2024**, 21, 1422–1429.
- (357) Singh, N.; Lane, S.; Yu, T.; Lu, J.; Ramos, A.; Cui, H.; Zhao, H. *Nature Communications* **2025**, 16, 5648.
- (358) Swanson, K.; Wu, W.; Bulaong, N. L.; Pak, J. E.; Zou, J. *bioRxiv* **2024**, DOI: [10.1101/2024.11.11.623004](https://doi.org/10.1101/2024.11.11.623004).
- (359) Long, Y.; Abbasinejad, F.; Li, F.-Z.; Reinprecht, P.; Wittmann, B.; Kennemur, J. L.; Carder, H.; Yang, J.; Lambert, T.; O'Meara, R.; Radtke, L.; Qin, Z.; Brinkmann-Chen, S.; Arnold, F.; Mora, A. *Nucleic Acids Research* **2025**, gkaf1142.

- (360) Armer, C.; Kane, H.; Cortade, D. L.; Redestig, H.; Yusuf, A.; Rollins, N.; Spinner, H.; Marks, D.; Brunette, .; Kelly, P. J.; DeBenedictis, E. *bioRxiv* **2024**, DOI: [10.1101/2024.08.12.606135](https://doi.org/10.1101/2024.08.12.606135).
- (361) Gao, W.; Coley, C. W. *Journal of Chemical Information and Modeling* **2020**, *60*, 5714–5723.
- (362) Freschlin, C.; Yang, K.; Romero, P. A. *bioRxiv* **2025**, DOI: [10.1101/2025.03.22.644747](https://doi.org/10.1101/2025.03.22.644747).
- (363) Zhu, D.; Brookes, D. H.; Busia, A.; Carneiro, A.; Fannjiang, C.; Popova, G.; Shin, D.; Donohue, K. C.; Lin, L. F.; Miller, Z. M.; Williams, E. R.; Chang, E. F.; Nowakowski, T. J.; Listgarten, J.; Schaffer, D. V. *Science Advances* **2024**, *10*, eadj3786.
- (364) Markin, C. J.; Mokhtari, D. A.; Sunden, F.; Appel, M. J.; Akiva, E.; Longwell, S. A.; Sabatti, C.; Herschlag, D.; Fordyce, P. M. *Science* **2021**, *373*, eabf8761.
- (365) Ikononova, S. P.; Wittmann, B. J.; Piorino, F.; Ross, D. J.; Schaffter, S. W.; Vasilyeva, O.; Horvitz, E.; Diggans, J.; Strychalski, E. A.; Lin-Gibson, S.; Taghon, G. J. *bioRxiv* **2025**, DOI: [10.1101/2025.05.15.654077](https://doi.org/10.1101/2025.05.15.654077).
- (366) Nov, Y. *Applied and Environmental Microbiology* **2012**, *78*, 258–262.
- (367) Gibson, D. G.; Young, L.; Chuang, R.-Y.; Venter, J. C.; Hutchison, C. A.; Smith, H. O. *Nature Methods* **2009**, *6*, 343–345.
- (368) Balandat, M.; Karrer, B.; Jiang, D. R.; Daulton, S.; Letham, B.; Wilson, A. G.; Bakshy, E. **2020**.
- (369) Gardner, J. R.; Pleiss, G.; Bindel, D.; Weinberger, K. Q.; Wilson, A. G. **2021**.
- (370) Rasmussen, C. E.; Williams, C. K. I., *Gaussian Processes for Machine Learning*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, Mass, 2006.
- (371) Chen, T.; Guestrin, C. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp 785–794.
- (372) Desautels, T.; Krause, A.; Burdick, J. W. **2014**, *15*, 4053–4103.
- (373) Rahimi, A.; Recht, B. **2007**.
- (374) Liu, S.; Wang, H.; Liu, W.; Lasenby, J.; Guo, H.; Tang, J. In *International Conference on Learning Representations*, 2022.
- (375) Johnson, L. S.; Eddy, S. R.; Portugaly, E. *BMC Bioinformatics* **2010**, *11*, 431.
- (376) Katoh, K.; Standley, D. M. *Molecular Biology and Evolution* **2013**, *30*, 772–780.

- (377) Nichol, A. Q.; Dhariwal, P. In *Proceedings of the 38th International Conference on Machine Learning*, ed. by Meila, M.; Zhang, T., PMLR: 2021; Vol. 139, pp 8162–8171.
- (378) Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; Poole, B. In *Proceedings of the International Conference on Learning Representations (ICLR)*; ICLR 2021 — Outstanding Paper Award, 2021.
- (379) Mardani, M.; Song, J.; Kautz, J.; Vahdat, A. In *The Twelfth International Conference on Learning Representations*, 2024.

METHOD DETAILS

A.1 Methods for Chapter 2

Cloning of Random ParPgb Variants

Cloning for Single Site-Saturation Mutagenesis. Chemically competent *Escherichia coli* (*E. coli*) cells (T7 Express Competent *E. coli*) were purchased from New England Biolabs (NEB, Ipswich, MA). Phusion polymerase and *DpnI* were purchased from NEB. SSM experiments were performed using primers bearing degenerate codons (NNK) using a modified QuikChange™ protocol [366]. The PCR conditions were (final concentrations): Phusion HF Buffer 1x, 0.2 mM dNTPs each, 0.5 μ M of forward primers, 0.5 μ M reverse primer, and 0.02 U/ μ L of Phusion polymerase. The standard Phusion PCR protocol was used. Upon completion of PCRs, the remaining template was digested with *DpnI*. Gel purification was performed with a Zymoclean Gel DNA Recovery Kit (Zymo Research Corp, Irvine, CA). The purified PCR product was then assembled using the Gibson assembly protocol [367].

Transformation of Single Site Mutants. 96-well deep-well plates are shaken in an INFORS HT Multitron Shaker in all instances. The assembly products obtained were used to transform T7 Express Competent *E. coli* (High Efficiency) cells (NEB, Ipswich, MA) following the recommended protocol. Upon heat-shock transformation, mixtures were recovered in 0.4 mL Luria-Bertani medium (LB) (Research Products Int.), after which the cells were incubated at 37 °C with shaking at 220 rpm for 30 minutes before being plated on LB-agar plates with 100 μ g/mL ampicillin (LB-amp agar plates). Single colonies from LB-agar plates were picked using sterilized toothpicks, which were used to individually inoculate 400 μ L of LB containing 100 μ g/mL of ampicillin (LB-amp) in 2 mL 96-well deep-well plates. The plates were incubated at 37 °C and shaken at 220 rpm for 16-18 hours. The following morning 50 μ L of preculture from each well were added to the wells of a 96-well flat-bottom tissue culture plate (ThermoFisher) preloaded with 50 μ L of 50% glycerol solution. These glycerol stocks were stored at -80°C for future inoculation. Additionally, the sequences of protoglobin genes contained in every well were sequenced using the evSeq protocol [166].

Cloning for Multisite-Saturation Mutagenesis. Mutations were simultaneously incorporated as with single SSM using the ParLQ_quadNNK primers. The library transformation was recovered in 0.4 mL LB. 50 μ L of transformation mixture were used to inoculate 6 mL of LB-Amp in a 15 mL plastic culture tube. This culture was allowed to shake overnight at 37°C. The following morning, this library preculture was miniprep using a QIAprep Spin Miniprep Kit (Qiagen, Hilden, Germany). This miniprep sample was used as the new template for mutagenesis with the primers for SSM of site 89. The Gibson products for the new five-site library were transformed using the recommended protocol into T7 Express Competent *E. coli*. Upon heat-shock transformation, mixtures were recovered in 0.4 mL Luria-Bertani medium (LB) (Research Products Int.), after which the cells were incubated at 37 °C with shaking at 220 rpm for 30 minutes before being plated on LB-agar plates with 100 μ g/mL ampicillin (LB-amp agar plates). Single colonies from LB-agar plates were picked using sterilized toothpicks, which were used to individually inoculate 400 μ L of LB containing 100 μ g/mL of ampicillin (LB-amp) in 2 mL 96-well deep-well plates across 4 separate plates. The plates were incubated at 37 °C and shaken at 220 rpm for 16-18 hours. The following morning 50 μ L of preculture from each well were added to the wells of a 96-well flat-bottom tissue culture plate (ThermoFisher) preloaded with 50 μ L of 50% glycerol solution. These glycerol stocks were stored at -80°C for future inoculation. Additionally, the sequences of protoglobin genes contained in every well were sequenced using LevSeq sequencing [151].

Cloning of ParPgb Predicted Sequences

96-Well Plate Gibson Protocol. Exact genes encoding ParLQ mutants predicted by Active Learning-Assisted Directed Evolution (ALDE) were synthesized and delivered by Elegen Corp. (San Carlos, CA). DNA fragments were received as dry residues in 96-well PCR plates in 2-4 μ g quantities. These DNA samples were dissolved in 100 μ L of double-distilled H₂O (ddH₂O), yielding concentrations between 20-40 ng/ μ L. 0.7 μ L of these gene solutions were added to the wells of a 96-well PCR plate (Globe Scientific Inc., Mahwah, NJ). 1.0 μ L of an aqueous solution containing 60 ng/ μ L of linearized pET-22b(+) backbone with overhangs designed for Gibson ligation with the ordered DNA sequences was added to each of the wells of this plate. Finally, to each well was added 5 μ L of Gibson assembly mix. The 96-well plate was then incubated at 50°C for 60 minutes, after which the Gibson products were placed on ice. These Gibson products could then either be

directly used for transformation or stored at -20°C for later use.

96-Well Plate Transformation Protocol. To each well of the previously described Gibson assembly plate was added 5 μ L of T7 Express Competent *E. coli*. The cell solutions were allowed to incubate on ice for 20 minutes, after which they were heat-shocked at 42°C for 10 seconds in a water bath. The cells were then recovered with the addition of 100 μ L of LB. Without outgrowth at 37°C, 10 μ L of each transformation mixture was used to inoculate the wells of a 2 mL 96-well deep-well plate in which the wells had been preloaded with 400 μ L LB-Amp. This plate was incubated at 37 °C and shaken at 220 rpm for 16-18 hours. The following morning the plate was removed from shaking and allowed to sit at room temperature for 8-10 hours. After this rest phase, 1 μ L from each well was used to reinoculate yet another 96-well deep-well plate preloaded with 400 μ L LB-Amp. This cell passage plate was incubated at 37 °C and shaken at 220 rpm for 16-18 hours. The following morning 50 μ L of preculture from each well was added to the wells of a 96-well flat-bottom tissue culture plate (ThermoFisher) preloaded with 50 μ L of 50% glycerol solution. These glycerol stocks were stored at -80°C for future inoculation. The sequences of transformants generated in this manner were confirmed by LevSeq long-read sequencing.

Protocols for the Screening of ParPgb Variants

96-Well Plate Library Expression. The wells of 2 mL, 96-well deep-well plates were filled with 400 μ L LB-Amp. Previously generated 96-well plate glycerol stocks were removed from -80°C storage and placed on dry ice. Multichannel pipet tips were used to scratch the frozen glycerol stock surface and used to inoculate the aforementioned deep-well plate. These pre-expression cultures were incubated at 37 °C and shaken at 220 rpm for 16-18 hours. For expression cultures, the following morning 50 μ L of these precultures were used to inoculate 900 μ L of Terrific Broth (TB) (Research Products Int.) with 100 μ g/mL of ampicillin (TB-amp) per well in 96-well deep-well plates. These expression cultures were initially incubated at 37 °C and 220 rpm for 2.5 hours, at which point they were allowed to sit at room temperature for 30 minutes. Expression of proteins was induced with isopropyl- β -D-thiogalactoside (IPTG) and cellular heme production was increased with 5-aminolevulinic acid (ALA). An induction mixture containing IPTG and ALA in TB-amp (50 μ L) was added to each well such that the final concentrations of IPTG and ALA were 0.5 mM and 1.0 mM, respectively. The total culture volumes were

1 mL. The plates were then incubated at 22 °C and 220 rpm overnight.

96-Well Plate Library Reactions and Screening. Expression cultures containing *E. coli* expressing hemoproteins of interest were centrifuged at $4000 \times g$ for 10 minutes at 4 °C. The supernatant was discarded, and nitrogen-free M9 minimal medium (M9-N, 380 μ L) was added to each well. The pellets were resuspended in this medium via shaking at room temperature for 30 minutes. The plates were then transferred into a vinyl Coy anaerobic chamber (0 – 30 ppm O₂). To each well was added 20 μ L of a MeCN solution with 200 mM of the desired styrene substrate and 300 mM of ethyl diazoacetate (EDA). The final reaction volume was 400 μ L, and the final concentrations of the styrene and EDA were 10 mM and 15 mM, respectively. The plates were then sealed carefully with a foil cover and shaken at room temperature for 16 hours in the Coy chamber. Once complete, plates were worked up for processing by adding 600 μ L of a 1:1 solution of ethyl acetate:cyclohexane containing 1,3,5-trimethoxybenzene as an internal standard (1.0 mM concentration). A silicone sealing mat (AWSM1003S, ArcticWhite) was used to cover the plate and the two layers were thoroughly mixed by rapid inversion of the plate. The plate was then centrifuged ($5000 \times g$ for 5 minutes at room temperature) to separate the phases. Afterwards, a 200 μ L aliquot of the organic layer was transferred to a GC vial insert in a GC vial, and the samples were analyzed by GC-FID.

Machine Learning Details

The initial training data for the ParPgb campaign was obtained by merging sequencing data with screening yield data. Measured yields were averaged for sequences with the same amino acid combination and normalized to the yield of the *cis* product formation of the parent variants (WYLQF) on each plate. These normalized values were used for model training and acquiring new points, which followed the same protocol as the computational simulations on GB1 and TrpB. For the wet-lab campaign, we trained the model with onehot encodings, the DNN ensemble with 5 models and bootstrapping using 90% of the available training data for each model, and Thompson sampling as the acquisition function. These design choices correspond to the most consistent strategy based on the computational simulations. Detailed instructions on how to reproduce our results and run ALDE for other engineering campaigns are provided at <https://github.com/jsunn-y/ALDE>.

Most Bayesian optimization algorithms consist of two main components: (1) a prob-

abilistic surrogate model of the objective function and (2) an acquisition function. The surrogate model predicts the objective function values at unobserved inputs, while the acquisition function quantifies the potential benefit of evaluating any given batch of inputs based on these predictions. In each iteration of the Bayesian optimization loop, a new batch of inputs is selected by maximizing the acquisition function. After evaluating the objective function at these new inputs, the surrogate model is updated, and the process repeats. Below, we describe in detail the probabilistic models and acquisition functions explored in this work, which were implemented using BoTorch [368] and GPyTorch [369].

Probabilistic Models for Bayesian Optimization Let $\mathbf{X}$ denote the input space (i.e., the space of feasible protein sequences) and let $f : \mathbf{X} \rightarrow \mathbf{R}$ denote the objective function (i.e., the metric we wish to optimize). In this work, we explore four classes of probabilistic surrogate models of the objective function: regular Gaussian processes (GP), deep kernel Gaussian processes (DKL), deep ensembles (DNN ensemble), and boosting ensembles.

Gaussian Processes. A Gaussian process model is defined in terms of a prior mean function $\mu_0 : \mathbf{X} \rightarrow \mathbf{R}$ and a prior covariance function $K_0 : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{R}$ and it encodes a Bayesian prior distribution over f . Given a dataset of n evaluations of the objective function, denoted as $\mathcal{D}_\backslash = \{ (x_i, y_i) \}_{i=1}^n$, one can derive the posterior distribution of f given $\mathcal{D}_\backslash$. If these evaluations are corrupted by i.i.d. additive Gaussian noise, i.e., $y_i = f(x_i) + \epsilon_i$, where $\epsilon_1, \dots, \epsilon_n$ are i.i.d. Gaussian with mean zero and variance σ^2 , the posterior is again a Gaussian process characterized by a posterior mean function $\mu_n : \mathbf{X} \rightarrow \mathbf{R}$ and a posterior covariance function $K_n : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{R}$. These functions can be computed in closed form in terms of the prior mean and covariance functions as well as the data using the classical Gaussian process regression formulas [370]. The noise variance σ^2 and other hyperparameters of the model (such as the length-scale parameters) can be estimated by maximizing the log marginal likelihood.

Deep Kernel Learning. Gaussian process models with classical covariance functions, such as the Matern or squared exponential covariance functions, are known to perform poorly in high-dimensional input spaces [142]. To address this limitation, Wilson et al. (2015) proposed *deep kernel learning* [141]. Succinctly, this approach uses a covariance function of the form $K(x, x') = k(\phi_w(x), \phi_w(x'))$, where k is

a regular covariance function (e.g., squared exponential) and ϕ_w is a deep neural network with weights w . These weights are treated like hyperparameters of the model, which can also be estimated by maximizing the log marginal likelihood.

Boosting Ensembles. Boosting models leverage a sequential training strategy where each new model is trained to correct the errors of the previously combined models [371]. The final prediction is often a weighted sum of the predictions made by earlier models, where the weights reflect each model's accuracy. Unlike methods such as bagging, which train models independently and in parallel, boosting specifically designs each new model to address the weaknesses of the existing ensemble, thereby creating a strong predictive model from a sequence of weaker ones. While boosting does not inherently offer a probabilistic interpretation like Bayesian methods, it is highly effective for reducing bias and variance in predictive modeling tasks. Here, we train the boosting ensembles with bootstrapping; each ensemble consists of 5 models where 90% of the total training data is randomly seen during training.

Deep Ensembles. Deep neural network (DNN) ensemble models are constructed by training identical deep neural network architectures multiple times, each with different random initializations of the weight parameters. Here, we train the deep ensembles with bootstrapping; each ensemble consists of 5 models where 90% of the total training data is randomly seen during training. These independently trained networks are then collectively used as if they were samples from a Bayesian posterior distribution over the objective function f . Unlike Gaussian processes, deep ensembles lack a proper Bayesian interpretation. However, Izmailov and Wilson argue it is possible to see these models as a form of approximate Bayesian inference [165]. We adopt this view in our work.

Acquisition Functions for Bayesian Optimization

Expected Improvement. The expected improvement (EI) acquisition function is given by $\alpha_n(x) = E_n [\{f(x) - f_n^*\}^+]$, where $f_n^* = \max_{i=1,\dots,n} f(x_i)$ and the expectation is computed with respect to the posterior distribution given $\mathcal{D}_\setminus$ [158]. For Gaussian posterior distributions and noise-free observations (where f_n^* is a constant rather than a random variable), the EI can be expressed in a closed form using the posterior mean and variance. In scenarios where these conditions do not hold, computing the EI often requires approximate calculation, typically through

Monte Carlo sampling techniques. When extending the EI to the batch setting, the acquisition function becomes $\alpha_n(X) = E_n [\{\max_{x \in X} f(x) - f_n^*\}^+]$, where $X = (x_1, \dots, x_q) \in \mathbf{X}^q$ is a batch of q inputs (qEI). Maximizing the batched EI poses significant computational challenges due to the requirement to optimize over $\mathbf{X}^q$. However, by exploiting the submodularity of the acquisition function, an efficient approximation can be achieved through a greedy optimization strategy, selecting each input in the batch sequentially. In this study, we tested qEI, but it ran slowly without noticeable improvement, so it was not included in the final results.

Upper Confidence Bound. The upper confidence bound (UCB) acquisition function is defined by $\alpha_n(x) = \mu_n(x) + \beta_n^{1/2} \sigma_n(x)$, where $\mu_n(x)$ and $\sigma_n(x)$ are the posterior mean and standard deviation, respectively, and β_n is a parameter that controls the exploration-exploitation trade-off. In our experiments, we set $\beta_n = 4$. While there are sophisticated batch extensions of the UCB acquisition function available in the literature [372], our approach utilizes a straightforward heuristic. Specifically, we form batches by selecting the q inputs that yield the highest values of $\alpha_n(x)$, evaluated across all discrete x in the design space. The Greedy acquisition function can be thought of as a specific case of UCB with $\beta_n = 0$ so the acquisition function becomes $\alpha_n(x) = \mu_n(x)$. For the frequentist ensemble models, we evaluate $\mu_n(x)$ and $\sigma_n(x)$ as the mean and standard deviation of all models in the ensemble, respectively.

Thompson Sampling. Thompson Sampling (TS) is a randomized selection strategy where the next input to evaluate is obtained by drawing a sample (function) from the posterior distribution of f and selecting the point that maximizes this sample. For the GP and DKL models, we approximate samples from the posterior using 1000 random Fourier features [373]. For the frequentist ensemble models, the random function sample is drawn as one of the models in the ensemble. In the batch setting, each input in the batch is obtained as an independent sample. Unlike the EI and UCB, TS is inherently stochastic as opposed to deterministic; however, we note that since our ensembles have five models each, TS is less stochastic in this setting.

A.2 Methods for Chapter 3

Dataset Processing

Our data processing and splitting workflow is shown in Fig. A.1. Protein sequence data, paired to EC number(s), were downloaded from UniProt, selecting only re-

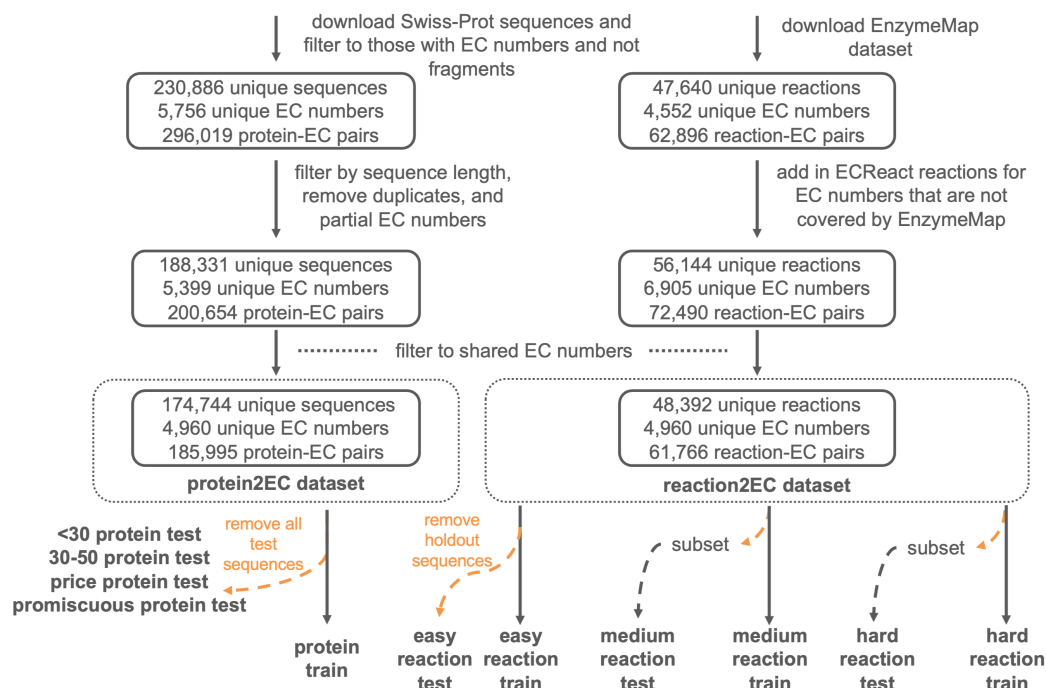


Figure A.1: Workflow used to process the datasets containing EC numbers, protein sequences, and reactions used in this study.

viewed sequences in Swiss-Prot, resulting in a total of 571,282 sequences on the 13th of May 2024. These were filtered to only retain sequences with an EC number that were not annotated as fragments. After, sequences were filtered to length between 100 and 1024, partial EC numbers (containing a dash) were removed, duplicate entries (based on EC and sequence) were dropped, and only EC numbers with associated reactions were kept, leaving 200,654 sequence-EC pairs. We call this the protein2EC dataset.

Reaction data were downloaded from ECRreact and EnzymeMap. ECRreact contains 62,222 entries of reaction in SMILES as SMARTS strings format coupled with EC number from a range of data sources. EnzymeMap had 349,458 entries containing sequence-reaction-EC triplets with the reaction in several formats, SMILES as SMARTS string, an atom-level mapping of bond formation and breaking (mapped reaction), and reactions written as text based on IUPAC compound names. Starting with EnzymeMap, protein sequence information was dropped (leaving reaction-EC pairs), and duplicate reaction-EC pairs were removed leaving 62,896 reaction-EC pairs. EC numbers that existed in ECRreact but not in Enzyme map were added to EnzymeMap providing our reaction dataset of 72,490 reaction-EC pairs. We did

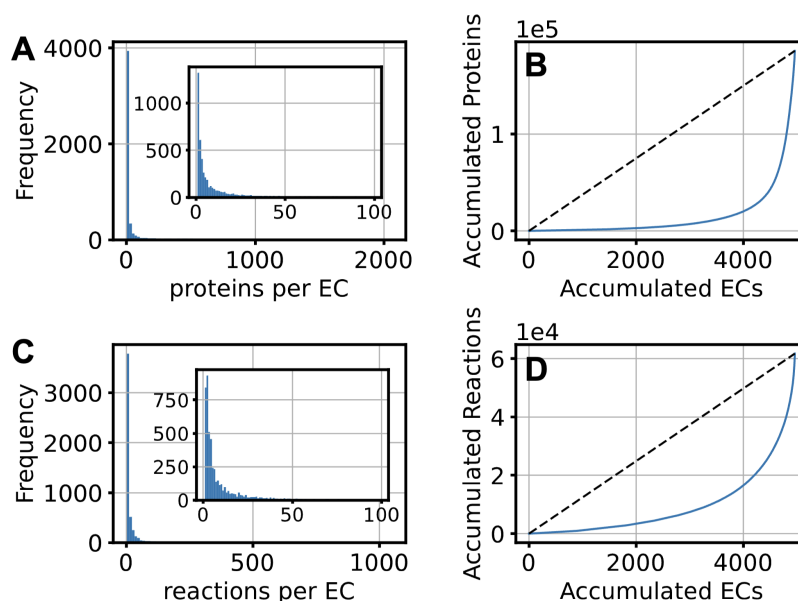


Figure A.2: Distribution of the protein2EC dataset, shown as (A) a histogram of proteins per EC and as (B) a cumulative distribution plot, analogous to a Lorenz curve. Distribution of the reaction2EC dataset, shown as (C) a histogram of reactions per EC and as (D) a cumulative distribution plot, analogous to a Lorenz curve. Both datasets are heavily skewed with most examples belonging to a few EC numbers, and many EC numbers only having one example. If samples were evenly distributed across ECs, they would follow the dashed line in (B) and (D).

this because EnzymeMap is a higher quality reaction dataset, with fewer incorrect entries. Note the ECReact only contains reactions formatted as SMILES/SMARTS, not as mapped reactions. Together, we call this the reaction2EC dataset.

Finally, we filtered the protein2EC and reaction2EC datasets, such that only EC numbers present in both datasets are retained, leaving 61,766 reaction-EC pairs and 185,995 sequence-EC pairs, with 4,960 unique EC numbers. Clustering using MMseqs2 [245] was performed at 30%, 50%, 70%, and 90% sequence identity and included in the protein2EC dataset for downstream use. The distributions of the curated datasets are shown in Fig. A.2.

Dataset Splitting Details

First, we sought to create test sets capturing different types of out-of-domain generalization for Task 1, EC classification (visualized in Fig. A.3A).

Using the results from clustering, we generated two test sets, with approximately <30% and 30-50% sequence identity, respectively, to sequences in the training set.

We also only considered ECs at level 4 which had isolated clusters, in other words, an MMseqs2 cluster with only a single member. For every EC level 3 (X.X.X.-), we sampled up to three random ECs at level 4 (X.X.X.X), and then randomly selected a single isolated sequence from each class to add to the test set. This provided balanced test datasets across the functions at EC level 3. For this set, we also enforced that the sequences did not map to multiple EC numbers. Test sets were balanced across functions (e.g. EC number), rather than across their prevalence as protein sequences in nature, based on the idea that most synthesis planners and enzyme engineers would find more value in predictive ability across a broad range of functions.

Next, we looked for "promiscuous" enzymes, which in this work, we define as those annotated with multiple EC numbers. For our promiscuous test set, we only considered using promiscuous enzymes that had a combination of ECs that occurred at least twice. From this set we only took a single sample from each EC combination, and finally randomly selected a single entry from those which have ECs with high surprise levels (i.e. X.X.-.-, where variation occurs at the third EC level or higher). From the test sets, we did not include sequences that were labeled as belonging to part of a heterogeneous multi-complex protein.

Finally, we compiled a commonly misclassified test set from Price et al., filtered to sequences of length 100-1024, and dropped test sequences that are present in our protein2EC dataset. The pooled set of all proteins in the test sets were removed from our training data leaving 184,529 sequence-EC pairs for training. As a result of the holdout process, a few of the 4,960 EC numbers are missing from the train set, which should minimally affect performance on the test sets.

For Task 2, we sought to test reaction extrapolation, namely, retrieval of enzymes based on unseen reactions (visualized in [Fig. A.3B](#)). For the easy and medium test sets, we randomly sampled an EC level 4 EC (X.X.X.X) from every level three EC (X.X.X.-). From each of the held-out level 4 ECs, three random reactions were sampled to be in the test set. For the test set, we only considered (1) reactions that do not map to multiple EC numbers, (2) EC numbers with at least four reactions, and (3) reactions from EnzymeMap, which are higher fidelity and consequently have fewer misannotations.

For the easy set, only the test set reactions were held out. For the medium set, all reactions under the corresponding level 4 ECs were held out. Note that the easy and medium test sets are exactly the same, but the easy set has fewer held out reactions

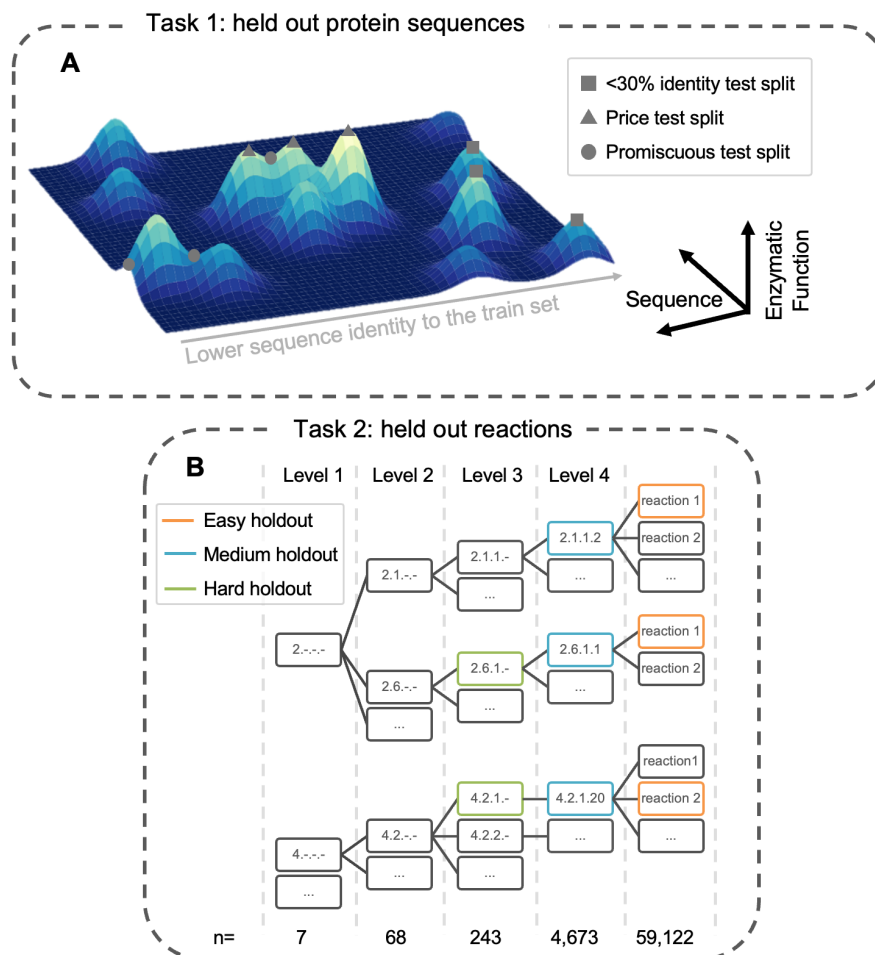


Figure A.3: (A) Visualization of the types of extrapolation found in the test splits for Task 1. (B) Visualization of how different subsets of reaction space are held out in the splits for Task 2.

compared to the medium set, thus their training sets are different.

The hard set was evenly balanced across a random sample from EC level 2 (X.X.-.-) to ensure there was an even distribution across level 3 (EC3, X.X.X.-), from which 53 random EC3s are sampled, with all reactions under the 53 EC3s held out. 460 reactions associated with the 53 ECs at level 3 were used as the test set, which are evenly spread across the ECs at level 3, with up to three reactions from each EC number at level 4. Note we ensure the hard test reactions are shared with the easy and medium ones, when possible, to correlate the performance of our test splits. It should be noted that the hard train-test split is different from the easy and medium splits. For Task 2, all of the protein-EC pairs can be used during training.

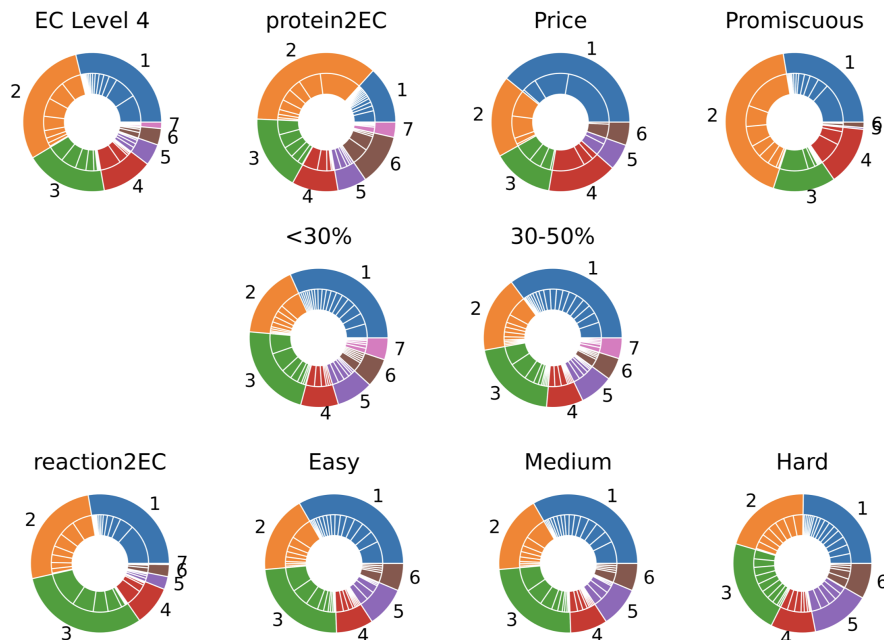


Figure A.4: Distribution of the EC numbers found in the test splits, and compared to a few reference datasets. Outer ring shows the level 1 category (X.-.-), and the inner ring shows the level 2 category (X.X.-). EC Level 4 refers the distribution of unique ECs at level 4 (X.X.X.X). protein2EC shows the distribution of all protein sequences in protein2EC across ECs, and reaction2EC shows the distribution of all reactions in reaction2EC across ECs.

CREEP Model Details

All steps needed to reproduce CREEP training and downstream retrieval can be found at <https://github.com/jsunn-y/CARE/>.

During the pretraining stage, we follow the contrastive learning paradigm that maximizes mutual information between two views [374]. For each enzyme family (EC number) x , we extract three views: protein sequence x_p (protein), reactions smiles x_r (reaction), and textual description of EC number from gene ontology x_t (text) [240]. In CREEP, the goal is to maximize the mutual information between all three modalities. For illustration purposes, we use the example of protein-reaction contrastive learning, but the derivations for reaction-text and text-protein contrastive learning follow a similar approach.

We follow the method proposed in GraphMVP by adopting EBM-NCE to estimate mutual information between our modalities [374]. In our case, EBM-NCE aligns the protein sequence and reaction SMILES pairs for the same enzyme family and contrasts the pairs for different enzyme families simultaneously. The objective

function is

$$\begin{aligned} \mathcal{L}_{\text{EBM-NCE}} = & -\frac{1}{2} \left(\mathbb{E}_{\mathbf{x}_p, \mathbf{x}_r} [\log \sigma(E(\mathbf{x}_p, \mathbf{x}_r))] + \mathbb{E}_{\mathbf{x}_p, \mathbf{x}'_r} [\log(1 - \sigma(E(\mathbf{x}_p, \mathbf{x}'_r)))] \right) \\ & + \mathbb{E}_{\mathbf{x}_p, \mathbf{x}_r} [\log \sigma(E(\mathbf{x}_p, \mathbf{x}_r))] + \mathbb{E}_{\mathbf{x}'_p, \mathbf{x}_r} [\log(1 - \sigma(E(\mathbf{x}'_p, \mathbf{x}_r)))] \end{aligned} \quad (\text{A.1})$$

where $\mathbf{x}_p$ and $\mathbf{x}_r$ form the (protein sequence, reaction SMILES) pair for each reaction, and $\mathbf{x}_{p'}$ and $\mathbf{x}_{r'}$ are the negative samples which are produced by randomly sampling from a Gaussian distribution, which we use as an approximation for the empirical data distribution. $E(\cdot)$ is the energy function with a flexible formulation, and we use the dot product as a metric for similarity within the latent space.

For our default CREEP model, we project representations to 256 dimensions in the shared latent space for all modalities. We train for 40 epochs, and in each epoch, we loop over each EC number and mine each batch such that all protein-reaction-text triplets come from different EC numbers. Note that we also use 50% sequence clustering to help select protein sequences with increased diversity.

Downstream retrieval using models such as CREEP and CLIPZyme is shown in Fig. A.5. The standard workflow starts with a query reaction and ranks ECs by sorting based on distance from the reaction representation to the EC cluster centroids of protein sequence representations belonging to each EC number (Fig. A.5A). For CREEP (w/ Text), a similar ranking can be determined by using text as a query (Fig. A.5B), and the overall ranking can be calculated as an average of the two rankings. To reduce inference time, we used protein sequences clustered at 50% identity to calculate the protein representation centroids clustered by EC. Finally, the similarity baseline can be understood as sorting based on distance from a query reaction to reaction cluster centroids of reactions in the train set, clustered by EC (Fig. A.5C).

Implementation Details for Benchmarking Methods

All implementation details for methods used in the study can be found at <https://github.com/jsunn-y/CARE/>. We focus on evaluating the accuracy of retrieval for $k=1$. A level 1 accuracy means the top level EC class has to be correct, for level 2 accuracy, both level 1 and level 2 have to be correct, and so on. Accuracy is calculated as the number of true answers over the total number of examples. An invalid EC number from language models (ChatGPT and Pika) is considered incorrect on every level. For the non-promiscuous splits, the accuracy for each entry in the test set is 1 or 0. For the promiscuous split in Task 1, for each entry, we enumerate over each true EC to find the maximum accuracy within the pool of

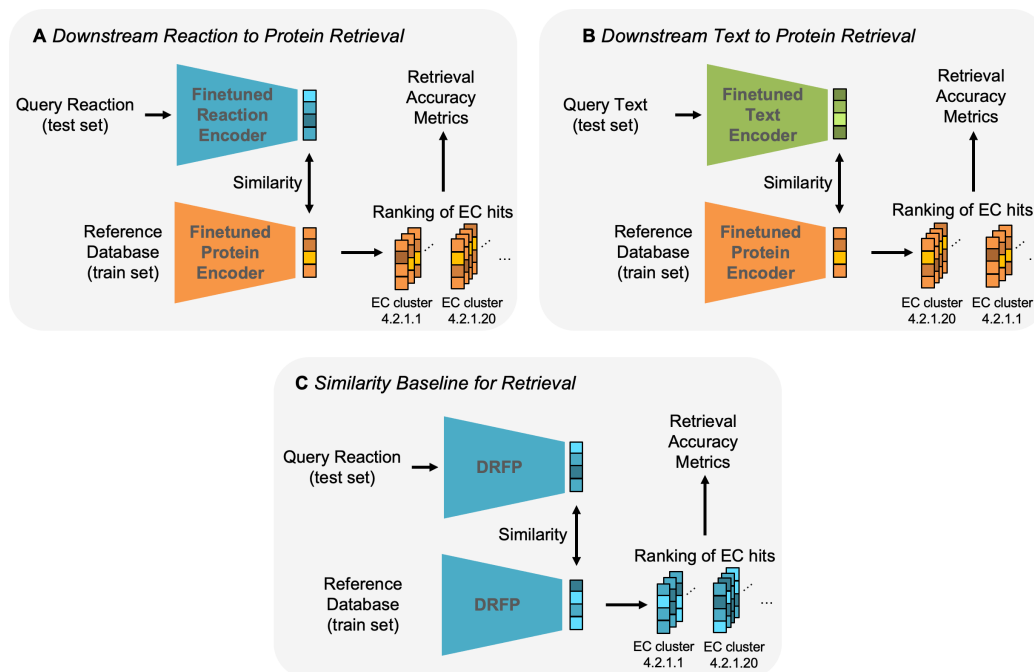


Figure A.5: Downstream retrieval using contrastively pretrained models. CREEP and CLIPZyme both utilize (A) reaction to protein retrieval. CREEP (w/ Text) can incorporate (B) text to protein retrieval, and consider the rankings from both (A) and (B). (C) The similarity baseline involves reaction to reaction retrieval using a fixed representation from DRFP [232].

predicted ECs, and the overall accuracy for the entry is the average of the accuracies for each true EC.

ChatGPT. ChatGPT was used for both Task 1 and Task 2, using the API and gpt-4o-mini model. We performed minimal prompt engineering, with our initial prompt being the most standard initial question: "give me the top N EC numbers associated with this amino acid sequence:" for Task 1. The query was modified to: "You are protein engineer capable of predicting EC numbers from the sequence alone. You are also a skilled programmer and able to execute the code necessary to predict an EC number when you can't use reason alone. Given a protein sequence you are able to determine the most likely enzyme class for a sequence because you are that skilled. You don't give up when faced with a sequence you don't know, you will use tools to resolve the most likely enzyme sequence. You only return enzyme commission numbers in a comma separated list, no other text is returned, you have failed if you do not return the EC numbers. You only return the most likely EC number." This resulted in a k=1 response, we found this more consistent then asking

for a list of k results. Note that we do not do any filtering or data cleaning on the output of ChatGPT. Doing this could improve the accuracy, for example, in one case we identified in the hard set the output was EC4.6.1.18 which we scored as 0 despite the true EC being 4.6.1.2 for the reaction **UTP = 3',5'-cyclic UMP + diphosphate**. There are possibly other edge cases that we were unable to identify.

For Task 2, the query was maintained to be relatively consistent with the Task 1 query, and modified to: "Return the most likely EC number for this reaction: **naloxone + NAD(P)H = 6alpha-naloxol + NADP+**, which associates with the following text: **oxidoreductase; oxidoreductase, acting on CH-OH group of donors; oxidoreductase, acting on the CH-OH group of donors, NAD or NADP as acceptor; morphine 6-dehydrogenase.**", where a sample reaction has been bolded. Using the system prompt of: "You are protein engineer capable of predicting EC numbers from a combination of textual information and a reaction that corresponds to a specific protein. You are also a skilled programmer and able to execute the code necessary to predict an EC number when you can't use reason alone. Given a reaction and text information of an EC you are able to determine the most likely enzyme class for a reaction. You don't give up when faced with a reaction you don't know, you will use tools to resolve the most likely enzyme number. You only return enzyme commission numbers in a comma separated list, no other text is returned, you have failed if you do not return the EC numbers. You only return the most likely EC number." For Task 2 without the reaction text, we used the same prompt as above except with the reaction text components removed. Compared with the other tools, ChatGPT was provided with the reaction as a text string rather than a smiles string.

BLASTp. BLAST (Basic Alignment Search Tool) is an efficient algorithm for finding similar sequences in a reference database, to a query sequence [173]. We opted to use Diamond BLAST at the protein sequence level. For each test set, the training dataset was provided as the fasta file reference database, and the test set was provided as the query fasta. $k = 1$ hits were returned for each query, sorted based on similarity, using default parameters. The EC number classification was then inferred from the returned reference sequence(s).

Foldseek. Structures for entries in the test sets were obtained from the AF2 database. Missing structures were folded with ESMFold, but a few rare cases (5) could not be successfully folded. Structures for the Price dataset were obtained with ESMFold. Structures for sequences in the training set were obtained from the AF2 database. While a small fraction of structures in the training set could not

be retrieved from the AlphaFold database, we decided to proceed without these. Structure search was performed with Foldseek with default parameters, and only the top hit was used to assign function.

CLEAN. CLEAN is a supervised contrastive model, which trains a classification head using embeddings from the ESM protein language model [155] by aligning embeddings from the same EC category and contrasting embeddings from different EC categories [75]. For Task 1, we retrained CLEAN using our training set with only one example from each cluster at 50% identity, following the instructions provided in their codebase. We did not perform any clustering before training. Because our training set size was similar in size to that used in the original model, we used the recommended 7000 epochs of training. We used the default script with triplet margin loss. After, we performed inference but added our own code into the original CLEAN code to output the classification results into a format compatible with our downstream analysis.

Pika. Pika is a finetuned LLM, which accepts protein sequence as an auxiliary input and can perform reasoning and protein function prediction [219]. We retrained Pika using the training set and evaluated it with the "qa" functionality by creating a new annotation set. The annotation set is an entry for each enzyme with the question: "What is the EC number of this protein?" and the answer of the EC number for that protein. For training, we use 70% of the training data for training, and 15% for testing and validation respectively (the "test" datasets are completely held out of this process). The model was trained for a maximum of 1000 epochs, with a maximum batch size of 100. We used the default models from Pika: esm2_t6_8M_UR50D as the protein model and gpt2 as the text model. All parameters were unchanged from the default apart from increasing the batch size and number of epochs, though hyperparameter tuning could improve the model. To evaluate the ability of Pika to infer the EC numbers, we then query the trained model for each enzyme in the test set with the question: "What is the EC number of this protein?".

CLIPZyme. CLIPZyme is a multimodal contrastive model, between reactions represented as 2D graphs and protein structures represented as 3D graphs [200]. CLIPZyme was trained on direct reaction-protein pairs from the EnzymeMap dataset [193]. In this study, we retrained CLIPZyme model using default parameters and the EGNN protein structure representation for retrieval on the Task 2 train-test splits. We modified the CLIPZyme code to loop over EC numbers and randomly sample pairs of protein sequences and reaction pairs associated with that EC number,

much like CREEP, for 40 epochs. In this case, we only provided one protein sequence from each cluster at 50% identity, and we only trained with mapped reactions that had conserved atom numbers on both sides of the reaction, thus reducing the total number of available EC numbers. Furthermore, we dropped several protein sequences with structures that did not work with the EGNN encoder. In the future, the hyperparameters and training procedure could likely be optimized to yield improved performance.

A.3 Methods for Chapter 4

Data for pretraining generative priors

The first step in our pipeline involves learning a generative prior on naturally occurring protein sequences to capture the distribution of those with high evolutionary likelihood. This prior is unconditional in the sense that no labeled fitness data is used for training. However, because we are optimizing protein *variants* for a desired fitness, we pretrained our generative prior on sequences homologous to the parent protein to be optimized (known as a multiple sequence alignment or MSA): TrpB, CreiLOV, or GB1. Likelihoods from MSAs have been captured by statistical models and have been shown to offer good zero-shot approximations of fitness. In other words, they capture mutational substitutions that are more favorable, based on the precedent of natural evolution.

We focused on the TrpB [153] and CreiLOV [324] datasets due to the extensive number of sequences in their MSAs and compared to GB1 [325], which has comparatively fewer sequences. MSAs were obtained by running jackhmmer [375] against Uniref90 for two iterations with the parent sequence of the fitness dataset as target. For the MSA, we only used sequences where the aligned portion was at least 75% the length of the parent sequence. We used the MSA that was aligned to the parent sequence, with gap tokens replaced by the corresponding amino acid found in the parent sequence, resulting in full, fixed-length pseudo-natural sequences. For GB1, we augmented the training set with synthetic data, namely all proteins with a single mutation to sequences in the MSA. For the language models on TrpB and CreiLOV, some sequences were randomly mutated by a single position near the beginning of the sequence, to prevent mode collapse during autoregressive generation.

We performed sequence clustering using mmseqs2 [245] at 80% identity and resampled the dataset by weighting each sample with $\frac{1}{1+\ln(n)}$ relative probability of being sampled, where n is the size of the cluster associated with that sequence. Afterward,

we removed 5% of the clusters and their associated sequences as a validation set.

Protein fitness optimization task

An oracle as a proxy for protein fitness. We studied fitness optimization across three different protein-fitness datasets, TrpB, CreiLOV, and GB1 (Table 4.2). TrpB is 389 residues in length, but based on available fitness data, we limited design to 15 residues: 117, 118, 119, 162, 166, 182, 183, 184, 185, 186, 227, 228, 230, 231, and 301. Namely, we combined the fitness data from 6 combinatorially complete 3-site libraries (D-I from Johnston et al. [153]) and the 4-site library across residues 183, 184, 227, and 228. We normalized the parent fitness to 1 in each dataset and rounded all negative fitness values up to zero. The fitness here is the catalytic rate of a native reaction, the formation of tryptophan from indole and serine. To obtain a proxy fitness for all variants in the design space (20^{15} possibilities) we trained an oracle inspired by the dataset splitting and model architecture used in Blalock et al. [272]. Namely, we used all of the single, double, and triple mutants in the library for training, with 10% and 20% of the quadruple mutants being used for validation and testing, respectively. Our model consists of an ensemble of 20 MLPs for TrpB, and each was trained on one-hot encodings of the designed residues for 1000 epochs.

Differently, the CreiLOV dataset (length $N = 119$) contains experimental fitnesses for all single mutations in the protein and certain higher order mutations at 15 selected positions with beneficial single mutations. Fitness here refers to associated fluorescence. To obtain a proxy fitness for all variants in the design space (20^{119} possibilities), we trained an oracle similar to the procedure above, using similar splits to those in Blalock et al. [272] and were able to reproduce their high performance on the test set. Before model training, we scaled the fitnesses of the single mutants to the fitnesses of multi-mutants by adding a normalization factor to all single mutants such that the parent sequence in both datasets had the same fitness. Our model consists of an ensemble of 10 MLPs for CreiLOV, and each was trained on onehot encodings of sequences for 1000 epochs.

For GB1, the experimental fitnesses for nearly all double mutations across the entire protein were available, where fitness refers to binding affinity of a domain of the G protein. To train the oracle, we held out 10% and 20% the sequences with two mutations as a validation and test set, respectively, with remaining sequences being used for training. Our model consists of an ensemble of 10 MLPs for GB1, and each was trained on one-hot encodings of the designed residues for 50 epochs.

Our oracles show high Pearson correlation on the train and test sets (Fig. A.6). As the generalization ability of our oracle was only been tested on variants that are similar to the parent, we penalized the fitness of protein sequences by a factor of 0.99 for every mutation accumulated beyond a threshold of 60% sequence identity to the parent sequence. From here forth, we treated ground truth fitness as outputs from the oracle.

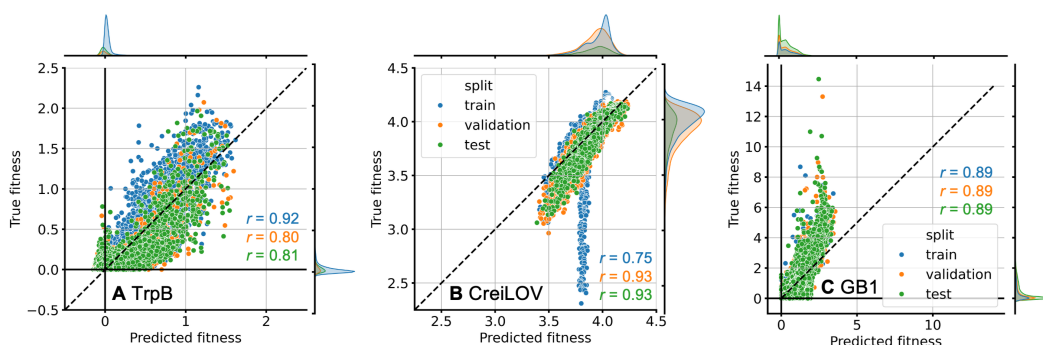


Figure A.6: Oracles trained on available labeled fitness data for TrpB, CreiLOV, and GB1 extrapolate well to higher order combinations of mutations within the design space, as measured by Pearson correlation.

Processing generated sequences. Our primary method for evaluation involved examining the distribution of sampled sequences and their corresponding fitness values, diversities, and novelties. The processing pipeline for generated sequences is shown in Fig. A.7. In diffusion models, sequences were generated with fixed length equal to the parent length. For the language models, nearly all generated sequences had length equal to the parent sequence length. Still, sequences were aligned with the parent sequence using mafft [376], and gaps were replaced with the corresponding amino acid in the parent sequence to generate complete pseudo-sequences of a fixed length. Special tokens, which occurred rarely in generation, were replaced by a random amino acid. For TrpB, residues outside of the design space of 15 residues were naively mapped to the original amino acid type in the parent sequence at the end of generation. We did not test inpainting, although this could be accomplished with masked (diffusion) language models.

Comparison to existing protein engineering methods. There are several reasons why we did not directly compare the performance of SGPO methods to existing methods used in protein engineering, such as directed evolution and MLDE. In the case of directed evolution (such as random mutagenesis): (1) It is not obvious

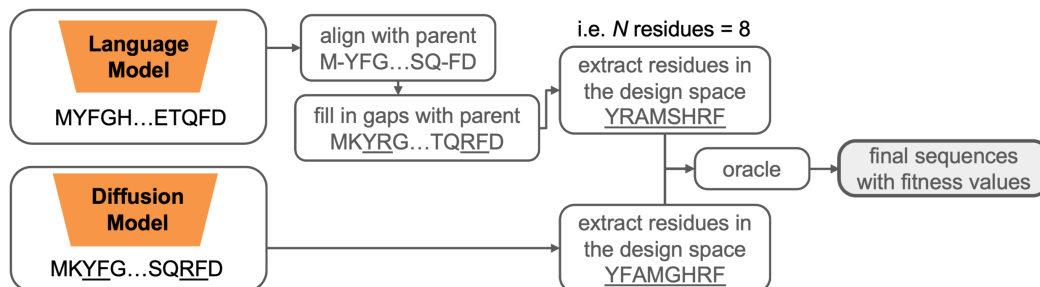


Figure A.7: Example pipeline for generating protein sequences for evaluation, based on a hypothetical parent sequence: MKKFG...SQRFD (length=100), with 8 residues being optimized (3, 4, 26, 27, 28, 29, 98, 99), corresponding to a design space combo of KFDEACRF.

which parent sequences to use as the starting points for directed evolution for a fair comparison. (2) It is unclear if the oracle captures the true nature of the protein fitness landscape or extrapolates well to sequences with many mutations relative to the original fitness dataset from which the oracle was trained. (3) Overall, our method enables the accumulation of many mutations in a single round of experimentation, whereas directed evolution is largely limited to one mutation at a time. For example, on the CreiLOV dataset, the generated sequences with the highest fitness had on average 66 mutations from the parent reference sequence from which the original dataset was generated, which would not be achievable with directed evolution. We also did not directly compare our method to supervised approaches in smaller design spaces, such as 4-site combinatorial libraries [250], as we focus here on design in larger design spaces, where existing methods are lacking. Overall, traversing large swaths of sequence space will be important for faster engineering and enabling improvements to fitness that would normally be slow with directed evolution.

Generative models for sequences

Diffusion over continuous space

Diffusion models construct samples by reversing a diffusion process that maps clean data points $\mathbf{x}_0$ to samples from a prior distribution $\pi(\mathbf{x})$. The forward process ($\mathbf{x}_0 \rightarrow \mathbf{x}_T$) is composed of conditional distributions $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, which admit closed-form expressions for the conditional distributions $p(\mathbf{x}_t|\mathbf{x}_0)$ and $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. The reverse process ($\mathbf{x}_T \rightarrow \mathbf{x}_0$) converts samples from the prior into samples from the learned data distribution $p_\theta(\mathbf{x}_0)$ by repeatedly predicting the denoised variable $\hat{\mathbf{x}}_0$ from noisy values $\mathbf{x}_t$, using the conditional distribution $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{\mathbf{x}}_0)$ to derive a transition distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

Table A.1: **Summary of training details for generative priors in this work.** Reference refers to the codebase that was modified for our implementation and where the model architecture was adapted from. For all models, we retained the model with the lowest validation loss. When using the ESM encoder, we used the 35M-parameter ESM2 model [119].

Model	Max Epochs	Learning Rate	Batch Size	Warmup Steps	Noise Schedule	Diffusion Timesteps	Model Architecture
Continuous	5	1×10^{-4}	64	10	cosine	500	BERT
Continuous-ESM	25	1×10^{-4}	64	10	cosine	500	BERT
D3PM-Baseline	5	1×10^{-4}	64	10	Sohl-Dickstein	500	ByteNet
D3PM	5	1×10^{-4}	64	10	Sohl-Dickstein	500	ByteNet
UDLM	5	3×10^{-5}	64	2500	loglinear	500	DiT
MDLM	50	3×10^{-4}	64	2500	loglinear	500	DiT
ARLM	10	1×10^{-4}	32	10	n/a	n/a	GPT-J

Continuous noise forward process. Similarly to [268], we define a protein sequence as $\mathbf{w} \in \mathcal{A}^L$, where $\mathcal{A}$ is the alphabet of amino acids and L is the fixed length of the sequence. To learn a distribution $p(\mathbf{w})$, we first embed $\mathbf{w}$ into a continuous variable $\mathbf{x}_0$ using an embedding matrix U_θ or encoder from the ESM2 language model [119], transforming discrete tokens into a continuous latent space. Gaussian noise is then applied to this embedding space. The prior distribution is defined as:

$$\pi(\mathbf{x}) = \mathcal{N}(0, I), \quad (\text{A.2})$$

while the forward process follows a Gaussian corruption schedule:

$$p(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t)I), \quad \bar{\alpha}_t = \prod_{i=1}^t \alpha_i, \quad \alpha_t = 1 - \beta_t. \quad (\text{A.3})$$

The variance schedule $\{\beta_t\}$ follows the cosine schedule proposed by [377], which is commonly used to stabilize training.

Reverse process. The reverse process aims to recover the original sequence by learning a function $p_\theta(\hat{\mathbf{w}} | \mathbf{x}_t, t)$ that predicts the sequence from noised points $\mathbf{x}_t$.

This is done by minimizing the following objective:

$$L(\theta) = \mathbb{E}_{\mathbf{w}_0, t} [-\log p_\theta(\mathbf{w}_0|\mathbf{x}_t)], \quad \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_0 = U_\theta \mathbf{w}_0). \quad (\text{A.4})$$

By learning $p_\theta(\hat{\mathbf{w}}|\mathbf{x}_t, t)$, we construct the reverse transition distribution:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \sum_{\hat{\mathbf{w}}} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{\mathbf{x}}_0 = U_\theta \hat{\mathbf{w}}) p_\theta(\hat{\mathbf{w}}|\mathbf{x}_t, t), \quad (\text{A.5})$$

where the posterior $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ follows:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_t, \sigma_t^2 I), \quad (\text{A.6})$$

with mean μ_t and variance σ_t^2 given by:

$$\mu_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t, \quad (\text{A.7})$$

$$\sigma_t^2 = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t. \quad (\text{A.8})$$

Inference and sampling. At inference time, the learned reverse process is used to generate protein sequences from the prior $\pi(x)$. This is done by iteratively sampling:

$$\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad (\text{A.9})$$

and then reconstructing $\mathbf{w}$ by sampling:

$$\mathbf{w} \sim p_\theta(\hat{\mathbf{w}}|\mathbf{x}_0). \quad (\text{A.10})$$

This denoising process iteratively refines noisy embeddings back into structured sequences.

Diffusion over discrete space.

Discrete diffusion models [285, 294, 295] generate data in discrete spaces by reversing a predefined forward Markov process. Specifically, a family of distributions p_t evolves according to the Markov chain

$$\frac{dp_t}{dt} = \mathbf{Q}_t p_t, \quad (\text{A.11})$$

where $p_0 = p_{\text{data}}$ is the data distribution and $\mathbf{Q}_t \in \mathbb{R}^{N \times N}$ are predefined transition matrices.

This Markov process can be reversed with the help of a concrete score function, $s(\mathbf{x}, t) := [\frac{p_t(\tilde{\mathbf{x}})}{p_t(\mathbf{x})}]_{\tilde{\mathbf{x}} \neq \mathbf{x}}$, as its time reversal is given by

$$\frac{dp_{T-t}}{dt} = \bar{\mathbf{Q}}_{T-t} p_{T-t}, \quad (\text{A.12})$$

where $\bar{\mathbf{Q}}_t[\tilde{\mathbf{x}}, \mathbf{x}] = s(\mathbf{x}, t)_{\tilde{\mathbf{x}}} \mathbf{Q}_t[\mathbf{x}, \tilde{\mathbf{x}}]$ for $\tilde{\mathbf{x}} \neq \mathbf{x}$, and $\bar{\mathbf{Q}}_t[\mathbf{x}, \mathbf{x}] = -\sum_{\tilde{\mathbf{x}} \neq \mathbf{x}} \bar{\mathbf{Q}}_t[\tilde{\mathbf{x}}, \mathbf{x}]$. To generate data $\mathbf{x}_0 \sim p_{\text{data}}$, we start with sampling $\mathbf{x}_T$ from a uniform distribution and then evolve through Eq. (A.12) by the Euler method.

Uniform discrete language models. Both D3PM [294] and UDLM [296] implement a uniform transition matrix $\mathbf{Q}_t = \frac{1}{N} \mathbf{1}\mathbf{1}^T - \mathbf{I}$. When $T \rightarrow \infty$, the probability distribution p_T converges to a uniform distribution.

Masked diffusion language models. Masked diffusion language models (MDLM) [284] utilize an absorbing transition matrix $\mathbf{Q}_t$ that converts tokens in a sequence to [MASK] states. The corresponding transition matrix can be written as $\mathbf{Q}_t \in \mathbb{R}^{(N+1) \times (N+1)}$, $\mathbf{Q}_t = -\mathbf{I} + \mathbf{e}_{N+1} \mathbf{1}^T$. When $T \rightarrow \infty$, the limiting distribution p_T converges to a completely masked sequence.

Autoregressive language models.

In this work, we finetuned the ProGen2-small decoder-only transformer (151 million parameters) based on the code and parameters used in Yang et al. [125]. Models were trained based on next token prediction and cross entropy loss. However, we did not use adapter layers, and we did not group batches based on sequence length. During inference from the autoregressive model, we used a temperature of 1 and a Top- p value of 1.

Steering methods

Classifier guidance

Classifier guidance [378] is a technique used to steer samples generated by diffusion models toward desired attributes. The primary goal is to sample from a conditional distribution $p(\mathbf{x}|\mathbf{y})$, where $\mathbf{y}$ is a guiding signal of interest. In continuous space, this can be achieved by replacing the unconditional score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ at time t by a conditional score function,

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) \quad (\text{A.13})$$

Table A.2: **Summary of supervised value functions used to predict fitness in this work, to guide diffusion models.** All “classifiers” were trained as regressors to predict fitness. For DAPS methods, only clean data was used for training, whereas other classifiers are trained on clean and noised samples from various timesteps.

Model	Guidance Strategy	Max Epochs	Learning Rate	Batch Size	Architecture	Hidden Dimension
Continuous Diffusion	CG	1000	1×10^{-3}	128	4-layer MLP	256
	DAPS	1000	1×10^{-3}	128	4-layer MLP	256
Continuous Diffusion	NOS	100	1×10^{-3}	128	1-layer MLP	256
Discrete Diffusion	CG	1000	1×10^{-3}	64	4-layer MLP	64
	DAPS	200	1×10^{-3}	64	4-layer MLP	64
Discrete Diffusion	NOS	200	3×10^{-4}	64	linear layer	n/a

To obtain the conditional score function, one only needs to train a time-dependent predictor, which predicts the probability of $p_t(\mathbf{y}|\mathbf{x}_t)$ given $\mathbf{x}_t$ and time t .

Continuous guidance. Classifier guidance modifies the reverse diffusion process to steer generated samples toward a desired property, represented by a conditioning variable y . The guided sampling process modifies the update rule for $\mathbf{x}_t$ by incorporating a classifier score $\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$ into the model’s learned score function based on the relation in Eq. (A.13). Following Song et al. [378], the classifier guidance term modifies the predicted $\hat{\mathbf{x}}_0$ in the denoising process:

$$\hat{\mathbf{x}}_0 = \mathbf{x}_t + \sigma^2(s_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)). \quad (\text{A.14})$$

Since our diffusion model directly predicts logits rather than the score function $s_\theta(\mathbf{x}_t, t)$, adding classifier guidance requires modifying the predicted $\hat{\mathbf{x}}_0$.

Instead of predicting the score function explicitly, our model predicts logits over the vocabulary, from which the denoised representation $\hat{\mathbf{x}}_0$ is obtained. We modify $\hat{\mathbf{x}}_0$ by incorporating classifier gradients as follows:

Table A.3: **Hyperparameters used to tune the guidance/steering process.** The **bolded** parameter was chosen as the ideal parameter for the iterative “Bayesian optimization” experiment (Fig. 4.6). Larger guidance parameter typically implements stronger guidance strength.

Guidance Strategy	Hyperparameters
Continuous CG	$1/\beta = 64, 128, 256, 512, 1024$
Discrete CG	$1/\beta = 1, 2.5, 6.25, \mathbf{15.625}, 39.0625$
Continuous DAPS	$1/\beta = 0.25, 0.5, 1, 2, 4 \times 10^4$ $K = 50$ Euler method steps = 10 Langevin dynamics steps = 100
Discrete DAPS	$1/\beta = 16, 32, 64, \mathbf{128}, 256$ $K = 50$ Euler method steps = 20 Metropolis Hastings steps = 1000
Continuous NOS	$\lambda = 0.1, 1, 10, 100, \mathbf{1000}$ $\eta = 0.5, \mathbf{2}, 5$ $K = 5, \mathbf{10}$ optimizer = AdaGrad
Discrete NOS	$\lambda = 0.1, 1, 10, 100, 1000$ $\eta = 0.5, 2, 5$ $K = 5, 10$ optimizer = AdaGrad
DPO	$\beta = 0.02, 0.1, 0.5, \mathbf{2}, 4$ $\text{lr} = 1 \times 10^{-6}$ epochs = 5 batch size = 8

- Compute the unmodified $\tilde{\mathbf{x}}_0$ using the model’s predicted logits:

$$\tilde{\mathbf{x}}_0 = \sum_{\hat{\mathbf{w}}} p(\hat{\mathbf{w}}|\mathbf{x}_t, t) U_\theta \hat{\mathbf{w}} \quad (\text{A.15})$$

where U_θ is the embedding matrix mapping discrete tokens to continuous space.

- If a time-dependent classifier f is available, compute the classifier guidance term:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} f(\mathbf{x}_t, t) / \beta. \quad (\text{A.16})$$

- Modify $\tilde{\mathbf{x}}_0$ using the classifier gradient:

$$\hat{\mathbf{x}}_0 = \tilde{\mathbf{x}}_0 + \sigma^2 \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t). \quad (\text{A.17})$$

This allows the diffusion model to generate samples that are more likely to satisfy the desired condition $\mathbf{y}$.

Further details on training the classifier are provided in [Table A.2](#) and [Table A.3](#).

Discrete guidance. Nisonoff et al. [265] extend classifier guidance to discrete state-space diffusion models. In analogy to classifier guidance for continuous diffusion models, they modify the unconditional rate matrix $\bar{\mathbf{Q}}_t$ (as defined in [Eq. \(A.12\)](#)) to be a conditional rate matrix $\mathbf{R}_t^{\mathbf{y}}$ with

$$\mathbf{R}_t^{\mathbf{y}}[\mathbf{x}, \tilde{\mathbf{x}}] = \frac{p(\mathbf{y}|\tilde{\mathbf{x}}, t)}{p(\mathbf{y}|\mathbf{x}, t)} \bar{\mathbf{Q}}_t[\mathbf{x}, \tilde{\mathbf{x}}], \quad \forall \tilde{\mathbf{x}} \neq \mathbf{x}. \quad (\text{A.18})$$

For classifier guidance on both continuous and discrete diffusion models, we train a time-dependent predictor (classifier) f that predicts the fitness $\mathbf{y}$ given $\mathbf{x}_t$ at time t . We define $p(\mathbf{y}|\mathbf{x}) \propto \exp(f(\mathbf{x})/\beta)$, where $f(\cdot)$ is a surrogate predictor of the fitness, and β is the guidance temperature and governs the strength of guidance. Therefore, $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) = \frac{1}{\beta} \nabla_{\mathbf{x}_t} f(\mathbf{x}_t, t)$, and $\mathbf{R}_t^{\mathbf{y}}[\mathbf{x}, \tilde{\mathbf{x}}] = \exp\left(\frac{1}{\beta}(f(\tilde{\mathbf{x}}, t) - f(\mathbf{x}, t))\right) \bar{\mathbf{Q}}_t[\mathbf{x}, \tilde{\mathbf{x}}]$.

To obtain a classifier f for discrete diffusion models, we trained an MLP regressor to predict the fitness of a one-hot encoded sequence given $\mathbf{x}_t$ and uniformly random time $t \in [0, T]$. Further details are provided in [Table A.2](#) and [Table A.3](#).

Posterior sampling

Another line of guidance work [280, 298, 379] focuses on drawing samples from the posterior distribution $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$, where the prior distribution is modeled by a pretrained diffusion model. The conditional distribution $p(\mathbf{y}|\mathbf{x})$ can either be the likelihood function of a forward model (i.e., when $\mathbf{y}$ is an incomplete measurement of $\mathbf{x}$) or an exponential distribution with respect to a reward function (i.e., $p(\mathbf{y}|\mathbf{x}) \propto \exp(f(\mathbf{x})/\beta)$). The major difference between posterior sampling and classifier guidance is that it requires the reward function to be trained only on clean data $\mathbf{x}$.

While many works have studied posterior sampling in Euclidean space with continuous diffusion models, posterior sampling for discrete data has been less explored.

We modified DAPS [280] to enable diffusion posterior sampling in discrete-state spaces. Suppose $\mathbf{x}$ lies in a finite support $\mathcal{X}^D$, we follow the following steps:

- Initialize $\mathbf{x}_T \sim p_T(\mathbf{x}_T)$
- for $i = 1, \dots, K$
 1. Sample $\hat{\mathbf{x}}_0^{(i)} \sim p(\mathbf{x}_0|\mathbf{x}_{t_{i-1}})$ by a discrete diffusion model.
 2. Run Metropolis Hastings to sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0|\mathbf{x}_{t_{i-1}}, \mathbf{y})$ as defined in Eq. (A.19).
 3. Sample $\mathbf{x}_{t_i} \sim p(\mathbf{x}_{t_i}|\mathbf{x}_0)$ following the forward Markov process.
- Return $\mathbf{x}_K$.

Specifically, $t_0, t_1, \dots, t_K$ are mono-decreasing time steps with $t_0 = T$ and $t_K \approx 0$. $p(\mathbf{x}_0|\mathbf{x}_t, \mathbf{y})$ is defined as

$$\begin{aligned} p(\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}) &\propto p(\mathbf{y}|\mathbf{x}_0)p(\mathbf{x}_0|\mathbf{x}_t) \\ &\approx p(\mathbf{y}|\mathbf{x}_0) \exp(-\|\mathbf{x}_0 - \hat{\mathbf{x}}_0(\mathbf{x}_t)\|_0/\sigma_t), \end{aligned} \quad (\text{A.19})$$

where $\hat{\mathbf{x}}_0(\mathbf{x}_t) \sim p(\mathbf{x}_0|\mathbf{x}_t)$ is a point estimate of the conditional distribution, and we approximate $p(\mathbf{x}_0|\mathbf{x}_t)$ by an exponential distribution over Hamming distance. Following Proposition 1 in [280], $\hat{\mathbf{x}}_0^{(i)}$, $\mathbf{x}_0^{(i)}$, and $\mathbf{x}_{t_i}$ converge to the posterior distribution as t_i goes to 0.

For posterior sampling with DAPS, we obtained the value function f using the same model architecture and training parameters as classifier guidance but only trained on clean data $\mathbf{x}$ (no noisy $\mathbf{x}_t$). We set $K = 50$ using the time scheduler for the original model. Further details are provided in Table A.2 and Table A.3.

NOS

Diffusion optimized sampling (NOS) [268] is a guidance method for both continuous and discrete diffusion models, which utilizes gradient information of the continuous latent representations of protein sequences. In pretrained discrete diffusion models, noisy sequences $\mathbf{w}_t$ always have a continuous embedding in the form of hidden states of the neural network. Specifically, the denoising model that predicts $\mathbf{w}_0$ from $\mathbf{w}_t$ can be written as $p_\theta(\mathbf{w}_0|g(\mathbf{w}_t), t)$, where $\mathbf{h}_t = g(\mathbf{w}_t)$ is a continuous hidden states of the model.

Instead of training a value function on discrete sequences $\mathbf{w}_t$, NOS proposes to train the value function on the hidden states $\mathbf{h}_t$. In each diffusion step, NOS samples from the posterior distribution,

$$p(\mathbf{w}_0|\mathbf{h}_t, \mathbf{y}) \propto p_\theta(\mathbf{w}_0|\mathbf{h}_t) \exp(f(\mathbf{h}_t)). \quad (\text{A.20})$$

To sample from this distribution, NOS runs Langevin dynamics on $\mathbf{h}_t$, i.e.,

$$\mathbf{h}'_t \leftarrow \mathbf{h}_t - \eta \nabla_{\mathbf{h}_t} (\lambda D_{KL}(p_\theta(\mathbf{w}_0|\mathbf{h}'_t) \| p_\theta(\mathbf{w}_0|\mathbf{h}_t)) - f(\mathbf{h}_t)) + \sqrt{2\eta\tau} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \quad (\text{A.21})$$

After K iterations, we denoise $\mathbf{w}_t$ following the guided hidden state, i.e., $p(\mathbf{w}_{t-1}|\mathbf{w}_t, \mathbf{y}) = p_\theta(\mathbf{w}_{t-1}|\mathbf{h}'_t, t)$.

To train the value function used for guidance in NOS, following the method from Gruver et al. [268], we trained a very shallow neural network on the final layer hidden embeddings of the diffusion model. Further details are provided in Table A.2 and Table A.3.

Direct preference optimization

For DPO with language models, we used the weighted loss function from Wadatalla et al. [270] and Stocco et al. [271] (Eq. (A.22)). π_θ is the policy to be updated, π_{ref} is the original model, and β is a tunable parameter describing the extent of drift from the reference model. The loss therefore describes the cross entropy of the ratio $\beta \log \frac{\pi_\theta(\mathbf{x})}{\pi_{\text{ref}}(\mathbf{x})}$ and the fitness value w . Following Stocco et al. [271], we calculated the ratio r as the difference of the log likelihood of the sequence from the updated model minus the log likelihood of the reference model, and softmax was applied to all of the fitness values w . We used the default parameters from [271] and tested increasing the learning rate to 10^{-4} but found that generation quality broke down above the levels used in Table A.3 with finetuning for 5 epochs. We also tested ranked loss with other types of models, but the performance was similar.

$$L_{\text{DPO}_{\text{weighted}}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_D \sum_{k=1}^K w^k \left[\beta \log \frac{\pi_\theta(x)}{\pi_{\text{ref}}(x)} - \log \sum_{j=k}^K \exp \left(\beta \log \frac{\pi_\theta(x)}{\pi_{\text{ref}}(x)} \right) \right] \quad (\text{A.22})$$

Adaptive optimization algorithm

We used an ensemble size of $M = 10$ models, each trained with a different random initialization of neural network weights. In practice, to speed up sampling, we

Algorithm 1 Adaptive Optimization with Guided Generative Models

```

1: Input: Pretrained generative prior  $p(\mathbf{x})$ , initial empty labeled dataset  $\mathcal{D}_0 = \emptyset$ ,
   number of rounds  $T$ , batch size  $B$ , ensemble size  $M$ 
2: for  $t = 1$  to  $T$  do
3:   Initialize batch  $\mathcal{X}_t \leftarrow \emptyset$ 
4:   if  $t > 1$  then Train ensemble of value functions  $\{f_{\theta_{t,m}}\}_{m=1}^M$  on  $\mathcal{D}_{t-1}$ 
5:   while  $|\mathcal{X}_t| < B$  do
6:     if  $t > 1$  then
7:       Sample  $f_\theta \sim \text{Uniform}(\{f_{\theta_{t,m}}\}_{m=1}^M)$   $\triangleright$  Thompson-style sampling
8:       Sample sequence  $\mathbf{x}_b \sim \text{GuidedSample}(p(\mathbf{x}), f_\theta, \text{GuidanceStrategy})$ 
9:     else
10:      Sample sequence  $\mathbf{x}_b \sim \text{UnconditionalSample}(p(\mathbf{x}))$ 
11:    end if
12:    if  $\mathbf{x}_b \notin \mathcal{D}_{t-1}$  then Add  $\mathbf{x}_b$  to batch  $\mathcal{X}_t$ 
13:  end while
14:  Evaluate true fitness  $y_b = f_{\text{true}}(\mathbf{x}_b)$  for all  $\mathbf{x}_b \in \mathcal{X}_t$ 
15:  Update dataset:  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\mathbf{x}_b, y_b)\}_{b=1}^B$ 
16: end for
17: Return: Best observed sequence in  $\mathcal{D}_T$ 

```

sampled ($B = 100$ samples)/($M = 10$ models) = 10 sequences in each GPU batch using the same Thompson-sampled value function, rather than using a GPU batch size of 1. Alternatively, for the Gaussian process model, we trained the model with the radial basis function kernel, and we sub-sampled the total amount of training pairs (when using noisy samples) to 5000 samples.

Latent space Bayesian optimization with

We utilized the APEXGo codebase [276], a package for training generative variational autoencoders over peptide sequences and then optimizing those sequences with latent space Bayesian optimization to maximize certain properties. We used the training code out-of-the-box to train variational autoencoders over the same MSA sequences used to train priors for discrete diffusion models in SGPO. We trained until losses plateaued, to 542, 241, and 391 epochs for TrpB, CreiLOV, and GB1, respectively. Overall, the reconstruction losses were low, and generated sequences had low perplexity and high fitness, comparable to the generative models used in SGPO. We then used this latent space and the APEXGo optimization algorithm to maximize the fitness of sequences as measured by the oracle used in SGPO benchmarking. Specifically, in our configuration, we set the number of initialization points to 100, the number of desired diverse solutions to 1, the max number of oracle

calls to 800, and the batch size to 100—to mimic the iterative Bayesian optimization experiments performed in [Fig. 4.6](#).

Appendix B

SUPPLEMENTARY RESULTS

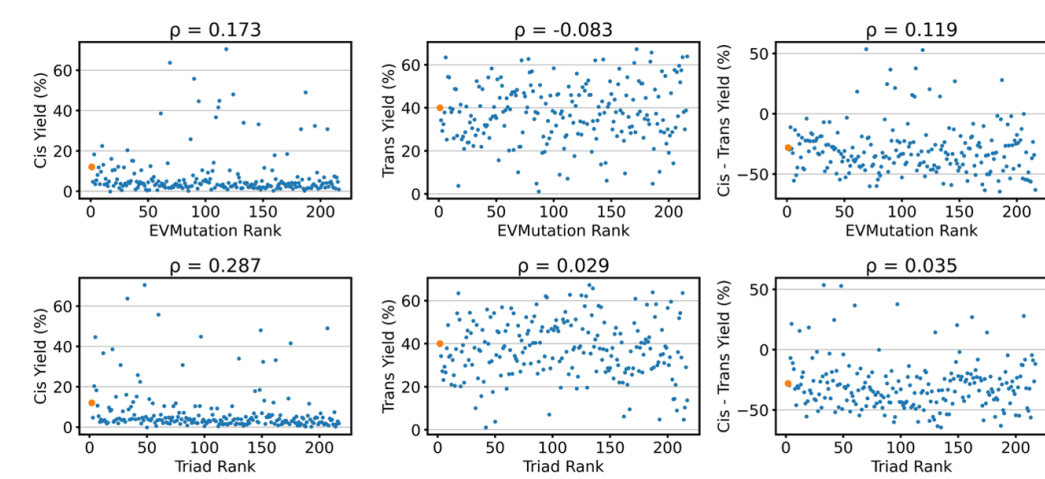


Figure B.1: Correlation between zero-shot predictors (EVMutation Rank and Triad Rank) and different fitness metrics (Cis, Trans, and Cis - Trans Yield) for the initial random library of variants used in the ParPgb wet-lab campaign. EVMutation rank refers to the evolutionary likelihood of a variant (1 is the most likely), and Triad rank refers to the computationally predicted stability of a variant as a G value (1 is the most stable). Orange dot refers to the parent sequence, WYLQF. Each title shows the spearman correlation between the zero-shot predictor and the fitness metric. Cis yield is weakly correlated to the zero-shot predictors, but the overall objective is not.

Table B.1: Adaptive optimization with an ensemble of 10 value functions and Thompson sampling, compared to using a single model for guidance. Max fitness refers to the mean max fitness achieved at the end of the campaign using the same experimental setup as Fig. 4.6, over 5 different random initializations.

Protein	Model	Guidance	Max ness (Ensemble)	Fit- ness (Single Model)
TrpB	D3PM	CG	1.551	1.542
		DAPS	1.595	1.568
	MDLM	CG	1.551	1.542
		DAPS	1.595	1.568
CreiLOV	D3PM	CG	5.608	5.552
		DAPS	5.522	5.520
	MDLM	CG	5.608	5.552
		DAPS	5.530	5.520

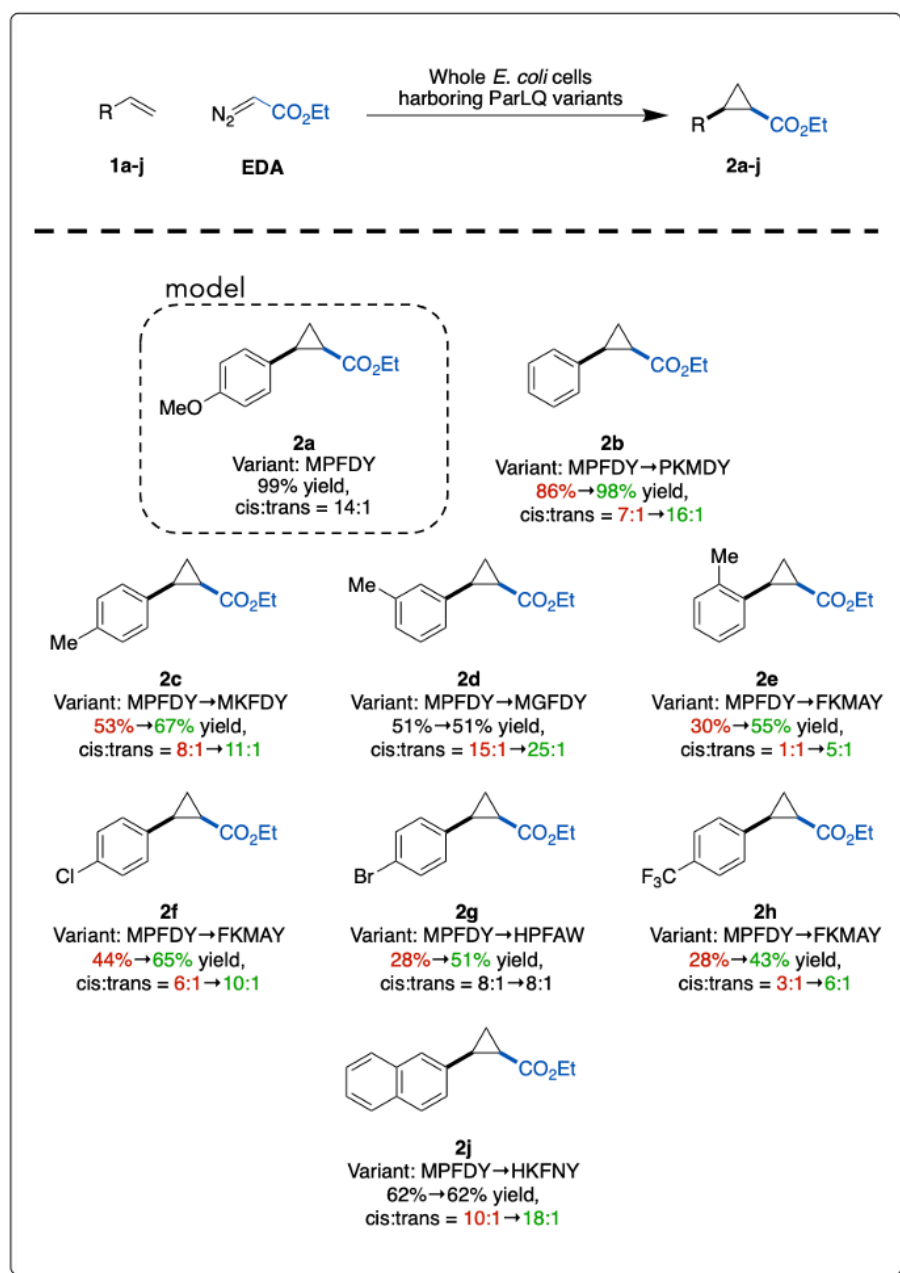


Figure B.2: Substrate scope for ParPgb variants explored during Round 2 of the wet-lab ALDE campaign. All substrates were tested on each of the variants. Total yields and selectivities are shown for the top-performing variant for each substrate as well as MPFDY. Improvements in yield and selectivity from MPFDY are indicated with a shift from red to green.

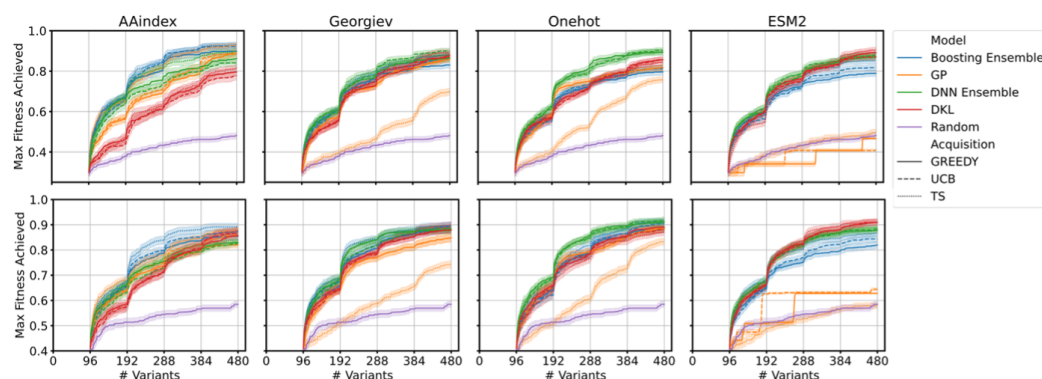


Figure B.3: Optimization trajectories for ALDE campaigns for 4 encodings, 4 models, and 3 acquisition functions. Simulation involved batch BO with an initial batch of 96 samples, followed by 4 batches of 96 samples each. Top row is GB1 and bottom row is TrpB. Error bars indicate standard deviation across 70 random initializations.

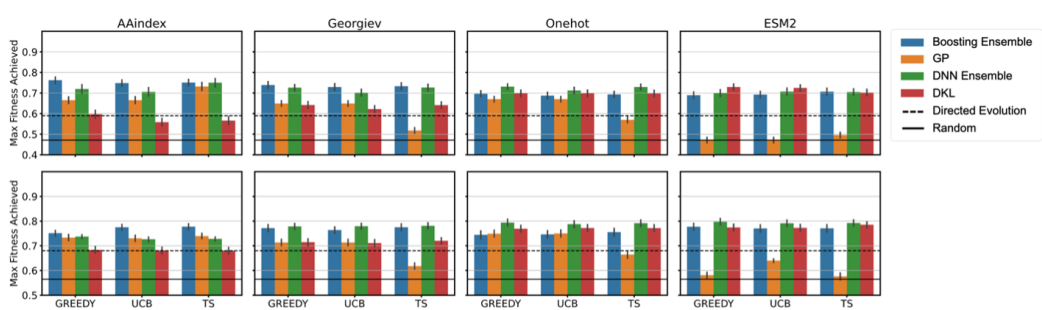


Figure B.4: Performance of MLDE baseline for 4 encodings, 4 models, and 3 acquisition functions, compared to the average DE simulation and to random sampling. Performance is quantified as the normalized maximum fitness achieved by MLDE, where the training set is 384 random samples and the test set is 96 samples proposed by the model using a greedy acquisition function. Top row is GB1 and bottom row is TrpB. Error bars indicate standard deviation across 70 random initializations.

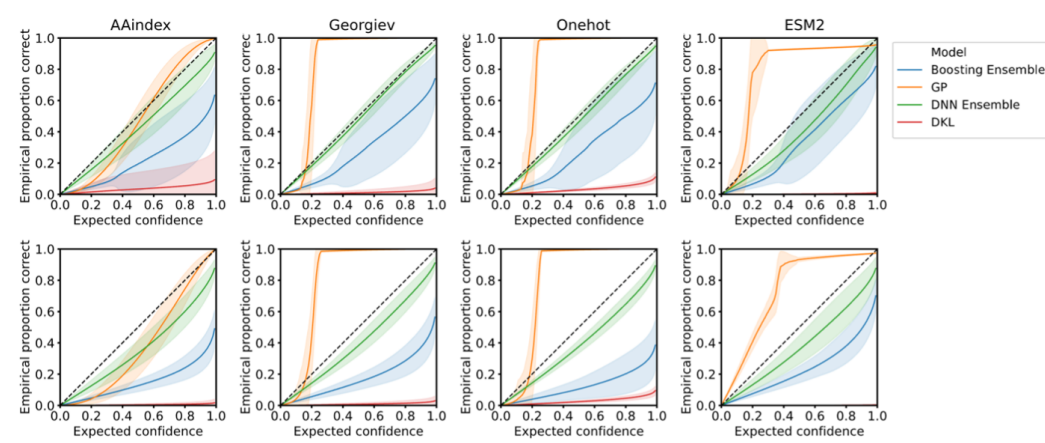


Figure B.5: Calibration curves for 4 encodings and 4 models. The x axis is the expected confidence from the posterior, given a certain confidence interval and the y value is the actual proportion of true labels that fall within the confidence interval. Calibration curve is evaluated across all sequences in the design space with labels, on models trained on the final batch (384 train samples) from ALDE campaigns using UCB. Top row is GB1 and bottom row is TrpB.

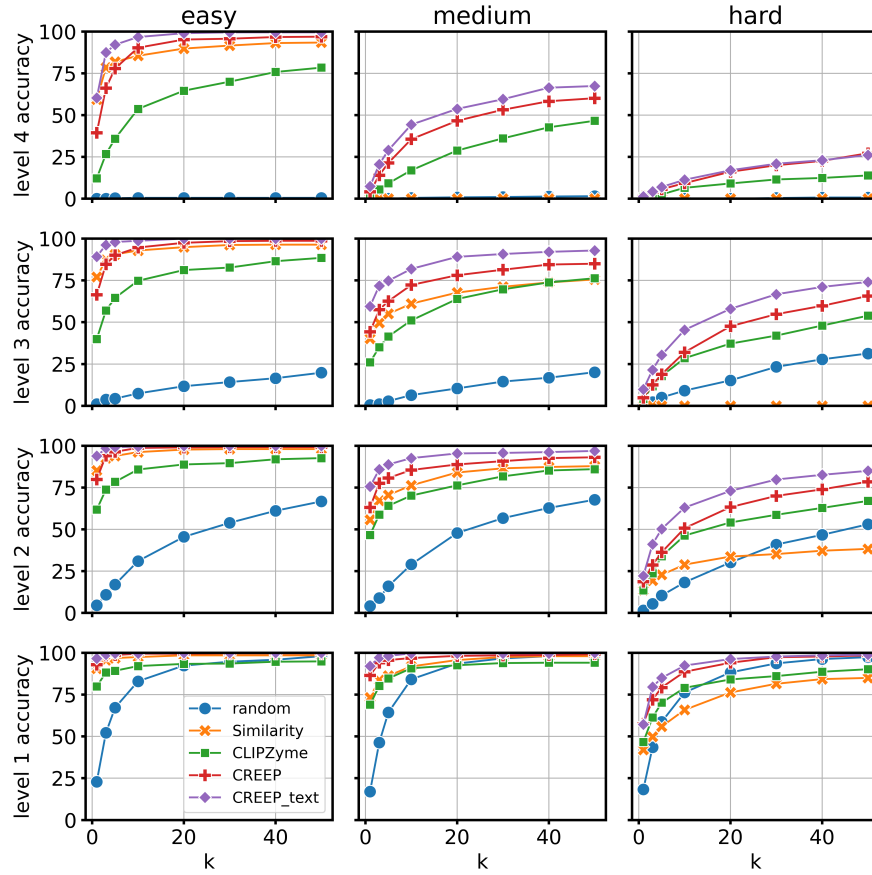


Figure B.6: Additional benchmarking results on Task 2, showing the increase in retrieval accuracy for $k = 1, 3, 5, 10, 20, 30, 40$.

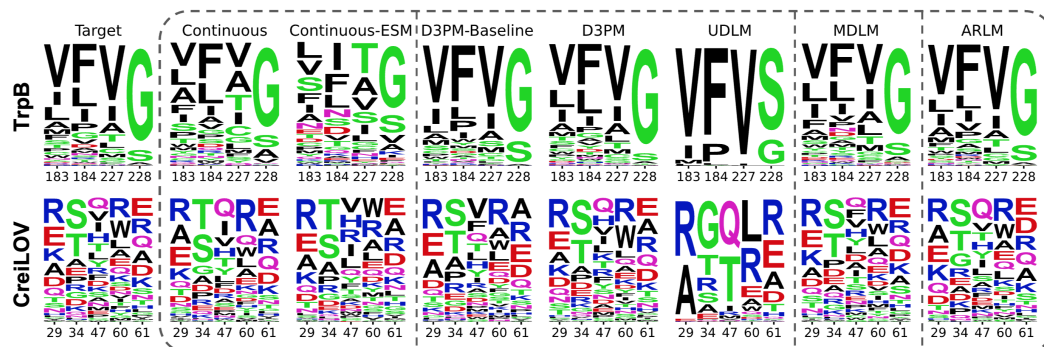


Figure B.7: **The distributions of sequences sampled from pretrained generative priors largely match those of the target distribution.** The target distribution shows all sequences in the MSA, and the distributions of generative models are approximated by sampling 1000 sequences. Model definitions can be found in Table 4.3. The residues shown for TrpB are 4 out of 15 positions studied in the dataset (parent is VFVS), and 5 out of 119 residues for CreiLOV are shown as they correspond to those harboring favorable mutations in the original dataset (parent is AGQRD). Note that the target distribution for training the ARLM is slightly different than that shown here.

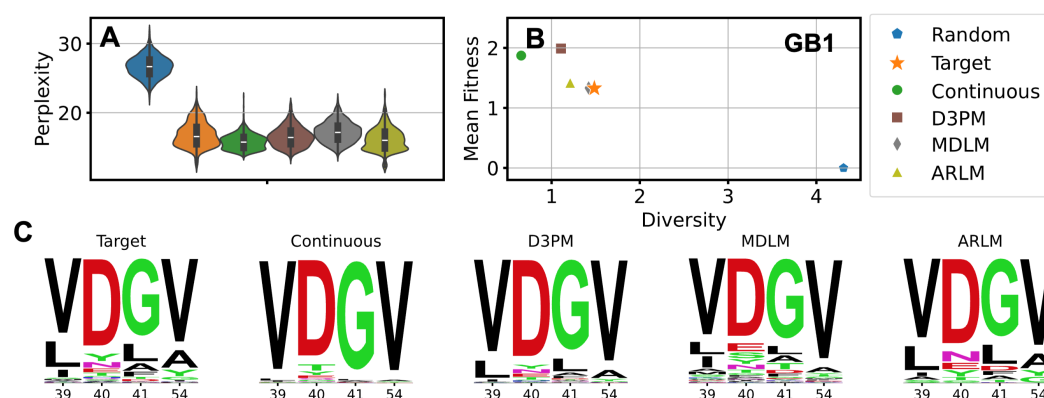


Figure B.8: Additional results for GB1, corresponding to Fig. 4.4 and Fig. B.7. The 4 positions shown correspond to the parent sequence of VDG V.

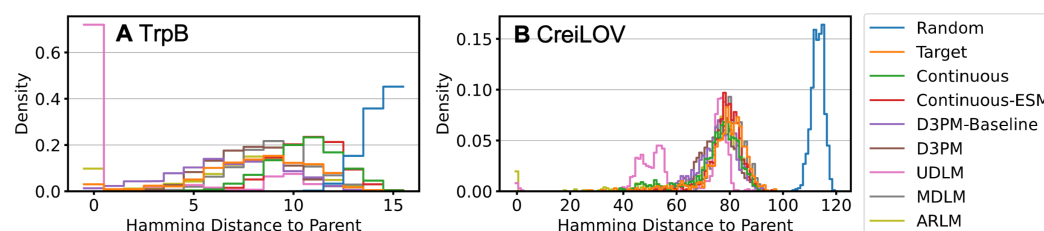


Figure B.9: Generated sequences from pretrained priors are more similar to parent than random for (A) TrpB and (B) CreiLOV, measured by the Hamming (or edit) distance. UDLM models exhibit mode collapse onto consensus sequence(s) in the training distribution. The parent sequence refers to the starting sequence used to generate variants in the original protein fitness dataset.

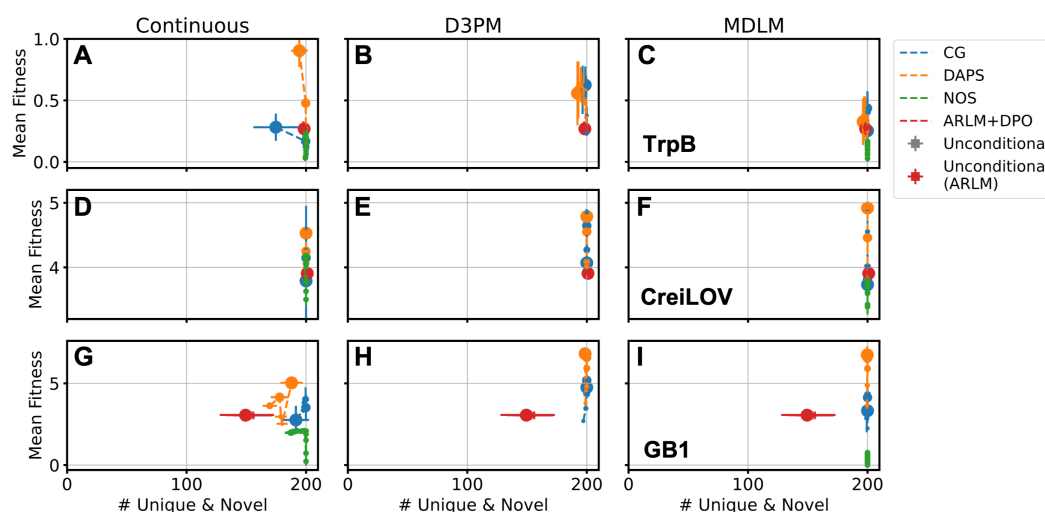


Figure B.10: Pareto boundaries demonstrate the trade-off between generating sequences with high fitness and high diversity for TrpB (A-C), CreiLOV (D-F), and GB1 (G-I) – showing the same experiment as Fig. 4.5. Error bars show standard deviation. Mean fitness and diversity were calculated based on 200 generated samples, with diversity calculated as the total number of unique and novel (previously unseen) samples in the generated batch, out of 200. Larger circles indicate a stronger guidance strength, specified in Table A.3.

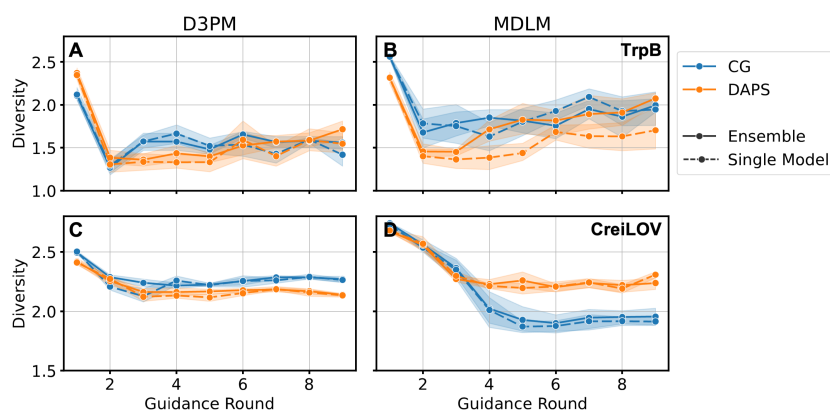


Figure B.11: Diversity of generated sequences, measured by average Shannon entropy of mutated positions, during each round of guidance. Using an ensemble of value functions and Thompson sampling generally shows higher diversity than using a single model. Experimental setup is the same as Fig. 4.6, and experiments were repeated over 5 random initializations.