Evaluation of the Generalizability of Machine Learning-Assisted Protein Engineering Methods

Thesis by

Francesca-Zhoufan Li

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

Caltech

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2025
(Defended May 12th, 2025)

© 2025

Francesca-Zhoufan Li
ORCID: 0000-0002-5710-9512

ACKNOWLEDGEMENTS

My Ph.D. has been quite the journey—one that certainly would not have been possible without the immense support of so many. Words can only begin to express the depth of my gratitude, both professionally and personally, to those who paved the way and stood by me throughout. First, I would like to thank my advisors, Prof. Frances Arnold and Prof. Yisong Yue. It has truly been a privilege to be mentored by both of you. Your guidance, encouragement, and support—through both breakthroughs and hardships—have been invaluable.

Frances, I became a fangirl after first learning about directed evolution during my junior year in college. I still vividly remember meeting you at Berkeley College of Chemistry Dean's Dinner in 2019 after you won the Nobel Prize. You continue to inspire me not only as an engineer and a scientist but also as a visionary and as a storyteller. Your keen eye for identifying impactful and tractable scientific problems, your swift and detailed feedback (even down to punctuation!), and your tireless commitment to science and public service are truly remarkable. Thank you for constantly challenging me to grow and for all the support you have provided.

Yisong, I aspired to work with you when I set out to explore the intersection of machine learning and protein engineering in my Ph.D. I was initially terrified—I had no formal CS or ML background, just a MATLAB class and some self-taught Python. Thank you for taking a chance on me, welcoming me into your group, and giving me unique opportunities such as serving as a volunteer chair at a top-tier ML conference. Your support, guidance, and resources enabled me to learn and grow in a new field. I admire your wide-ranging collaborations, your mentorship, and the wisdom you shared to help me navigate critical transitions. I will always remember to “commit and pivot,” not “spray and pray.”

To my thesis committee—Prof. Richard Murray, Prof. Steve Mayo, and Dr. Kevin Yang—thank you for your time, insight, and encouragement. Richard, thank you for an amazing first rotation in the middle of the pandemic, for your feedback on my fellowship proposal even before I officially joined, and for your guidance on navigating both scientific and administrative aspects of graduate life. Steve, I appreciate your expertise on computational protein design and your continued support. Kevin, I am still in awe that I got to be mentored by someone whose work I, and so many in the entire field, deeply admire. Your continued work at the frontier of ML for protein engineering and your community engagement are deeply inspiring.

To the Arnold Lab and the Yue Group: thank you for being my intellectual and emotional home. Dr. Sabine Brinkmann-Chen, I truly could not have done this without you. Thank you for your guidance on writing (and for helping me understand how my brain works!), and your unwavering support throughout it all. To the ML subgroup in the Arnold Lab: Dr. Bruce Wittmann, thank you for your guidance during my rotation and welcoming me to the group; Dr. Kadina Johnston, thank you for your scientific rigor and deep insight, and your invaluable support throughout science and life; Dr. Patrick Almhjell, thank you for helping with all my practice talks and I adore your dedication to visualizations; Dr. Ariane Mora, your mentorship and emotional support have meant so much; Yueming Long, joining LevSeq was pivotal for me—thank you for your generosity, kindness, energy, and leadership; Jason Yang, thank you for our many collaborations; Dr. Chenghao Liu, thank you for teaching me DFT and offering thoughtful feedback; Jarrid Rector-Brooks, for your computational expertise; Dr. Zach Wu and Dr. Lucas Schaus, for your guidance and professional advice. To the diligent and talented students that I had the pleasure to work with—Grace Liu, Emre Gürsoy, Lukas Radtke, and Mikhail Hameedi—thank you for letting me grow as a mentor. To the honorary member, Vignesh Bhethanabotla, thank you for our many physics-based discussions. And to the rest of the ML subgroup for the support and many discussions.

To my incredible experimental collaborators and mentors: Dr. Kathleen Sicinski, thank you for taking a chance on me and for your energy and positivity; Ravi Lal, thank you for being enthusiastic and generous with experiments; Deirdre Hanley, Dr. Ziqi Li, and Dr. Jae Kennemur, thank you for your patience in helping me understand your work; Dr. Nicholas Porter, thank you for teaching me PyMOL and sharing your structural biology knowledge. To my wholesome Arnold Lab community who brought support and joy: Dr. Edwin Alfonzo, Dr. Julia Reisenbauer, Dr. Casey Ritts—thank you for cheering me on; Ryen O'Meara, thank you for generously feeding me all the delicious treats and for brightening my days, along with Catherine Ko, Dr. Doris Taylor, Dr. Hayden Carder. Thanks to Dr. Ella Watkins-Dulaney, Dr. Nicholas Sarai, Dr. Shilong Gao, Dr. Juner Zhang, Dr. Anuvab Das, Dr. Julie Ducharme, Dr. Ziyan Zhang, Dr. Runze Mao, and so many others that I have had the pleasure to overlap with for professional and personal support. From group retreats and pig parties to holiday celebrations and softball leagues—you made this journey memorable, one that I will always cherish.

To the Yue Group, especially the AI4Science subgroup and those who brought immense energy to the group: Ivan Jimenez Rodriguez and Dr. Yujia Huang for helping me get started with compute; Geeling Chau, Sabera Talukder, Julie Wang, and Dr. Victor Dorobantu for organizing group events and bringing so much positive energy to the group; Dr. Jeremy Bernstein, Dr. Cameron Voloshin, Dr.

Uriah Israel, Dr. Jennifer Sun, Chris Yeh, and the alumni for your kindness, support, and wisdom. It was an honor to celebrate Yisong's ten-year anniversary with you all.

Thank you to the Biocircuits Group and my rotation mentor Dr. David Garcia. Despite the pandemic, my rotation was rewarding thanks to Dr. Zoila Jurado, Dr. Rory Williams, Dr. Chelsea Hu, Miki Yun, and alumni including Dr. Zachary Sun and Dr. Victoria Hsiao. Thank you to Cheryl Nakashima, Alma Rangel-Fuentes, Kenya Zeigler, and Suresha Guptha for handling logistics and compute support and for keeping everything running behind the scenes.

To my Microsoft Research internship mentors—Dr. Kevin Yang, Dr. Alex Lu, and Dr. Ava Amini—thank you for the phenomenal experience and for your patience and encouragement. I am also grateful to Dr. Sarah Alamdari, Dr. Nicolo Fusi, Dr. Lorin Crawford, Dr. Kristen Severson, Dr. Philip Rosenfield, Dr. David Alvarez-Melis, Dr. Sebastien Bubeck, Dr. Tristan Naumann, and many more exceptional researchers and mentors at MSR New England and Health Futures. Thank you to my intern cohort, especially Dr. Amy Wang, Dr. Kevin Wu, Nitya Thakkar, Dr. Wenjun Wu, Alexander Lin, and Dr. Kianoush Falahkheirkhah. Sean Whitzell, thank you for all the compute help. I was also incredibly honored to meet Dr. Susan Dumais, Dr. Eric Horvitz, Dr. Peter Lee, Dr. Desney Tan, and other inspirational leaders and outstanding researchers.

I am also grateful for the mentors, programs, and experiences beyond the research labs that supported me and enriched my time at Caltech. Dr. Jackie Gish, thank you for being there for me through every high and low. The Graduate Summer Research Institute, especially mentors like Dr. Zoila Jurado and Dr. Renée Wang, equipped me with resources and guidance that helped me begin my Ph.D. on solid footing. The Biotechnology Leadership Training Program, generously supported by the Donna and Benjamin M. Rosen Bioengineering Center, enriched my perspective on translational research and entrepreneurship and supported my professional growth. To my Generative and Experimental Perspectives for Biomolecular Design Workshop co-organizers: Dr. Pranam Chatterjee, Dr. Chenghao Liu, Jarrid Rector-Brooks, Dr. Sidney Lisanza, Soojung Yang, Lauren Hong, Jacob Gershon, Dr. Jason Yim, and Hannes Stark, thank you for the opportunity to build bridges between machine learning and experimental biology with such an inspiring and collaborative team.

To those who paved my path to Caltech—thank you. I truly would not have made it here without you. I am grateful for my first research experience in the Dueber Lab at UC Berkeley. Prof. John Dueber, thank you for the opportunity which helped open doors along my journey. Dr. Judy Savitskaya and

Dr. Tammy Hsu, thank you for teaching me so much. I am also grateful to the rest of the lab, including Dr. Ryan Protzko and Dr. Parry Grewal, for making it such a meaningful and formative experience. My time in the Koide Lab at NYU Langone Health further deepened my interest in research and solidified my decision to pursue a Ph.D. Thank you to Prof. Shohei Koide, Prof. Akiko Koide, and Dr. Tanya Panchenko for the opportunity and for your mentorship. To the rest of the group, Dr. Padma Akkapeddi, Prof. Takamitsu Hattori, Dr. Lauren Wallon, Dr. Larizbeth Romero, Dr. Kevin Teng, Dr. Nadia Leloup, Dr. Linxiao Chen, Alexis Corrado, Gayatri Ketavarapu, Eric Denbaum, and Steven Tsai, thank you for making it such a wonderful and welcoming group to work with. To Dr. Paolo Boccazzi and the team at Zymergen, including Dr. Janet Matsen and Dr. Crystal Humphries: thank you for your mentorship, for encouraging my transition into computational work, and for facilitating early professional connections. Thank you to my internship at Tierra, Dr. Zachary Sun, Dr. Louis Metzger, and Abel Chiao for introducing me to Caltech and connecting me with Prof. Richard Murray. To my mentors at Berkeley—Prof. Jamie Cate, Prof. Adam Arkin, Prof. Martin Head-Gordon, and Dr. Michelle Douskey—and to college advisors and staff like Catherine Bouvier Dang, Sanjeev Chahal, Tiffany Reardon, and Gwyneth Terry—thank you for all your support.

I owe deep thanks to the Biology Scholars Program (BSP) for believing in me and showing me that I belonged in STEM. Prof. John Matsui, Prof. Caroline Kane, Dr. Sabriya Rosemond, and Katherine Castro—thank you for helping me see a future I could not have imagined for myself. I still remember how shocked I was to learn that one could get paid to pursue a Ph.D., and it was through BSP that I was first introduced to bioengineering, research, networking, and so much more. Thank you also to Brook Yu, Karen Aguilar, Diana Flores, and Dr. Missy Soto for your dedication to supporting the BSP community, and to all my fellow BSPers for the strength, inspiration, and support you have offered over the years. To my communities at Women in Science and Engineering, Alpha Chi Sigma, and Bioengineering Honor Society—thank you for helping me grow professionally and personally.

I am so grateful to have so many friends and support systems who brought joy, strength, meaning, and balance to life outside the lab, including late-night rambles, outdoor adventures, food hangouts, and homework grinds. Your support has been instrumental in carrying me through every high and low along the way. To my Caltech friends not yet mentioned above, Joe Slote, Tess Marlin, Masami Hazu, Maria Derda, Angela Gao, Zach Singer, Berthy Fang, Pau Batlle, Kate Radford, Will Rosencrans, Dan Pollak, Mark Legendre, Ishaan Dev, Ryan Cosner, Nikhil Ranganathan, Sam Solomon, Laure Delisle, Margaret Trautner, Evan Mun, Elin Larsson, Manisha Kapasiawala, Zach Martinez, Mihai Toma, Matt Levine, Roey Lazarovits, Andy DeLaitsch, Victor Garcia, Jacob

Golomb, Richard Horak, Noah Epstein, Kevin Le, Jordan Thompson, Arun Chakravorty, Brian Seymour, Hristos Courellis, Soyoung Shin, Cyrus Fiori, Yusrah Kaudeer, Sorina Lupu, Sahangi Dassanayake, Erika Lee, Abe Polayes, Volkan Gurses, Sam Davis, Steven Bulfer and countless others—thank you for all the memories we have shared together. Thank you to Caltech Alpine Club for all the fun adventures, especially to Dr. Jackie Dowling and Dr. Shaelyn Silverman, to my Strava friends who inspire me to train, especially Simon Schirren, Maria Carilli, Pippa Richter, and to my Duolingo friends who inspire me to learn.

To my non-Caltech friends from Berkeley, high school, and beyond—Aviv Spinner, Lucas Waldburger, Chelsea Chen, Angela Kwon, Raya Kuo, Tiff Jann, Heidi Huang, Angie Lee, Yian Wu, Sharon Tian, Shirely Chen, Sarah Lee, Jing-Yi Chung, Mariana Verdugo, Matt Hauwiller, Charlene Jin, Howard Chiao, Jo Ung, Dante Eisele, Roger Chang, Mo Mora, Emmy Tian, Nick Lue, Manni He, Jessie Li, Michelle Zhang, Brenda Wang, Edward Mu, Tae Kim, Nikko Jeffreys, Kristyn Fudge, Ariel Tennenhouse, Brian Choi, Phil Kim, Jordan Baker, Leslie Leung, Shyam Bhakta, Makoto Lalwani, Christine Kyauk, Lynn Mymala, Stephanie Won, Glendy Ramon, Yessenia Hernandez-Cruz, Mario Eusebio, Luis Caldera, Luis Gutierrez, Melisa Camano, Sofia Santos, Stephanie De La Melena, Melissa Yao, William Shin, Julia Rogers, Robert Wang, Forrest Li, Renee Su, Justin Wang, Weiwei Zhang, Yiwen Chen, April Tang, Rocky Wang, and Dean Wasil—I am so grateful to have you all in my life. Thank you for always cheering me up, believing in me, and traveling to visit me!

To my lovely Boston summer crew, Theo Olausson, Parmida Davarmanesh, Katie Matton, Renato Berlinghieri, Michael Wang, Yael Kirkpatrick, Selina Guter, Matt Bowers, and the extended group, thank you for the memories and so warmly welcoming me into the friend group and so generously supporting me during my time there. To our wonderful neighbors and local community, Bob, Aaron, Billy, and our sweet landlady Ann for always looking out for us and checking in on me.

My grandma Yue'E He and my grandpa Joseph Li are two of the most important people in my life. Thank you for raising me, for being my role models, and for shaping me into the person I am today. Part of what kept me going was the silly but very sincere desire to let the world hear about you both. Grandma, thank you for always supporting whatever path I chose and for instilling in me the importance of a good education. It is such a shame you never had the chance to receive any formal education yourself. You are truly the smartest, most diligent, and resilient person I have ever known. Life forced you to grow up early—supporting yourself and your younger brother from an extremely young age—yet you taught yourself so much, from history and literature to medicine. You stay on top of current events and modern technology, and you never fail to impress younger generations with your everyday life hacks and wisdom. Grandpa, you were always grateful for growing up in a Catholic orphanage, and you carried that gratitude with you throughout your life. You taught me how to be generous and kind, even in hard times. I was always amazed by how engaged you were with your community and your commitment to service. Like Grandma, you too, enjoyed learning—and remained eager to learn new languages and piano well into your 80s. I miss you deeply, but I know a part of you lives on in me. You both endured such difficult circumstances, yet always fought relentlessly for what was right, stood firmly for what you believed in, and loved selflessly. I will never forget your strength, your values, your philosophies on life, and the endless love you gave me. I am forever grateful. Thank you to my parents for your growing understanding and the sacrifices you have made. Thank you to my extended family, especially Kirsten and Jan, for your constant support, encouragement, and gifts from across the globe.

To my partner Tim-Henrik Buelles: no words can ever capture my gratitude and everything we have been through together. You have been my rock through it all—from my ACL surgery to late-night deadlines, from unforgettable adventures to everyday moments like your better-than-Michelin-star homemade dinners. Thank you for your belief in me, your devoted and steady support, and all the memories we have created together. Your love, understanding, and presence have meant the world to me. I could not have done this without you. Thank you for everything you have done and for walking every step of this journey with me.

And to my car, Leh-fee, whose name resembles “blue-fly-away” in Shanghainese and “the girl” in French—thank you for giving me the freedom to take me on this ride through grad school.

ABSTRACT

Engineered proteins can carry out a vast array of functions and have become indispensable across numerous industrial applications. To accelerate wet-lab protein engineering efforts, machine learning-based methods have advanced rapidly. However, a gap remains between state-of-the-art machine learning methods and their practical adoption. A key factor contributing to this disconnect is the lack of application-relevant benchmarking and generalizable insights across protein engineering tasks. This thesis evaluates machine learning-assisted protein engineering approaches to identify generalizable strategies. The central problem considered is learning the mapping from protein sequence to function—known as the *fitness landscape*—to enable the prediction of unseen variant fitness. **Chapter 1** introduces the background and context for machine learning-assisted protein engineering and highlights the practical constraint of limited experimental budgets. **Chapter 2** investigates transfer learning, which leverages models pretrained on large protein sequence databases to generate informative representations for modeling task specific sequence-function relationships. Evaluation across ten diverse tasks shows that while transfer learning is effective in structure prediction, it underperforms in variant fitness prediction—a key objective in protein engineering. **Chapter 3** evaluates alternative strategies with a focus on combinatorial fitness landscapes, a common setting in protein engineering. Across 16 diverse landscapes, *focused training* improves the performance of various machine learning approaches by strategically selecting training variants using zero-shot predictors, which estimate variant fitness from auxiliary information without relying on experimental data. Building on these insights, **Chapter 4** addresses the specific challenge of engineering enzymes—proteins that convert substrates into products—for novel chemistries. While six general zero-shot predictors without substrate information can predict enzyme activity on non-native substrates, they fail on more out-of-distribution, *new-to-nature* chemistries. Incorporating substrate information into zero-shot predictors leads to more generalizable performance across all tested chemistries, spanning 22 substrates. **Chapter 5** provides a brief outlook on future directions. Overall, this thesis identifies generalizable strategies for machine learning-assisted protein engineering by systematically evaluating and improving how sequence-to-function relationships are modeled across diverse tasks.

PUBLISHED CONTENT AND CONTRIBUTIONS

1. **Li, F.-Z.**, Amini, A. P., Yue, Y., Yang, K. K. & Lu, A. X. Feature reuse and scaling: Understanding transfer learning with protein language models. *In Proceedings of the 41st International Conference on Machine Learning (PMLR),* **235**, 27351–27375 (2024). https://proceedings.mlr.press/v235/li24a.html.

F.-Z.L. participated in the conception of the project, curated the data, developed the software, implemented the methodology, performed experiments, conducted formal analysis of the results, generated visualizations, and participated in the writing of the manuscript.

2. **Li, F.-Z.**, Yang, J.[1], Johnston, K. E.[1], Gürsoy, E., Yue, Y. & Arnold, F. H. Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes. *bioRxiv* (2024). doi: 10.1101/2024.10.24.619774.

F.-Z.L. participated in the conception of the project, curated the data, developed the software and the methodology, performed experiments, conducted formal analysis of the results, generated visualizations, and wrote the manuscript.

3. **Li, F.-Z.**, Radtke, L. A., Johnston, K. E., Liu, C.-H., Yue, Y. & Arnold, F. H. Substrate-aware zero-shot predictors for non-native enzyme activities. *GEM Workshop, ICLR* (2025).

F.-Z.L. conceived the project, curated the data, developed the software and the methodology, performed experiments, conducted formal analysis of the results, generated visualizations, and wrote the manuscript.

[1] These authors contributed equally.

PUBLISHED CONTENT NOT INCLUDED IN THESIS

4. Yang, J., Ducharme, J., Johnston, K. E., **Li, F.-Z.**, Yue, Y. & Arnold, F. H. DeCOIL: Optimization of Degenerate Codon Libraries for Machine Learning-Assisted Protein Engineering. *ACS Synth. Biol.* **12**, 2444–2454 (2023).

F.-Z.L. generated embeddings and edited the manuscript.

5. Yang, J., **Li, F.-Z.** & Arnold, F. H. Opportunities and Challenges for Machine Learning-Assisted Enzyme Engineering. *ACS Cent. Sci.* **10**, 226–241 (2024).

F.-Z.L. participated in the discussion of the writing, generated visualizations, and edited the manuscript.

6. Long, Y.[2], Mora, A.[2], **Li, F.-Z.**[3], Gürsoy, E.[3], Johnston, K. E., Arnold, F. H. LevSeq: Rapid Generation of Sequence–Function Data for Directed Evolution and Machine Learning. *ACS Synth. Biol.* **14**, 230–238 (2025). https://doi.org/10.1021/acssynbio.4c00625.

F.-Z.L. developed the visualization software, generated visualizations, and edited the manuscript.

7. Yang, J., **Li, F.-Z.**, Long, Y., & Arnold, F. H. Illuminating the Universe of Enzyme Catalysis in the Era of Artificial Intelligence (Manuscript submitted for peer review).

F.-Z.L. participated in the discussion of the writing, generated visualizations, and edited the manuscript.

[2] These authors contributed equally
[3] These authors contributed equally

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABBREVIATONS

AAV. Adeno-associated virus.

AF3. AlphaFold 3.

AL. Active learning.

ALDE. Active learning-assisted directed evolution.

BERT. Bidirectional Encoder Representations from Transformers.

CARP. Convolutional Autoencoding Representations of Proteins.

CoVES. Combinatorial Variant Effects from Structure.

CPU. Central processing unit.

CNN. Convolutional neural networks.

DE. Directed evolution.

DHFR. Dihydrofolate reductase.

DMS. Deep mutational scanning.

DNN. Deep neural network.

ESM. Evolutionary Scaled Modeling.

ESM-IF. Evolutionary Scaled Modeling inverse folding.

FLIP. Fitness landscape inference for proteins.

ft. Focused training.

ftALDE. Focused training active learning-assisted directed evolution.

ftMLDE. Focused training machine learning-assisted directed evolution.

GB. Gigabyte.

GB1. G Protein Domain B1.

GPU. Graphics processing unit.

GVP. Geometric vector perceptron.

KDE. Kernel density estimation.

LoRA. Low-rank adaptation.

ML. Machine learning.

MLDE. Machine learning-assisted directed evolution.

MLM. Masked language modeling.

MSA. Multiple sequence alignment.

MSE. Mean squared error.

NDCG. Normalized Discounted Cumulative Gain

PDB. Protein Data Bank

PLM. Protein language model.

PLP. Pyridoxal 5'-phosphate.

RNA. Ribonucleic acid.

ROC-AUC. Area under the receiver operating characteristic curve.

Spearman ρ. Spearman rank correlation coefficient.

SSM. Site-saturation mutagenesis.

SSMuLA. Site-Saturation Mutagenesis Landscape Assistant.

SS3. Three-class secondary structure.

TEV. Tobacco etch virus.

TrpB. Tryptophan synthase β-subunit.

TS. Thompson sampling.

UCB. Upper confidence bound.

WT. Wildtype.

XGBoost. Extreme gradient boosting.

ZS. Zero-shot.

Chapter 1

MACHINE LEARNING-ASSISTED PROTEIN ENGINEERING

1.1 Introduction

Proteins—sequences composed of amino acid building blocks—carry out a vast array of critical biological functions. By modifying their sequences, proteins can be engineered to enhance their existing properties or perform entirely new functions. Engineered proteins have become indispensable across numerous applications, serving as effective therapeutics to combat diseases, non-toxic agents to enhance crops, and green biocatalysts to synthesize chemicals.[1] For example, engineering of adeno-associated virus capsid proteins has enabled the development of gene delivery vehicles with enhanced tissue specificity, immune evasion, and therapeutic efficacy[2] and engineering of cytochrome P450 has enabled the synthesis of pharmaceuticals and novel chemical bonds not accessible through traditional chemistry.[3]

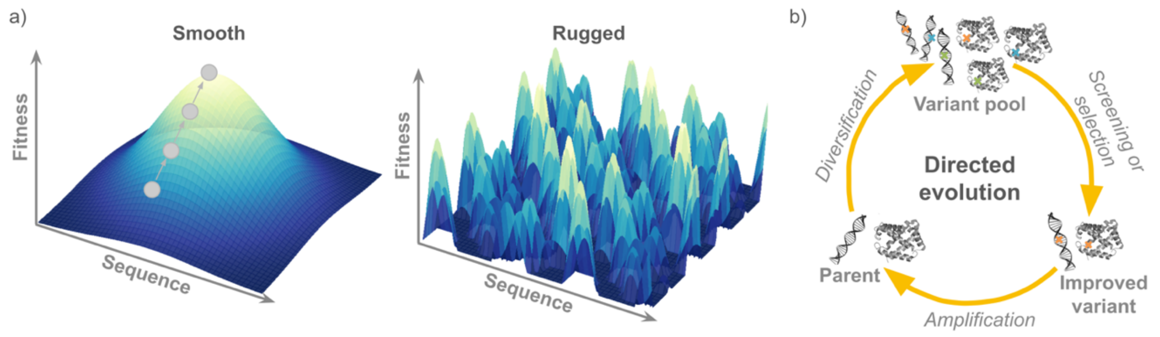


Figure 1.1. *Conceptual overview of protein fitness landscapes and the directed evolution process. a) Protein fitness landscapes map sequences to fitness values, where the landscape can be smooth (left) or rugged (right). In smooth landscapes, a global maximum may be reachable through successive single mutations with gradual fitness improvements. Rugged landscapes contain many local maxima, making navigating the landscape more challenging. b) Directed evolution is an iterative experimental strategy inspired by natural selection. It involves diversification of a parent sequence to generate a variant pool, followed by screening or selection to identify improved variants, which are then amplified and served as the parent for the next round.*

Protein engineering can be conceptually framed as navigating a "fitness landscape" (**Figure 1.1a**),[4,5] a high-dimensional surface where each point represents a unique protein variant defined by its amino acid sequence, and its height corresponds to a quantitative measure of the desired functional property—referred to as fitness (e.g., catalytic efficiency, binding affinity, or fluorescence). In a smooth or convex landscape, a global maximum can be reached through a greedy walk (i.e., a series of single amino acid substitutions that each improve fitness). By contrast, rugged landscapes containing many local maxima are much more difficult to traverse, as greedy steps may become trapped on suboptimal peaks.

The ultimate engineering objective is to achieve a variant sequence with high fitness for the desired objective. However, the vast size of protein sequence space (e.g., 20^L possible sequences for a protein of length L, where 20 is the number of canonical amino acids possible at each position) makes exhaustive search infeasible. Effective navigation of the fitness landscape, therefore, requires strategies that efficiently explore the sequence space and obtain variants with higher fitness values.

A widely adopted strategy for traversing the fitness landscape experimentally is directed evolution. Inspired by natural selection, directed evolution involves iterative cycles of (1) diversification of the protein sequences, through techniques such as random mutagenesis, site-saturation mutagenesis, or recombination, and (2) selection or screening, to identify improved variants for the subsequent round (**Figure 1.1b**).[6–8] The beneficial mutations are thus accumulated throughout the iterations of mutagenesis and functional assessment. Despite its widespread use and remarkable success, directed evolution remains time consuming and resource intensive: screening is expensive, and multiple rounds of mutation and screening are often needed to generate the desired improvements. Furthermore, many fitness assays—particularly those involving the synthesis of complex chemical products or *in vivo* activity in animal models—are inherently low-throughput, which restricts the number of variants that can be experimentally generated. Historically, collected negative fitness data could not be used because high sequencing cost limited access to the corresponding variant sequences.

These challenges are exacerbated when fitness landscapes are more rugged and difficult to traverse when we encounter non-additive effects of amino acid substitutions: epistasis.[9,10] Epistasis is often observed between mutations in close structural proximity[11] and is enriched at binding surfaces or enzyme active sites, due to direct interactions between residues, substrates, and/or cofactors, where residue-residue or residue-substrate interactions can lead to context-dependent mutational effects. For example, beneficial mutations in the context of the initial sequence may not be beneficial in combination with other mutations. Therefore, epistasis can present a significant challenge for directed evolution, which can cause campaigns to become trapped at local optima of the fitness landscapes.[7]

In response to these challenges, machine learning (ML)-assisted protein engineering approaches have emerged as powerful complements to traditional directed evolution.[12,13] ML models can learn sequence–fitness relationships from experimental data and predict the fitness of untested variants *in silico*, enabling prioritization of variants which high predicted fitness for experimental validation and thus facilitating more effective navigation of the fitness landscape and the identification of improved variants. Furthermore, in contrast to the incremental nature of directed evolution, ML-guided strategies can propose larger jumps in sequence space, helping to escape local optima and access more diverse regions of the landscape.

Recent advances in deep learning have dramatically improved our ability to model proteins along difference axes. Inspired by natural language processing, pretrained protein language models such as the Evolutionary Scale Modeling (ESM) family[14–17] and the ProGen family[18–20] leverage large sequence databases to learn generalizable representations that support specific downstream tasks. Protein structure-based models such as AlphaFold 3 have revolutionized protein structure prediction.[21] Generative models such as RFDiffusion[22,23] and ProteinMPNN[24] have expanded our ability to design proteins by generating backbone structures conditioned on user-defined constraints (e.g., motifs or targets) and by designing sequences conditioned on input backbone geometries, respectively. Although structure-based models do not directly encode protein function, they offer complementary structural insights that could inform sequence-function prediction. Similarly, although generative models are trained to produce plausible protein designs, the distributions of structures and sequences they learn are expected to reflect fundamental biophysical constraints, making model-derived likelihood scores potential proxies for protein fitness. Together, these advances have opened new opportunities for machine learning-assisted protein engineering.

However, despite these technological advances, a disconnect remains between the capabilities of state-of-the-art ML models and their practical impact on experimental protein engineering. There is a lack of application-relevant benchmarks and generalizable insights that reflect the resources constraints, optimization goals, experimental setups, and diversity of real-world engineering campaigns. These include limited screening budgets, the need to identify top-performing variants rather than maximize overall prediction accuracy, and the challenges of modeling combinatorial fitness landscapes arising from simultaneous site-saturation mutagenesis, a common engineering strategy where multiple positions are mutated simultaneously. This thesis addresses that gap by systematically evaluating ML approaches through a protein engineering-oriented lens, with a focus on strategies that are effective and generalizable across diverse tasks.

Specifically, this thesis evaluates the generalizability of machine learning-assisted protein engineering methods through three key questions (**Figure 1.2**). First, when does transfer learning with protein language models (PLMs) succeed? Transfer learning is an appealing approach, especially for data-scarce protein engineering tasks, as PLMs pretrained on large sequence databases can generate informative representations for modeling sequence–function relationships. However, **Chapter 2** shows that while transfer learning generalizes well for structure prediction, its benefits are limited in variant fitness prediction—particularly for combinatorial fitness landscapes that are common in protein engineering.

This motivates the second question: when and how should different ML strategies be applied for effective protein engineering? **Chapter 3** explores this question by evaluating alternative strategies across 16 combinatorial landscapes. It demonstrates that focused training improves the performance of various machine learning approaches by strategically selecting training variants using zero-shot (ZS) predictors, which estimate variant fitness from auxiliary information without experimental data. However, the datasets used in **Chapter 3** are limited to native or near-native functions.

The third question, then, asks which ZS predictors generalize to non-native activities, with a focus on the specific challenge of engineering enzymes—proteins that convert substrates into products—for novel chemistries. **Chapter 4** shows that while general-purpose ZS predictors can capture activities on non-native substrates, they do not generalize to more out-of-distribution, new-to-nature chemistries. In contrast, predictors that incorporate substrate information can generalize across 22 distinct substrates.

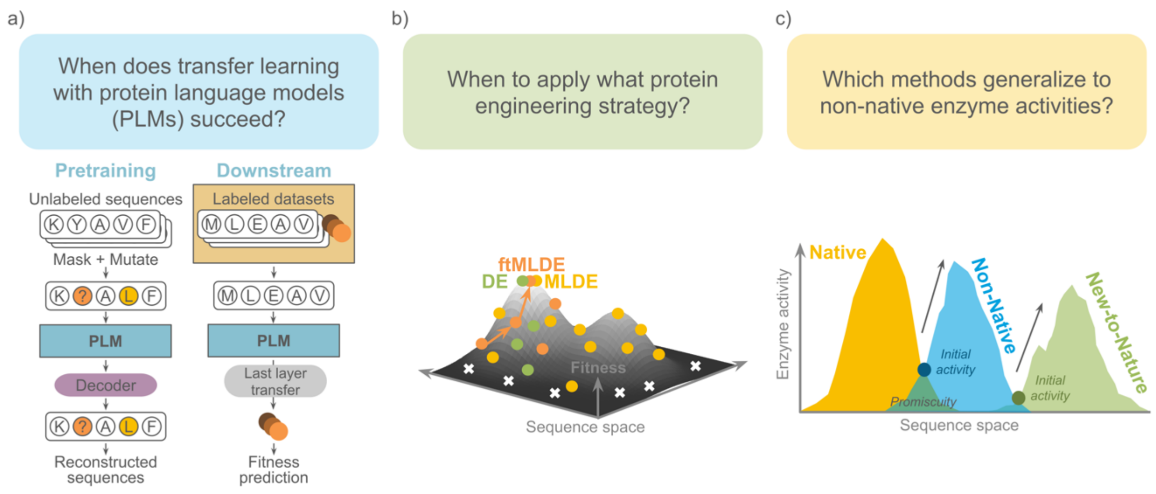


Figure 1.2. *Three main questions guiding this thesis. a) When does transfer learning with protein language models (PLMs) succeed?* ***Chapter 2*** *evaluates PLMs downstream task performances and finds that PLMs underperform in variant fitness prediction, especially for the combinatorial landscapes that are common in protein engineering. b) When to apply what protein engineering strategy?* ***Chapter 3*** *evaluates various ML-assisted strategies across 16 combinatorial fitness landscapes and shows that focused training—selecting training variants using zero-shot predictors—improves model performance especially under resource constraints. c) Which methods generalize to new-to-nature chemistries?* ***Chapter 4*** *extends these findings to enzyme engineering for non-native substrates and new-to-nature chemistries. While general-purpose zero-shot predictors fail on more out-of-distribution chemistries, incorporating substrate information yields more robust generalization across 22 substrates.*

The remainder of this chapter serves as overarching background for the rest of the thesis. **Section 1.2** introduces the use of machine learning models to approximate and navigate protein fitness landscapes, including supervised models and zero-shot predictors. **Section 1.3** describes the core components that influence ML performance in protein engineering, including data collection, sequence representation, learning strategies, and model evaluation. **Sections 1.4** and **1.5** outline the key application areas that motivate this thesis—epistatic fitness landscapes and enzyme engineering for non-native chemistries—while also highlighting the lack of benchmarking and generalizable insights in these challenging settings. **Section 1.6** summarizes the main findings from the three core projects (**Chapters 2–4**; **Figure 1.2**) and provides context for the outlook presented in **Chapter 5**.

1.2 Navigating protein fitness landscapes with supervised machine learning

To improve the efficiency and outcomes of protein engineering campaigns, supervised ML models have become a powerful tool to approximate and navigate protein fitness landscapes.[12,24–27] In supervised learning, a model is trained on labeled input–output pairs—here, protein sequences and their experimentally measured fitness values—to learn the relationship between them. Once trained, the model can be used to predict the fitness of new, untested variants. These predictions can help prioritize variants with high predicted fitness for experimental validation, reducing the need for exhaustive screening. Here, I focus on three key main aspects of a typical supervised ML pipeline for protein engineering: data collection, sequence representation, and learning strategy.

1.2.1 Assay-labeled data collection

Accurate fitness prediction depends critically on the quality, diversity, and size of labeled sequence-fitness datasets.[12,28] Traditionally, directed evolution does not require sequencing, as the best-performing variant from the previous round simply serves as the starting variant of the subsequent round. Sequencing is often reserved for identifying the best variant in each round and is typically not performed on the entire pool of variants due to time and cost constraints.

Recent advances in sequencing technology—lower costs, higher throughput, and faster turnaround—have made it feasible to sequence all variants in one round of a protein engineering campaign.[29,30] Tools developed in our lab now further support this adoption by combining high-throughput sequencing with automated analysis pipeline and interactive visualizations, pairing sequence and function data to provide actionable insights for experimentalists and further incentivizing the collection of comprehensive sequence–fitness datasets.[29]

Nevertheless, a key bottleneck remains to be overcome. While sequencing has become faster and cheaper, many functional assays remain low-throughput and resource-intensive. For example, quantifying product yield from enzymatic reactions often requires expensive analytical methods, and *in vivo* activity screening in animal models can take weeks or months. These intrinsic experimental bottlenecks limit both the number of assay-labeled variants available for model training and the predicted high-fitness variants that can be experimentally validated. Consequently, ML methods that can thrive in the *low-N regime*—where the number of labeled variants is small—are particularly valuable.[12] **Chapter 2** and **Chapter 4** explore how methods such as transfer learning and zero-shot prediction, which leverage unlabeled or auxiliary data, can improve learning in these challenging regimes.

1.2.2 Sequence representation

Protein sequences must be converted into numerical representations to serve as inputs to ML models to perform learning. A simple and widely used approach is one-hot encoding, which assigns a unique binary vector to each of the twenty canonical amino acids. Each vector is composed of "0"s except for a single "1" at the position corresponding to the amino acid's identity. While simple and commonly used as a baseline, one-hot encoding treats all amino acids equally, ignoring any underlying physicochemical properties. This can limit the model's ability to generalize, especially in low-data regimes where leveraging biochemical priors may be beneficial.

A more informative approach is to represent amino acids using numerical descriptors based on their physicochemical properties. The AAIndex database provides over 500 such descriptors, including experimentally measured properties such as hydrophobicity, steric bulk, and pKa values, as well as theoretical scores designed to capture functional or structural tendencies. These descriptors allow for more nuanced, domain-informed encodings of amino acids but their high dimensionality and redundancy across AAIndex features can introduce noise and prone to overfitting.[31] To address this, Georgiev encoding applies principal component analysis to a carefully selected subset of AAIndex features, reducing them to a small number of orthogonal components that capture the principal axes of biochemical variation among amino acids.[32]

Both one-hot and physicochemical encodings, however, are context independent: they represent each amino acid identically regardless of its position or role in the protein. However, the function of each amino acid in a protein is, in fact, extremely context dependent. For example, a tyrosine in a protein's active site that facilitates a chemical transformation performs a very different role than a tyrosine on the protein's surface. There is an analogy to natural languages: just as a word's meaning depends on sentence context, so too does a residue's effect depend on its structural and functional environment.

Pretrained protein language models (PLMs) address this limitation by learning context-aware representations from large-scale unlabeled sequence databases (e.g., UniRef).[33,34] Typically trained using a masked language modeling objective, these models are analogous to natural language transformers that fill in missing words in a sentence—learning a distribution over possible amino acids conditioned on surrounding residues.[14–16,35–38] The resulting embeddings encode evolutionary, structural, and functional priors that can be transferred to downstream prediction tasks. In supervised learning workflows for protein fitness prediction, PLM embeddings can be used either as frozen features, where the pretrained weights and features are passed unchanged into a downstream model, or through fine-tuning, where the model's parameters are updated during training on the labeled task.

Fine-tuning has often been shown to yield improved performance by adapting the representation to the specific functional landscape of interest.[39]

Despite their widespread adoption in recent years, however, it remains poorly understood why and when PLMs improve performance on diverse prediction tasks, especially for protein engineering tasks where a few amino acid substitutions are modified from a parent sequence (**Figure 1.2a**). Through a systematic evolution across a comprehensive suite of factors and tasks, **Chapter 2** shows that current PLMs often fail to generalize beyond structure-based tasks. This motivates the development and evaluation of alternative ML strategies tailored more specifically for protein engineering (**Chapter 3**).

1.2.3 Machine learning strategies

Once sequences are numerically encoded, they can be used to train supervised ML models to predict protein fitness. For a given design space, for example, mutating four residues to optimize fitness, a basic workflow begins with randomly sampling variant sequences from this design space and measuring their functional properties in the lab. These labeled sequence-fitness data points are then used to train a predictive model, which is subsequently applied to score all remaining variants in the design space. The top-ranked variants are then validated through the wet-lab experiments.

Instead of using a single iteration of training and testing, active learning breaks this process into multiple rounds.[40,41] At each round, the model is retrained with newly acquired labeled data and used to guide the selection of the next set of variants to test. Many active learning strategies rely on uncertainty quantification to prioritize variants, using methods such as Gaussian processes to estimate predictive confidence.[25] A prominent way to perform active learning is Bayesian optimization, which explicitly balances exploration and exploitation to efficiently identify high-performing variants.[42–44]

In both cases, the initial training set is often randomly sampled from the design space. Alternatively, focused training can be used to strategically select initial training variants based on zero-shot predictors[28]—methods that estimate variant fitness without any assay-labeled data.[17] Critically, the quality of the training data has been shown to impact the optimization outcome. Wittmann *et al.* (2021) demonstrated on the combinatorial G domain B1 (GB1) landscape[45] that focused training guided by zero-shot predictors outperforms standard supervised learning with randomly sampled training sets.[28]

Chapter 3 systematically benchmarks basic supervised learning, active learning, and focused training strategies across 16 diverse combinatorial fitness landscapes (**Figure 1.2b**). While all strategies outperform directed evolution, focused training guided by zero-shot predictors demonstrates consistently strong performance, particularly in the low-N regime relevant to protein engineering applications.

1.3 Zero-shot fitness prediction

Zero-shot (ZS) predictors estimate variant fitness without task-specific training data. Instead, they leverage prior assumptions (biologically motivated heuristics) and auxiliary information such as evolutionary conservation, structural heuristics, or likelihoods derived from pretrained protein language models or structure models. These predictors have augmented supervised models to identify higher-fitness variants, guided experimental data collection for ML model training, and scored *in silico* designs for reinforcement learning or experimental validation.[28,46–50] Recent benchmarks further highlight the broad applicability of ZS predictors.[51]

1.4 Combinatorial fitness landscape and benchmarks

ML in protein engineering has been demonstrated in different case studies. Most of the studies cover random mutations[50] (mostly single amino acid substitutions) spread across a protein, such as deep mutational scanning landscapes. In such settings, supervised ML models can often generalize well from a subset of single mutants to other single mutant variants.[52] In contrast, combinatorial landscapes—which enumerate all possible combinations of amino acid substitutions at a small number of functionally critical sites—

pose a greater challenge. These landscapes are often dominated by low-fitness variants, as the likelihood of retaining function decreases exponentially with the number of simultaneous mutations.[4] As a result, small, randomly sampled training sets are typically uninformative, containing few high-fitness variants. Models trained on such data struggle to identify promising candidates, limiting their practical utility in low-throughput experimental settings.[28]

Combinatorial mutagenesis is commonly used when engineering antibody binding interfaces or enzyme active sites, where mutations are typically in close structural proximity and interact with one another and the substrate. These regions tend to exhibit strong epistasis, with many combinations of individually beneficial mutations yielding non-additive or even deleterious effects. In contrast, mutations randomly distributed across the protein often exhibit near-additive behavior, making them more amenable to optimization using laboratory methods such as staggered extension process recombination.[53,54]

Because real-world protein engineering often involves such targeted, epistatic landscapes, they are of particular interest yet remain underrepresented in current ML benchmarking efforts. **Chapter 3** addresses this gap by systematically evaluating model generalization across 16 diverse combinatorial landscapes, focusing on three strategies mentioned in **Section 1.3**: basic supervised learning, active learning, and focused training guided by zero-shot predictors. These benchmarks provide practical insight into which strategies are most effective in the combinatorial, low-data regimes that are common in engineering real protein functions.

1.5 Generalization to protein engineering for non-native functions

One of the most compelling goals in protein engineering is to engineer proteins with novel functions, such as enzymes that catalyze reactions not observed in nature. These new-to-nature activities hold transformative potential in sustainable synthesis, therapeutics, and biotechnology. For example, the tryptophan synthase β-subunit (TrpB) catalyzes a native reaction between L-serine and indole to form tryptophan. Engineered TrpBs extend this function to non-native substrates such as serine analogues and substituted indoles, enabling the synthesis of tryptophan analogs and other noncanonical amino acids that are important precursors to pharmaceuticals and natural products.[55–58] Another example, heme-containing enzymes have been engineered to carry out a plethora of valuable reactions that have not been found in biological systems.[59] These new-to-nature reactivities include carbene transfers for stereoselective olefin cyclopropanation, traditionally requiring unsustainable transition metals,[3,60] and the formation of carbon–silicon[61] and carbon–boron bonds.[62]

Directed evolution has successfully evolved enzymes for new-to-nature activities, typically by generating targeted combinatorial libraries at enzyme active sites to enable such new chemistries. However, assessing variant activities often remains a major bottleneck due to the inherent constraints of low-throughput screening methods. As shown in **Chapter 3**, focused training with ZS predictors can excel in such low-N settings. However, standard ZS predictors are typically trained on natural sequences and do not explicitly incorporate substrate features or reaction mechanisms. In fact, these ZS predictors have anecdotally shown low predictive power for non-native enzyme activity.[40] Specifically, the general zero-shot predictors studied in **Chapter 3** do not account for the unique mechanistic features of enzymatic reactions. Enzyme catalysis involves complex steps such as substrate binding and transition-state stabilization; however, many ZS methods do not explicitly encode substrate or transition-state properties—features that are likely essential for predicting new-to-nature chemistries.

Chapter 4 thus evaluates the six general zero-shot predictor studied in **Chapter 3** alongside ten substrate-aware ZS predictors derived from generative modeling, molecular docking, and active-site heuristics (**Figure 1.2c**). This is tested on a newly generated dataset on 11 non-native substrates and three curated new-to-nature reactions with 11 additional substrates. **Chapter 4** shows that while six general zero-shot predictors can predict enzyme activity on non-native substrates, they fail to generalize to more out-of-distribution, new-to-nature chemistries. Incorporating substrate information into zero-shot predictors offer new-to-nature chemistry insights. Furthermore, predictor ensembles improve generalizations across all tested chemistries.

1.6 Summary of Chapter 1

This chapter introduces the background and motivation for evaluating the generalizability of ML-assisted protein engineering methods. A central gap identified is the lack of application-relevant benchmarks and generalizable insights, particularly under constraints such as limited labeled data, epistatic fitness landscapes, and the need to engineer proteins for non-native functions. This thesis addresses that gap through three interconnected projects: evaluating transfer learning with protein language models, benchmarking ML strategies for combinatorial fitness landscapes, and assessing generalization to non-native enzyme activities. Together, these studies aim to develop practical, application-aware methods for real-world protein engineering.

1.7 Bibliography for Chapter 1

1. Lutz, S. & Iamurri, S. M. Protein engineering: Past, present, and future. *Methods Mol. Biol. Clifton NJ* **1685**, 1–12 (2018).
2. Wang, J.-H., Gessler, D. J., Zhan, W., Gallagher, T. L. & Gao, G. Adeno-associated virus as a delivery vector for gene therapy of human diseases. *Signal Transduct. Target. Ther.* **9**, 1–33 (2024).
3. Renata, H., Wang, Z. J. & Arnold, F. H. Expanding the enzyme universe: Accessing non-natural reactions by mechanism-guided directed evolution. *Angew. Chem. Int. Ed.* **54**, 3351–3367 (2015).
4. Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat. Rev. Mol. Cell Biol.* **10**, 866–876 (2009).
5. Maynard Smith, J. Natural selection and the concept of a protein space. *Nature* **225**, 563–564 (1970).
6. Arnold, F. H. Directed evolution: Bringing new chemistry to life. *Angew. Chem. Int. Ed.* **57**, 4143–4148 (2018).
7. Packer, M. S. & Liu, D. R. Methods for the directed evolution of proteins. *Nat. Rev. Genet.* **16**, 379–394 (2015).
8. Wang, Y. *et al.* Directed evolution: Methodologies and applications. *Chem. Rev.* **121**, 12384–12444 (2021).
9. Starr, T. N. & Thornton, J. W. Epistasis in protein evolution. *Protein Sci.* **25**, 1204–1218 (2016).
10. Miton, C. M., Buda, K. & Tokuriki, N. Epistasis and intramolecular networks in protein evolution. *Curr. Opin. Struct. Biol.* **69**, 160–168 (2021).
11. Anishchenko, I., Ovchinnikov, S., Kamisetty, H. & Baker, D. Origins of coevolution between residues distant in protein 3D structures. *Proc. Natl. Acad. Sci.* **114**, 9122–9127 (2017).
12. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).

13. Yang, J., Li, F.-Z. & Arnold, F. H. Opportunities and challenges for machine learning-assisted enzyme engineering. *ACS Cent. Sci.* **10**, 226–241 (2024).

14. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci.* **118**, (2021).

15. Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).

16. Hayes, T. *et al.* Simulating 500 million years of evolution with a language model. *Science* **0**, eads0018 (2025).

17. Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. in *Advances in Neural Information Processing Systems* vol. 34 29287–29303 (Curran Associates, Inc., 2021).

18. Madani, A. *et al.* ProGen: Language modeling for protein generation. *bioRxiv* 2020.03.07.982272 (2020) doi:10.1101/2020.03.07.982272.

19. Madani, A. *et al.* Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.* **41**, 1099–1106 (2023).

20. Bhatnagar, A. *et al.* Scaling unlocks broader generation and deeper functional understanding of proteins. 2025.04.15.649055 Preprint at https://doi.org/10.1101/2025.04.15.649055 (2025).

21. Abramson, J. *et al.* Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630**, 493–500 (2024).

22. Watson, J. L. *et al.* De novo design of protein structure and function with RFdiffusion. *Nature* 1–3 (2023) doi:10.1038/s41586-023-06415-8.

23. Ahern, W. *et al.* Atom level enzyme active site scaffolding using RFdiffusion2. 2025.04.09.648075 Preprint at https://doi.org/10.1101/2025.04.09.648075 (2025).

24. Xu, Y. *et al.* Deep dive into machine learning models for protein engineering. *J. Chem. Inf. Model.* **60**, 2773–2790 (2020).

25. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl. Acad. Sci.* **110**, E193–E201 (2013).

26. Bryant, D. H. *et al.* Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39**, 691–696 (2021).

27. Saito, Y. *et al.* Machine-learning-guided mutagenesis for directed evolution of fluorescent proteins. *ACS Synth. Biol.* **7**, 2014–2022 (2018).
28. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026-1045.e7 (2021).
29. Long, Y. *et al.* LevSeq: Rapid generation of sequence-function data for directed evolution and machine learning. *ACS Synth. Biol.* **14**, 230–238 (2025).
30. Wittmann, B. J., Johnston, K. E., Almhjell, P. J. & Arnold, F. H. evSeq: Cost-effective amplicon sequencing of every variant in a protein library. *ACS Synth. Biol.* **11**, 1313–1324 (2022).
31. Kawashima, S. *et al.* AAindex: Amino acid index database, progress report 2008. *Nucleic Acids Res.* **36**, D202–D205 (2007).
32. Georgiev, A. G. Interpretable numerical descriptors of amino acid space. *J. Comput. Biol.* **16**, 703–723 (2009).
33. Bepler, T. & Berger, B. Learning the protein language: Evolution, structure, and function. *Cell Syst.* **12**, 654-669.e3 (2021).
34. Ofer, D., Brandes, N. & Linial, M. The language of proteins: NLP, machine learning & protein sequences. *Comput. Struct. Biotechnol. J.* **19**, 1750–1758 (2021).
35. Yang, K. K., Fusi, N. & Lu, A. X. Convolutions are competitive with transformers for protein sequence pretraining. *Cell Syst.* S2405471224000292 (2024) doi:10.1016/j.cels.2024.01.008.
36. Elnaggar, A. *et al.* ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
37. Brandes, N., Ofer, D., Peleg, Y., Rappoport, N. & Linial, M. ProteinBERT: A universal deep-learning model of protein sequence and function. *Bioinformatics* **38**, 2102–2110 (2022).
38. Elnaggar, A. *et al.* Ankh: Optimized protein language model unlocks general-purpose modelling. Preprint at https://doi.org/10.48550/arXiv.2301.06568 (2023).
39. Schmirler, R., Heinzinger, M. & Rost, B. Fine-tuning protein language models boosts predictions across diverse tasks. *Nat. Commun.* **15**, 7407 (2024).

40. Yang, J. *et al.* Active learning-assisted directed evolution. *Nat. Commun.* **16**, 714 (2025).

41. Vornholt, T. *et al.* Enhanced sequence-activity mapping and evolution of artificial metalloenzymes by active learning. *ACS Cent. Sci.* (2024) doi:10.1021/acscentsci.4c00258.

42. Rapp, J. T., Bremer, B. J. & Romero, P. A. Self-driving laboratories to autonomously navigate the protein fitness landscape. *Nat. Chem. Eng.* **1**, 97–107 (2024).

43. Qiu, Y., Hu, J. & Wei, G.-W. Cluster learning-assisted directed evolution. *Nat. Comput. Sci.* **1**, 809–818 (2021).

44. Qiu, Y. & Wei, G.-W. CLADE 2.0: Evolution-driven cluster learning-assisted directed evolution. *J. Chem. Inf. Model.* **62**, 4629–4641 (2022).

45. Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife* **5**, e16965 (2016).

46. Li, F.-Z. *et al.* Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes. 2024.10.24.619774 Preprint at https://doi.org/10.1101/2024.10.24.619774 (2024).

47. Landwehr, G. M. *et al.* Accelerated enzyme engineering by machine-learning guided cell-free expression. *Nat. Commun.* **16**, 865 (2025).

48. Johnson, S. R. *et al.* Computational scoring and experimental evaluation of enzymes generated by neural networks. *Nat. Biotechnol.* 1–10 (2024) doi:10.1038/s41587-024-02214-2.

49. Stocco, F. *et al.* Guiding generative protein language models with reinforcement learning. Preprint at https://doi.org/10.48550/arXiv.2412.12979 (2024).

50. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* **40**, 1114–1122 (2022).

51. Notin, P. *et al.* ProteinGym: Large-scale benchmarks for protein fitness prediction and design. *Adv. Neural Inf. Process. Syst.* **36**, 64331–64379 (2023).

52. Notin, P. *et al.* Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. in *Proceedings of the 39th International Conference on Machine Learning* 16990–17017 (PMLR, 2022).

53. Zhao, H., Giver, L., Shao, Z., Affholter, J. A. & Arnold, F. H. Molecular evolution by staggered extension process (StEP) in vitro recombination. *Nat. Biotechnol.* **16**, 258–261 (1998).

54. Almhjell, P. J. *et al.* The β-subunit of tryptophan synthase is a latent tyrosine synthase. *Nat. Chem. Biol.* **20**, 1086–1093 (2024).

55. Buller, A. R. *et al.* Directed evolution of the tryptophan synthase β-subunit for stand-alone function recapitulates allosteric activation. *Proc. Natl. Acad. Sci.* **112**, 14599–14604 (2015).

56. Romney, D. K., Murciano-Calles, J., Wehrmüller, J. E. & Arnold, F. H. Unlocking reactivity of TrpB: A general biocatalytic platform for synthesis of tryptophan analogues. *J. Am. Chem. Soc.* **139**, 10769–10776 (2017).

57. Almhjell, P. J., Boville, C. E. & Arnold, F. H. Engineering enzymes for noncanonical amino acid synthesis. *Chem. Soc. Rev.* **47**, 8980–8997 (2018).

58. Boville, C. E. *et al.* Engineered biosynthesis of β-alkyl tryptophan analogues. *Angew. Chem. Int. Ed.* **57**, 14764–14768 (2018).

59. Brandenberg, O. F., Fasan, R. & Arnold, F. H. Exploiting and engineering hemoproteins for abiological carbene and nitrene transfer reactions. *Curr. Opin. Biotechnol.* **47**, 102–111 (2017).

60. Coelho, P. S., Brustad, E. M., Kannan, A. & Arnold, F. H. Olefin cyclopropanation via carbene transfer catalyzed by engineered cytochrome P450 enzymes. *Science* **339**, 307–310 (2013).

61. Kan, S. B. J., Lewis, R. D., Chen, K. & Arnold, F. H. Directed evolution of cytochrome c for carbon–silicon bond formation: Bringing silicon to life. *Science* **354**, 1048–1051 (2016).

62. Kan, S. B. J., Huang, X., Gumulya, Y., Chen, K. & Arnold, F. H. Genetically programmed chiral organoborane synthesis. *Nature* **552**, 132–136 (2017).

C h a p t e r 2

PROTEIN LANGUAGE MODEL TRANSFER LEARNING

Material from this chapter appears in: "**Li, F.-Z.**, Amini, A. P., Yue, Y., Yang, K. K. & Lu, A. X. Feature reuse and scaling: Understanding transfer learning with protein language models. *In Proceedings of the 41st International Conference on Machine Learning (PMLR),* **235**, 27351–27375 (2024)."

ABSTRACT

Large pretrained protein language models (PLMs) have improved protein property and structure prediction from sequences via transfer learning, in which weights and representations from PLMs are repurposed for downstream tasks. Although PLMs have shown great promise, currently there is little understanding of how the features learned by pretraining relate to and are useful for downstream tasks. We perform a systematic analysis of transfer learning using PLMs, conducting 370 experiments across a comprehensive suite of factors including different downstream tasks, architectures, model sizes, model depths, and pretraining time. We observe that while almost all downstream tasks do benefit from pretrained models compared to naive sequence representations, for the majority of tasks performance does not scale with pretraining, and instead relies on low-level features learned early in pretraining. Our results point to a mismatch between current PLM pretraining paradigms and most applications of these models, indicating a need for better pretraining methods.

2.1 Introduction for Chapter 2

Proteins perform a myriad of critical biological functions, and thus the ability to design proteins has vast impacts on healthcare, environment, and industry.[1] Since a protein's function is largely determined by its amino acid sequence, specifying a sequence that will yield a desired function is feasible in principle. However, the relationship between amino acid sequence and function remains poorly understood, and most experimental methods for measuring function are costly and low-throughput.[2,3] To overcome the challenge presented by limited labeled data, researchers have sought to use transfer learning, in which models are pre-trained in a self-supervised fashion on large public datasets in the hope that the pretrained features or model weights will improve performance on downstream tasks where supervised data is limited (**Figure 2.1a-b**).

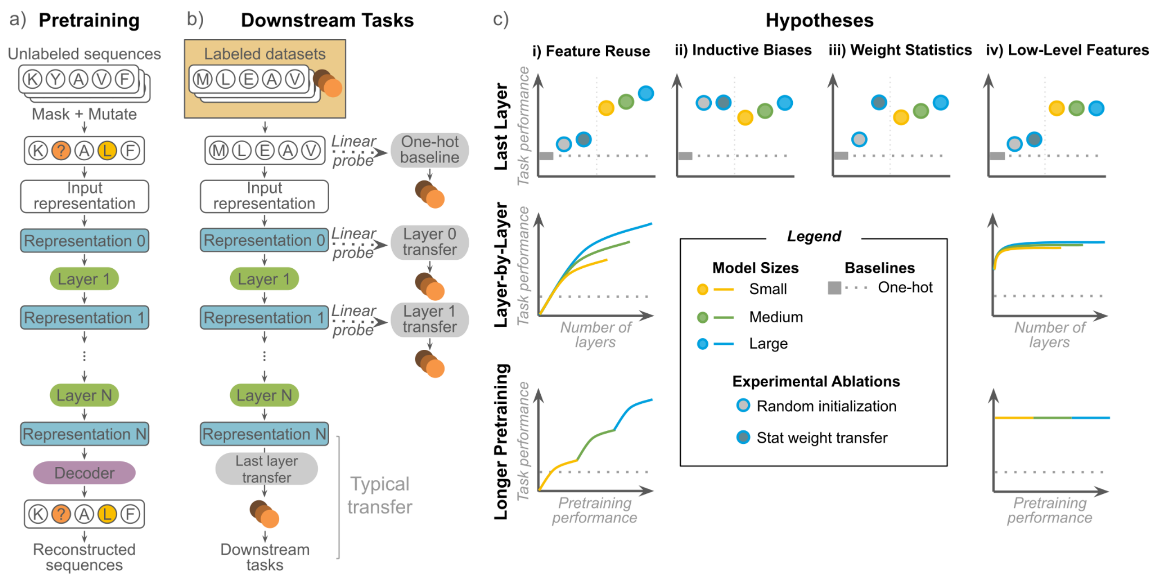


Figure 2.1. *Summary of the transfer learning procedure and our analyses. a) PLMs are pretrained using masked language modeling. b) Typically, transfer learning uses representations from the last layer of the PLM for downstream tasks. We evaluate downstream task performance at every layer in the model. c) We compare to baselines and ablations and evaluate the effects of PLM size, model depth, and pretraining time. These experiments characterize behavior consistent with either feature reuse (i), or an alternative hypothesis (inductive biases/overparameterization - ii, weight statistics - iii, or reuse of low-level features only - iv).*

Protein language models have emerged as the most popular framework for transfer learning for proteins.[4–10] Most PLMs pretrain using the masked language modeling (MLM) task, in which the model is trained to predict the original identity of masked or corrupted amino acids. PLMs have been effective at improving performance on many protein function prediction tasks, and some are now integrated into bioinformatics and structure prediction tools.[11–14] Despite their widespread adoption, it is not understood how or why PLMs improve performance on downstream tasks.

Drawing from other domains like computer vision where investigations of transfer learning are more established, we synthesize a set of possible hypotheses to explain improvement in downstream tasks, and design and conduct a comprehensive series of experiments to test them. We structure our study around the following hypotheses:

Feature reuse (Figure 2.1c-i). One popular hypothesis is that MLM pretraining learns general features of protein biology, and that these features can be reused across tasks. Previous work has shown that transfer learning improves performance across diverse downstream tasks.[15,16] However, the degree of feature reuse is also important: ideally, the pretrain and downstream tasks should be aligned, such that transferring PLM representations improves downstream function prediction accuracy and that this improvement increases with larger model sizes, deeper layers, and better pretraining performance.

If this does not occur, it suggests that pretraining primarily learns features that cannot be reused on downstream tasks. To determine whether or not this is the case for PLMs, we explore three alternative hypotheses.

Inductive biases and overparameterization (Figure 2.1c-ii). The large number of parameters in pretrained models may lead to some alignment with useful signals by chance.[17] If inductive biases are sufficient, then transferring from randomly-initialized versions of the same model architecture should provide similar performance.

Statistics of pretrained weights (Figure 2.1c-iii). The primary benefit of pretraining may be initializing weights to a sensible scale.[17,18] If pretraining primarily provides better weight initialization, resampling weights from the empirical distribution after pretraining should provide similar performance.

Reuse of low-level features (Figure 2.1c-iv). It is possible for only less complex features learned early in pretraining to contribute to transfer learning.[19] If low-level features are sufficient, then features extracted from earlier layers of the pretrained model may provide better or similar performance to those extracted from the last layer. Similarly, earlier pretraining checkpoints or smaller, less performant models should provide similar performance to the full-size, fully-pretrained model.

Critically, while all three alternative hypotheses can still lead to improvements in downstream task performance, they do not predict that downstream task performance can be improved by transferring representations from larger, better trained models.[17,20]

Contributions

Our work evaluates the scalability of transfer learning for PLMs and makes the following contributions:

1. The most comprehensive evaluation, to date and to the best of our knowledge, of transfer learning with PLMs, spanning 370 experiments over a diverse suite of downstream tasks.
2. The discovery that current MLM pretraining paradigms underserve many aspects of protein biology, as supported empirically by evidence from both structure and function prediction tasks
3. Systematic evidence that performance on many protein property prediction tasks does not scale with PLM size or pretraining. Our results uncouple improvements in downstream performance from scaling properties.

Together, our results predict that scaling PLMs with current MLM pretraining paradigms may not scale performance on many protein function prediction tasks, but provides an evaluation framework for identifying if future pretraining efforts are scalable across more aspects of protein biology.

2.2 Related work for Chapter 2

2.2.1 Pretrained protein language models

While numerous pretrained PLMs have been proposed in the past few years,[4–10] these works primarily focus on validating that pretraining improves performance on downstream tasks. In contrast, our work primarily seeks to understand the factors impacting transfer learning, which have not been rigorously studied to date for PLMs. Most PLM studies include comparisons to models with randomly initialized weights[9,10] to confirm that pretrained models do not improve downstream task performance due to overparameterization or inductive biases alone. Other studies show that under some circumstances, PLMs yield no detectable improvement over a simple one-hot representation of sequences.[16,21,22] Compared to these individual baselines and benchmarks, our paper conducts a systematic analysis over many different factors impacting transfer learning.

The most similar work to ours is Detlefsen *et al.* (2022),[23] which analyzes the effects of model architecture, fine-tuning, and different pooling schemes on transfer learning performance. However, we use MLMs trained on complete sequences instead of autoregressive models trained on Pfam domains. While they train proprietary, unreleased models for analysis, we use established models in the public domain. This makes our analysis more relevant to applications currently using these models and also improves documentation around these models. For example, neither their paper nor their released code describes the pretrained models in detail, so it is uncertain what the size of their model is, whereas we systematically vary the model size. More importantly, we evaluate a larger and more diverse set of downstream tasks with experiments designed to differentiate possible mechanisms by which transfer learning improves performance on downstream tasks. Critically, our systematic analysis identifies cases where transfer from PLMs is empirically effective in improving downstream task performance but the improvement is due to factors that are not expected to scale with further pretraining or larger models. However, to the extent that our analyses reach similar conclusions (e.g., both our studies observe that performance on the pretraining task does not always correlate with downstream task performance), we view our work as complementary: Detlefsen *et al.* (2022) use different architectures, pretraining tasks, and pretraining datasets than us, suggesting that our observations are general across more factors than either paper analyzes independently.[23]

2.2.2 Understanding transfer learning in computer vision

While our analysis is differentiated as we focus on protein sequences, we take inspiration from computer vision studies that have sought to understand factors underlying successful transfer learning. Many are motivated by the observation that ImageNet-trained models are effective when transferred to medical images, raising the question of whether transfer performance is really due to reuse of features (given the extreme mismatch in domain), or due to more trivial factors. Raghu *et al.* (2019) compare pretrained models against random initialization to demonstrate that in some situations transfer performance is due to overparameterization.[17] By randomly initializing models to match the weight statistics of pretrained models, the authors further demonstrate that improvements from pretraining may arise from good weight scalings rather than learning reusable features. Similarly, He *et al.* (2019) show that hyperparameter tuning can often explain improvements from transfer learning.[24] By scrambling input images, Neyshabur *et al.* (2021) show that improvements from transfer learning can at least partially be attributed to the pretrained models learning low-level statistics of data rather than more sophisticated feature use.[19] Matsoukas *et al.* (2022) further demonstrate that these factors vary depending upon downstream task dataset and model architecture.[18]

Beyond models pretrained on ImageNet, some papers have looked at factors more specific to self-supervised pretraining. Abnar *et al.* (2022) show that improvements on the self-supervised pretraining task do not necessarily translate to improved performance on downstream tasks, and in some cases, are even anti-correlated.[20] Pioneering work in generative self-supervised models also demonstrates that these models often saturate in downstream task performance in an intermediate layer of the model and degrade after.[25] This is reinforced by empirical studies showing that the representations learned by self-supervised models versus supervised models rapidly diverge in the last few layers,[26] consistent with previous observations that later layers may be more specialized to the original network,[27] underscoring the importance of a layer-by-layer evaluation.

Beyond computer vision, transfer learning has been extensively studied in natural language processing, and particularly in "BERTology",[28] which seeks to understand what self-supervised Transformer models learn and their transferability. Among other factors, studies have similarly analyzed the effects of pretraining,[29] overparameterization,[30] and layer-by-layer content.[31]

2.3 Datasets and pretrained models

To understand why and when transfer learning with PLMs improves downstream performance and how the improvements scale with increasingly large PLMs, we conducted 370 experiments on a diverse suite of downstream tasks with PLMs of different sizes, architectures, and at different checkpoints in training. The downstream tasks are summarized in **Tables 2.1** and **A.1.1**.

Table 2.1. *Summary of downstream prediction tasks.*

Dataset	Description	Tasks	Task type
SS3	Secondary structure	CB513, TS115, CASP12	Residue-level classification
Thermostability	Melting temperature	Thermostability	Regression
Subcellular localization	Cellular location	Subcellular localization	Classification
GB1	Immunoglobulin binding	Sampled, low vs. high, two vs. rest	Regression
AAV	Viral viability	Two vs. many, one vs. many	Regression

2.3.1 Downstream tasks

We test a diverse set of tasks covering both property and structure prediction, different types of distribution shift relevant to protein engineering, and global versus local variation over the sequence.

Structure prediction. We use the three-class secondary structure (SS3) task from TAPE with three independent test sets, SS3 – CB513,[32] SS3 – TS115,[33] and SS3 – CASP12,[34] where the objective is to predict whether each residue belongs to an α-helix, β-strand, or coil.[15]

Property prediction. We use the thermostability, subcellular localization, GB1, and AAV datasets from FLIP.[16]

Thermostability and subcellular localization are global protein properties measured for sequences spanning different functional families and domains of life. The thermostability dataset measures the melting temperature of 48,000 proteins across 13 species.[35] Subcellular localization is a classification task predicting the cell compartment to which a eukaryotic protein localizes.[36,37]

In contrast, the GB1 and AAV datasets measure the effects of local sequence variation. GB1 is the 56 amino-acid B1 domain of protein G, an immunoglobulin-binding protein. The GB1 dataset covers binding measurements for simultaneous mutations of up to 4 interactive sites.[38] VP1 is an adeno-associated virus (AAV) capsid protein, over 700 amino acids long.[39] The AAV dataset measures the effects of sparsely sampled mutations across a contiguous 28 amino-acid region over the binding interface on viral viability. For GB1 and AAV, FLIP provides different train-test splits with different distribution shifts, including sampled (in-distribution) and out-of-distribution splits, as described in **Table A.1.1**. Out-of-distribution splits more closely resemble protein engineering applications where a few low-functioning variants with a limited number of mutations are initially generated, but high-functioning variants across the larger sequence space are the engineering end goal. For GB1, we test three splits, in order of increasing difficulty:

- **Sampled:** sequences randomly partitioned between 80% training and 20% testing.
- **Low vs high:** models are trained on mutants with function worse than the parent and tested on those with better function.
- **Two vs rest:** Models are trained on single and double mutants and tested on triple and quadruple mutants.

For AAV, we test two splits, in order of increasing difficulty:

- **Two vs many:** Models are trained on single and double mutants and tested on variants with three or more mutations.
- **One vs many:** Models are trained on single mutants and tested on variants with more mutations.

2.3.2 Transfer learning with protein language models

While a number of pretraining tasks have been proposed for protein sequences, we focused on models trained using the popular BERT[40] masked language modeling (MLM) task. During pretraining, 15% of tokens are randomly selected. Of the 15%, 10% are replaced with a special masking token, 2.5% are randomly changed to another token, and the remaining 2.5% are unperturbed to encourage the model to preserve the input sequence. The corrupted sequence is passed to the model, which is trained to maximize the probability of the original tokens at the selected locations.

To evaluate the effect of model architecture, we chose two families of protein MLMs with comparable model sizes trained on UniRef50:[41] the ESM[9] family of transformer models and the Convolutional Autoencoding Representations of Proteins (CARP)[10] family of convolutional models. Due to the sequence length limit of the ESM-1b transformer model, the first and last 511 amino acids were taken for all sequences exceeding 1022 amino acids. This length restriction chiefly impacts the subcellular localization dataset: targeting signals often occur at the N- or C- terminal, and we reason that taking both terminals preserves biologically-relevant signals.

Following standard protein transfer learning practice when resources for full finetuning are not available,[16] we pass representations from each PLM layer to a linear model and compare the performance to a linear model on the one-hot encoding of the sequence for each task (**Figure 2.1b**). In addition to linear models, we also tested a learned attention pooling followed by a shallow MLP. However, we found last layer performance to be inferior to the linear models across almost all downstream tasks (**Figure A.1.1**), so we focus on the linear models in our analyses. For the SS3 and subcellular localization tasks, we train linear classifiers with mini-batches in PyTorch and perform early stopping based on the validation set. For the regression tasks, we train ridge regression models with Scikit-learn,[42] using a grid search on the validation set to tune the regularization strength. For all tasks except secondary structure prediction, we mean pool the representations over the length dimension from each layer. Secondary structure prediction requires a representation for every residue, so no pooling is performed.

As protein engineers often seek to identify top-ranked mutants as opposed to predicting the absolute function of mutations, we use ranking metrics, Spearman's rank correlation and Normalized Discounted Cumulative Gain (NDCG), as the primary metrics for the regression tasks. For concision, we report Spearman's rank correlation for regression tasks and accuracy for classification tasks in the main text. Complete results, including mean square error, cross-entropy loss, NDCG, and ROC-AUC are provided in the *online Supplemental Materials*.

2.4 Experimental setup

Baseline and ablations. We conduct baselines and model ablations to determine when transfer learning improves downstream task performance and whether improvements in downstream task performance can be attributed to mechanisms other than feature reuse (**Figure 2.1b-ii** and **2.1b-iii**).

- **One-hot baseline.** To determine whether transfer learning with PLMs improves performance, we test if representations from pretrained models perform better than a one-hot representation.

- **Random init.** To evaluate whether the effect of transfer learning is due to overparameterization and/or the inductive biases of the PLM architecture, we test the impact of randomly initialized weights.
- **Stat transfer.** To evaluate whether the effect of transfer learning is due to weight statistics and/or initializing the weights to a sensible scale, we test the impact of randomly initialized weights matching the weight distribution of the pretrained PLM by randomly permuting the pretrained weights.

For random init and stat transfer, we initialize models with 3 random seeds. We consider transfer learning from a PLM to improve performance over these baselines if it has a one tailed p-value < 0.05 in a one-sample t-test using the sample mean and standard deviation across random init/stat transfer models.

Scaling experiments. To further understand if the MLM pretraining task is aligned with downstream tasks, we sought to understand if improving PLM performance by scaling across three factors also improves transfer learning performance on downstream tasks (**Figure 2.1b-iv**):

- **Model size.** For both CARP and ESM, we test models with different numbers of layers and parameters (**Table A.1.2**). For concision, we refer to CARP-38M and ESM-43M as the "small" models, CARP-76M and ESM-85M as the "medium" models, and CARP-640M and ESM-650M (ESM-1b) as the "large" models.
- **Model depth.** For each architecture (CARP, and ESM) and model size, we test whether downstream task performance improves as we transfer deeper layers by determining whether the Spearman rank correlation between layer number and performance is greater than 0.9 (**Table A.1.6**). This experiment allows us to understand if tasks primarily reuse low-level features early in the pretrained models, or if more complex features deeper in the models also contribute to downstream task performance. Convolutional neural networks (CNNs) induce a stronger correlation between the depth of the layer and the complexity of the features than transformers, leading to different patterns of feature reuse in previous transfer learning studies.[18]

However, we find little empirical difference between CNNs (CARP) and transformers (ESM) in our analyses.

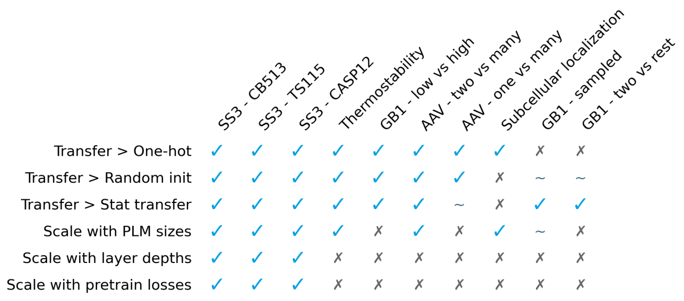
- **Model checkpoint.** For each model size, we test the effect of using checkpoints from earlier in pretraining. We order these checkpoints based upon their pretraining performance (perplexity, calculated on a held-out test set of 210k sequences from Uniref50 not used to train CARP), as earlier checkpoints have higher losses on the MLM pretraining task (**Table A.1.7**). We evaluate whether features from later in pretraining improve transfer learning by determining whether the Spearman rank correlation between the negative pretrain loss and downstream performance is greater than 0.9 (**Table A.1.8**). Unfortunately, checkpoints are only publicly available for CARP, so we cannot run this analysis with ESM.

Estimating error for our scaling experiments is infeasible as it would require re-training multiple PLMs from scratch. Thus, we chose arbitrary thresholds, which may impact the way we have categorized downstream tasks in our interpretation. For transparency, we have plotted performance for all PLMs in each of our scaling experiments in **Figures 2.3**, **2.4**, and **2.5**.

We define the MLM pretraining task to be *aligned* with a downstream task if transferring PLM representations improves downstream task performance over the baseline and ablations and this improvement scales with improvements to pretraining. Code for all experiments is available at *https://github.com/microsoft/protein-transfer*.

2.5 Results

Overall, our analyses reveal three clusters of transfer learning behavior across downstream tasks (**Figure 2.2**). First, we find that within our set of benchmarks, secondary structure prediction tasks are the only tasks where pretraining improves downstream performance and the pretrain and downstream tasks are aligned. Second, we observe that transfer learning improves performance for many downstream tasks despite the pretrain and downstream tasks not being well-aligned, indicating that performance on these tasks may not improve as PLMs scale on the axes we tested. Third, we observe that although transfer learning improves performance on almost all downstream tasks, for some tasks this improvement can be attributed to overparameterization, inductive biases, or sensible weight initialization. In subsequent sections, we expand on each of these clusters of observations in detail. The full results of our experiments are available in the *online Supplemental Materials*.



	SS3 - CB513	SS3 - TS115	SS3 - CASP12	Thermostability	GB1 - low vs high	AAV - two vs many	AAV - one vs many	Subcellular localization	GB1 - sampled	GB1 - two vs rest
Transfer > One-hot	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
Transfer > Random init	✓	✓	✓	✓	✓	✓	✓	✗	~	~
Transfer > Stat transfer	✓	✓	✓	✓	✓	✓	~	✗	✓	✓
Scale with PLM sizes	✓	✓	✓	✓	✗	✓	✗	✓	~	✗
Scale with layer depths	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
Scale with pretrain losses	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗

Figure 2.2. *Downstream task result summary. ✓ indicates true, ~ indicates true for only one architecture, and ✗ indicates false.*

2.5.1 Structure prediction benefits from transfer learning because it is well-aligned with MLM pretraining

For all three residue-level secondary structure prediction tasks, **Figure 2.3a** and **Table A.1.3** show that PLM embeddings outperform the one-hot baseline as well as the random init and stat transfer ablations, demonstrating that transfer learning improves secondary structure prediction performance and that the improvement is not due to the inductive biases or weight statistics of the models. Secondary structure prediction performance improves when transferring deeper PLM features (**Figure 2.3b**), indicating that more complex features from later layers continue to improve performance. Furthermore, transfer learning with features from larger models and from later in pretraining improve secondary structure prediction (**Figure 2.3a** and **2.3c**), as previously observed by Rives *et al.* (2021),[9] Elnaggar *et al.* (2022),[6] and Yang *et al.* (2024).[10] We therefore conclude that MLM pretraining is well-aligned to structure prediction, allowing PLM features to be reused when predicting secondary structure from sequence.

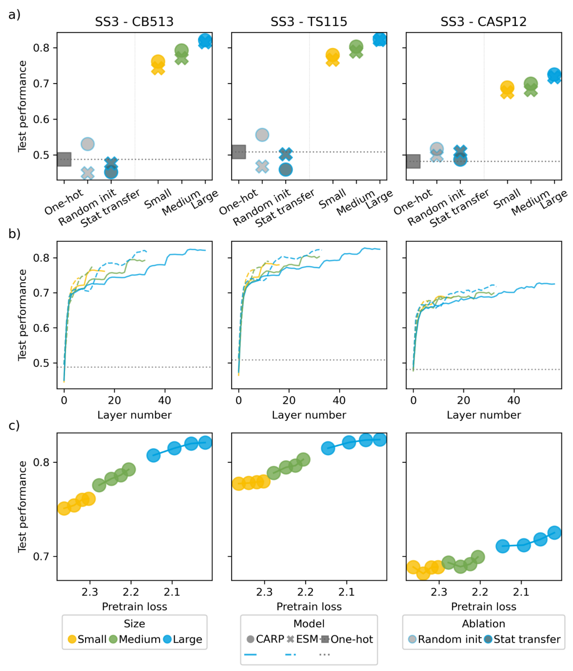


Figure 2.3. *Results for secondary structure prediction. a) Performance on downstream tasks when transferring the final layer representation from various sizes of ESM and CARP compared to baselines and ablations. b) Downstream task performance by depth of layer transferred. c) Downstream task performance by pretraining loss. Each dot is a model checkpoint. For all subplots, downstream task test performance is quantified using accuracy.*

2.5.2 Many tasks benefit from transfer learning despite lack of alignment with MLM pretraining

Next, we observe a cluster of four downstream tasks (thermostability, GB1 – low vs high, AAV – two vs many, and AAV – one vs many) where transfer learning improves performance over baselines even though the tasks do not align well with the pretraining task (**Figure 2.4**). For these tasks, transfer learning improves performance over both the random init and stat transfer ablations, indicating that transfer learning confers at least some benefit over the inductive biases, parameterization, or weight statistics of the models alone (with the exception of the AAV – one vs many task, where weight statistics may still explain transfer learning performance) (**Figure 2.4a** and **Table A.1.4**). However, for all of these tasks, downstream task performance does not improve as features from deeper layers are transferred (**Figure 2.4b**) or as the PLMs improve their pretraining loss over checkpoints

(**Figure 2.4c**), suggesting that these tasks may rely upon low-level features learned early in pretraining. Notably, the performance on these tasks typically saturates below fully fine-tuned models of the same architecture (and always below the best-performing model for each task) trained by Yang *et al.* (2022) (**Table A.1.4**),[10] indicating that the saturation is not because downstream task performance has hit an upper bound.

To supplement our quantitative cut-offs for alignment, we qualitatively assess trends in layer-by-layer performance across tasks. We observe that for all tasks where transfer learning improves performance over the baselines (including the secondary structure prediction tasks), the largest gains in performance occur in the first 3-5 layers of both the ESM and CARP models, across model sizes (**Figure 2.3b** and **2.4b**). However, unlike the secondary structure prediction tasks, which continue to improve in performance past this initial peak, improvement on the downstream tasks in this cluster generally plateaus (e.g., for the AAV – two vs many task), supporting our interpretation that features contributing to these tasks are already present within the first few layers of pretrained PLMs.

Interestingly, although none of these tasks scale with model depth or pretraining loss, two downstream tasks (thermostability and AAV – two vs many) scale with PLM size (**Figure 2.4a**). We reasoned that while our random init ablation rules out that improvements in downstream task performance is entirely due to parameterization, parameterization may still partially contribute to performance independently of feature reuse. To test this, we additionally evaluated the performance of small and medium randomly initialized models. Indeed, we observe that both types of randomly initialized models scale in performance with model size for both tasks, and in similar proportions to the improvements for the pretrained models (**Table A.1.4**). Together, this suggests that observing that downstream task performance scales with model size alone is not sufficient to conclude that pretraining and downstream tasks are aligned, and that demonstrating scaling across other axes (such as model depth and checkpoints in training, as we propose here) is necessary.

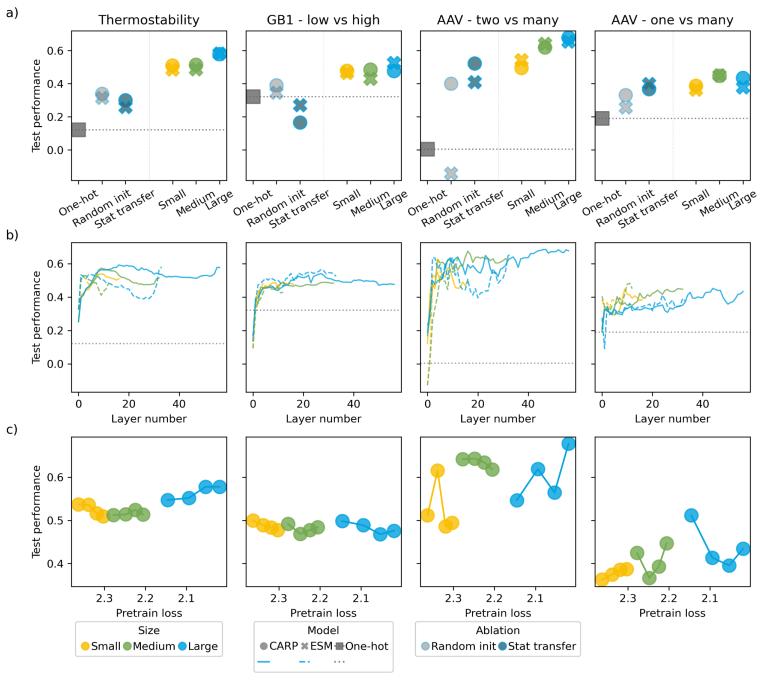


Figure 2.4. *Results for tasks where transfer learning improves downstream task performance, but the pretrain and downstream tasks are not aligned. a) Performance on downstream tasks when transferring the final layer representation from various sizes of ESM and CARP compared to baselines and ablations. b) Downstream task performance by depth of layer transferred. c) Downstream task performance by pretraining loss. Each dot is a model checkpoint. For all subplots, downstream task test performance is quantified using Spearman's rank correlation.*

2.5.3 Some tasks do not benefit from MLM pretraining

Finally, we observe a cluster of three downstream tasks (subcellular localization, GB1 – sampled, and GB1 – two vs rest) where pretraining does not improve transfer learning performance (**Figure 2.5**). For subcellular localization, although transfer learning improves over a one-hot representation, pretrained models perform no better than randomly initialized models, suggesting that the improvement can be entirely attributed to inductive biases and parameterization. In contrast, the GB1 tasks in this cluster fail to outperform a one-hot representation by at least 10% (**Figure 2.5a** and **Table A.1.5**). We hypothesize the GB1 splits in this cluster are either too trivial, or too challenging for any representation. GB1 – sampled is an in-distribution task with a relatively large training set, and all models and baselines perform well. Meanwhile, GB1 – two vs rest is a challenging out-of-distribution split. Intriguingly, our stats transfer ablation decreases performance for all GB1 tasks, including the GB1 – low vs high task in the previous section, compared to the one-hot and random initialization baselines (**Figure 2.4b** and **2.5b**; **Tables A.1.4** and **A.1.5**). We hypothesize that this is because the GB1 dataset is a highly local task, depending on finding interactions between just four mutated positions in a sequence.

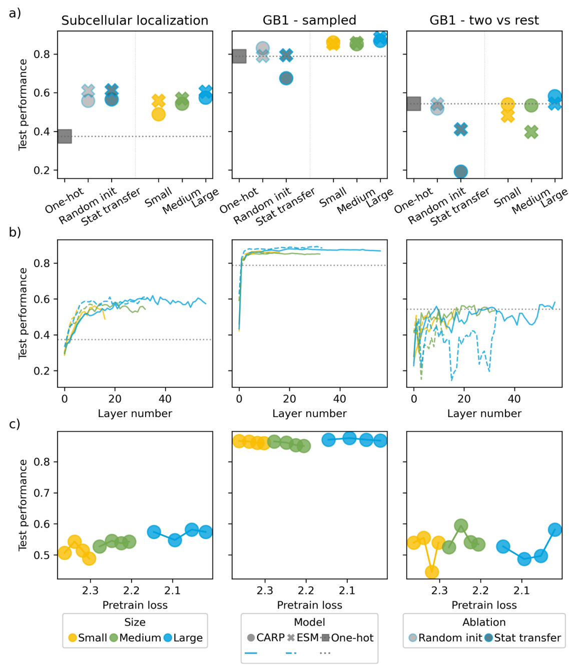


Figure 2.5. *Results for tasks where pretraining does not improve downstream task performance. a) Performance on downstream tasks when transferring the final layer representation from various sizes of ESM and CARP compared to baselines and ablations. b) Downstream task performance by depth of layer transferred. c) Downstream task performance by pretraining loss. Each dot is a model checkpoint. For subcellular localization, the downstream classification task performance is quantified using accuracy. For other tasks, the downstream regression task performance is quantified using Spearman's rank correlation.*

2.6 Discussion

In this work, we systematically evaluate the mechanisms via which transfer learning from large pretrained protein language models improve performance on downstream protein function and structure prediction tasks. While most downstream tasks benefit from transfer learning, of the tasks we evaluated, structure prediction is the only task where we observe pretrain-downstream alignment. Our results are consistent with previous studies that show MLM pretraining imparts information about protein structure. Previous work has shown that the attention matrices in pretrained PLMs recapitulate contact maps,[43,44] that it is possible to extract contact maps by perturbing the inputs to PLMs,[45] and that PLM representations contain similar co-evolution information as multiple sequence alignments.[8,14,46]

Other works have argued that larger models will not benefit fitness prediction when using zero-shot likelihoods from generative models,[47,48] as some degree of misspecification is important for generalizing from natural protein distributions to mutant variants. Our results, which show that fitness prediction tasks do not scale with pretraining, are consistent with these prior works, but we show this holds true even when transferring embeddings and even with some degree of fine-tuning. At the same time, by showing that structural prediction tasks do scale with pretraining, our results suggest that these prior results may not be general past fitness prediction; one possibility is that fitness prediction usually involves mutant variants not seen in the databases of natural proteins used for pretraining, whereas the structural prediction tasks benchmarked in our paper classify secondary structural elements that will be seen in natural proteins.

Our primary contribution is showing that scaling pretraining does not improve performance on prediction tasks that are less reliant on coevolutionary patterns, and that outperforming the one-hot and randomly initialized baselines does not imply that downstream task performance will scale with pretraining performance. While our observation that PLMs fail to scale on many downstream tasks may not generalize as our evaluation framework is extended over different pretraining tasks, architectures, datasets, and fine-tuning methods, by showing that performance over baselines is not always coupled with scaling, we provide a means for future models to improve on these limitations.

Limitations. There are factors known to impact transfer that we could not test for PLMs due to a lack of public models or to computational expense. First, pretraining dataset is important, both in terms of distance between the pretraining and downstream task data domains[49] and data size.[20] PLMs pretrain on large databases of natural sequences. In principle, this means that some downstream tasks may be out-of-distribution (e.g., those involving artificial variation or non-natural function), or subject to biases in data collection (e.g., taxonomies less-represented in UniProt).[50] Previous studies have shown differences in pretraining performance by taxonomy,[51] and that model likelihoods are biased towards more frequent species in UniProt.[52] However, Meier *et al.* (2021)[53] trained versions of ESM on UniRef100 instead of UniRef50, and Dallago *et al.* (2021)[16] show that they perform very similarly on function prediction tasks. Moreover, subsampling pretraining sequence datasets has not been explored beyond down-sampling redundant sequences, making the impact of data difficult to evaluate for pretrained PLMs.

Second, a variety of other pretraining tasks have been proposed for protein transfer learning, such as autoregressive next-token prediction.[54–56] Different pretraining tasks could potentially learn different aspects of protein biology, and thus have different patterns of scaling. While we only evaluated the MLM pretraining objective, future work that tests other pretraining tasks under our evaluation framework will be critical. However, from principle, we remain uncertain if existing tasks in literature will result in significant differences from MLMs. Many pretraining tasks still aim to reconstruct natural sequences[57–60] and so are also likely to primarily learn coevolutionary patterns. Other tasks use structure as an additional input or target, but they generally make only modest improvements on function prediction tasks.[61–64] Supporting the assertion that learning to predict structure may not improve function prediction, Hu *et al.* (2022)[65] show that transfer learning using the AlphaFold2[66] structure module is less effective for function prediction than transferring PLMs. Finally, Brandes *et al.* (2022)[5] and Xu *et al.* (2023)[67] reconstruct both sequence and functional annotations but also find that downstream performance does not always scale with pretraining time.

Finally, we only test probes on frozen models to limit computational cost, but previous work shows that for many tasks finetuning the PLM end-to-end outperforms a linear probe or training a small neural network on top of the frozen pre-trained weights,[10,16] and that mean-pooling is rarely optimal.[23,68] In computer vision, models trained on different datasets[49] and pretraining tasks[26] exhibit different finetuning dynamics, and there is some evidence for this in proteins as well.[23] More sophisticated approaches to finetuning (such as using automated

ML to select architecture of probe) could further improve performance, and introduce different patterns of alignment.

Besides studying further factors that may impact transfer learning, improvements to our evaluation could better understand details of mechanism. Transformer PLMs often learn sparse attention matrices,[43,44] so one question is if it is the sparsity that drives performance, or if pairwise attention must be placed on the correct pairs of residues (as opposed to any pairs). However, our current baselines do not permit this understanding: both random init and stat transfer do not guarantee that sparsity in attention matrices is preserved.

Finally, although we require that downstream task performance scale with improvements on the pretraining task and specifically analyze three axes of scaling (model size, depth of model, and checkpoint in pretraining), it is unclear if all three axes are necessary. For example, effective self-supervised learning tasks in computer vision often diverge in their final layer representations relative to supervised models,[26] suggesting that downstream task performance may not always improve monotonically with layer depth, even when pretraining is effective. However, in our specific context, we chose to include scaling with layer depth because we thought it to be an explanation of potential mechanism for why scaling other axes does not translate into downstream task performance. We observed that even though larger models and models pretrained for longer generally achieve better performance on the pretraining task, this often does not translate into downstream task performance. Our layer depth experiments show that downstream task performance often saturates alarmingly early: for the tasks where pretraining improves downstream task performance, but this improvement does not scale, downstream task performance usually saturates within the first 3-5 layers of models (**Figure 2.4**), even in models with 30+ layers. This observation suggests that PLMs are currently burning the majority of their capacity parameter-wise on modeling the pretraining task, with very few of the learned features contributing to both the pretraining and downstream tasks jointly. However, our rationale for why we included layer depth as an axis of scaling contributing to "alignment" is dependent on this context, so ultimately, our definition of "alignment" is oversimplified, and future work should further analyze the interplay and interaction between different axes of scaling.

Implications for future work. Together, the number of factors that can potentially impact transfer learning means that there are many opportunities for future work to address the limitations in scaling that we identified in our work. Towards this goal, our work provides an improved evaluation standard for PLMs. We show that checking for improved performance over baselines may overestimate the generality of PLMs across applications in protein biology, as it does not rule out that improvement may be due to alternate hypotheses that do not scale. However, most current works rely on comparisons to baselines to argue that PLMs are widely applicable, and to the extent scaling has been studied, most only use scaling on structure prediction accuracy alone to justify training larger models.[6,8,9,69] Future PLM evaluation should therefore assess scaling on diverse downstream function prediction and engineering tasks, and not just structure alone, to validate the generality of models.

Second, synthesizing our empirical results with how the current landscape of protein sequence pretraining tasks primarily align with structure prediction, our work points to a need for new pretraining tasks. For many downstream tasks, the lack of alignment prevents transfer learning from taking full advantage of the pretrained model, as features from deep in the PLM perform no better than features from early layers in the PLM. Likewise, for these tasks, simply scaling to larger PLMs trained for more steps on more data may not improve performance. Our study suggests that the field needs to explore diversified pretraining strategies instead of further scaling existing strategies in order to reach aspects of protein biology that are not currently well-served by PLMs.

2.7 Bibliography for Chapter 2

1. Lutz, S. & Iamurri, S. M. Protein engineering: Past, present, and future. *Methods Mol. Biol. Clifton NJ* **1685**, 1–12 (2018).
2. Maynard Smith, J. Natural selection and the concept of a protein space. *Nature* **225**, 563–564 (1970).
3. Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat. Rev. Mol. Cell Biol.* **10**, 866–876 (2009).
4. Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods* **16**, 1315–1322 (2019).
5. Brandes, N., Ofer, D., Peleg, Y., Rappoport, N. & Linial, M. ProteinBERT: A universal deep-learning model of protein sequence and function. *Bioinformatics* **38**, 2102–2110 (2022).
6. Elnaggar, A. *et al.* ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
7. Elnaggar, A. *et al.* Ankh: Optimized protein language model unlocks general-purpose modelling. Preprint at https://doi.org/10.48550/arXiv.2301.06568 (2023).
8. Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
9. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci.* **118**, (2021).
10. Yang, K. K., Fusi, N. & Lu, A. X. Convolutions are competitive with transformers for protein sequence pretraining. *Cell Syst.* (2024) doi:10.1016/j.cels.2024.01.008.
11. Flamholz, Z. N., Biller, S. J. & Kelly, L. Large language models improve annotation of prokaryotic viral proteins. *Nat. Microbiol.* 1–13 (2024) doi:10.1038/s41564-023-01584-8.
12. Teufel, F. *et al.* SignalP 6.0 predicts all five types of signal peptides using protein language models. *Nat. Biotechnol.* **40**, 1023–1025 (2022).

13. Thumuluri, V., Almagro Armenteros, J. J., Johansen, A. R., Nielsen, H. & Winther, O. DeepLoc 2.0: Multi-label subcellular localization prediction using protein language models. *Nucleic Acids Res.* **50**, W228–W234 (2022).
14. Wu, R. *et al.* High-resolution de novo structure prediction from primary sequence. 2022.07.21.500999 Preprint at https://doi.org/10.1101/2022.07.21.500999 (2022).
15. Rao, R. *et al.* Evaluating protein transfer learning with TAPE. *ArXiv190608230 Cs Q-Bio Stat* (2019).
16. Dallago, C. *et al.* FLIP: Benchmark tasks in fitness landscape inference for proteins. (2021).
17. Raghu, M., Zhang, C., Kleinberg, J. & Bengio, S. Transfusion: Understanding transfer learning for medical imaging. Preprint at http://arxiv.org/abs/1902.07208 (2019).
18. Matsoukas, C., Haslum, J. F., Sorkhei, M., Söderberg, M. & Smith, K. What Makes Transfer Learning Work For Medical Images: Feature Reuse & Other Factors. Preprint at http://arxiv.org/abs/2203.01825 (2022) doi:10.48550/arXiv.2203.01825.
19. Neyshabur, B., Sedghi, H. & Zhang, C. What is being transferred in transfer learning? Preprint at http://arxiv.org/abs/2008.11687 (2021).
20. Abnar, S., Dehghani, M., Neyshabur, B. & Sedghi, H. Exploring the limits of large scale pre-training. Preprint at https://arxiv.org/abs/2110.02095 (2022).
21. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026-1045.e7 (2021).
22. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* 1–9 (2022) doi:10.1038/s41587-021-01146-5.
23. Detlefsen, N. S., Hauberg, S. & Boomsma, W. Learning meaningful representations of protein sequences. *Nat. Commun.* **13**, 1914 (2022).
24. He, K., Girshick, R. & Dollar, P. Rethinking ImageNet pre-training. in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* 4918–4927 (2019).
25. Jing, L. & Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**, 4037–4058 (2020).

26. Grigg, T. G., Busbridge, D., Ramapuram, J. & Webb, R. Do self-supervised and supervised methods learn similar visual representations? Preprint at http://arxiv.org/abs/2110.00528 (2021).

27. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. How transferable are features in deep neural networks? Preprint at https://doi.org/10.48550/arXiv.1411.1792 (2014).

28. Rogers, A., Kovaleva, O. & Rumshisky, A. A primer in BERTology: What we know about how BERT works. *Trans. Assoc. Comput. Linguist.* **8**, 842–866 (2021).

29. Kovaleva, O., Romanov, A., Rogers, A. & Rumshisky, A. Revealing the dark secrets of BERT. Preprint at https://doi.org/10.48550/arXiv.1908.08593 (2019).

30. Gordon, M. A., Duh, K. & Andrews, N. Compressing BERT: Studying the effects of weight pruning on transfer learning. Preprint at https://doi.org/10.48550/arXiv.2002.08307 (2020).

31. Lin, Y., Tan, Y. C. & Frank, R. Open sesame: Getting inside BERT's linguistic knowledge. Preprint at https://doi.org/10.48550/arXiv.1906.01698 (2019).

32. Cuff, J. A. & Barton, G. J. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins Struct. Funct. Bioinforma.* **34**, 508–519 (1999).

33. Yang, Y. *et al.* Sixty-five years of the long march in protein secondary structure prediction: The final stretch? *Brief. Bioinform.* **19**, 482–494 (2016).

34. Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T. & Tramontano, A. Critical assessment of methods of protein structure prediction (CASP) – round XII. *Proteins* **86**, 7–15 (2018).

35. Jarzab, A. *et al.* Meltome atlas—Thermal proteome stability across the tree of life. *Nat. Methods* **17**, 495–503 (2020).

36. Almagro Armenteros, J. J., Sønderby, C. K., Sønderby, S. K., Nielsen, H. & Winther, O. DeepLoc: Prediction of protein subcellular localization using deep learning. *Bioinformatics* **33**, 3387–3395 (2017).

37. Stärk, H., Dallago, C., Heinzinger, M. & Rost, B. Light attention predicts protein location from the language of life. *Bioinforma. Adv.* **1**, vbab035 (2021).

38. Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife* **5**, e16965 (2016).

39. Bryant, D. H. *et al.* Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39**, 691–696 (2021).

40. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. Preprint at https://doi.org/10.48550/arXiv.1810.04805 (2019).

41. Suzek, B. E. *et al.* UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31**, 926–932 (2015).

42. Buitinck, L. *et al.* API design for machine learning software: Experiences from the scikit-learn project. Preprint at https://doi.org/10.48550/arXiv.1309.0238 (2013).

43. Vig, J. *et al.* BERTology meets biology: Interpreting attention in protein language models. Preprint at https://arxiv.org/abs/2006.15222 (2020).

44. Rao, R., Meier, J., Sercu, T., Ovchinnikov, S. & Rives, A. Transformer protein language models are unsupervised structure learners. *bioRxiv* 2020.12.15.422761 (2021) doi:10.1101/2020.12.15.422761.

45. Zhang, Z. *et al.* Protein language models learn evolutionary statistics of interacting sequence motifs. 2024.01.30.577970 Preprint at https://doi.org/10.1101/2024.01.30.577970 (2024).

46. Chowdhury, R. *et al.* Single-sequence protein structure prediction using a language model and deep learning. *Nat. Biotechnol.* **40**, 1617–1623 (2022).

47. Nijkamp, E., Ruffolo, J., Weinstein, E. N., Naik, N. & Madani, A. ProGen2: Exploring the boundaries of protein language models. Preprint at https://arxiv.org/abs/2206.13517 (2022).

48. Weinstein, E. N., Amin, A. N., Frazer, J. & Marks, D. S. Non-identifiability and the blessings of misspecification in models of molecular fitness. *bioRxiv* 2022.01.29.478324 Preprint at https://doi.org/10.1101/2022.01.29.478324 (2023).

49. Cherti, M. & Jitsev, J. Effect of pre-training scale on intra- and inter-domain, full and few-shot transfer learning for natural and X-Ray chest images. in *2022 International Joint Conference on Neural Networks (IJCNN)* 1–9 (2022). doi:10.1109/IJCNN55064.2022.9892393.

50. The UniProt Consortium. UniProt: A worldwide hub of protein knowledge. *Nucleic Acids Res.* **47**, D506–D515 (2019).

51. Armenteros, J. J. A., Johansen, A. R., Winther, O. & Nielsen, H. Language modelling for biological sequences – Curated datasets and baselines. 2020.03.09.983585 Preprint at https://doi.org/10.1101/2020.03.09.983585 (2020).

52. Ding, F. & Steinhardt, J. Protein language models are biased by unequal sequence sampling across the tree of life. 2024.03.07.584001 Preprint at https://doi.org/10.1101/2024.03.07.584001 (2024).

53. Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv* 2021.07.09.450648 (2021) doi:10.1101/2021.07.09.450648.

54. Madani, A. *et al.* Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.* **41**, 1099–1106 (2023).

55. Ferruz, N., Schmidt, S. & Höcker, B. ProtGPT2 is a deep unsupervised language model for protein design. *Nat. Commun.* **13**, 4348 (2022).

56. Hesslow, D., Zanichelli, N., Notin, P., Poli, I. & Marks, D. RITA: A study on scaling up generative protein sequence models. Preprint at http://arxiv.org/abs/2205.05789 (2022) doi:10.48550/arXiv.2205.05789.

57. He, L. *et al.* Pre-training co-evolutionary protein representation via a pairwise masked language model. *ArXiv211015527 Cs* (2021).

58. Notin, P. *et al.* Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. in *Proceedings of the 39th International Conference on Machine Learning* 16990–17017 (PMLR, 2022).

59. Tan, Y. *et al.* PETA: Evaluating the impact of protein transfer learning with sub-word tokenization on downstream applications. Preprint at https://doi.org/10.48550/arXiv.2310.17415 (2023).

60. Ma, C. *et al.* Retrieved sequence augmentation for protein representation learning. 2023.02.22.529597 Preprint at https://doi.org/10.1101/2023.02.22.529597 (2023).

61. Mansoor, S., Baek, M., Madan, U. & Horvitz, E. Toward more general embeddings for protein design: Harnessing joint representations of sequence and structure. *bioRxiv* 2021.09.01.458592 (2021) doi:10.1101/2021.09.01.458592.

62. Wang, Z. *et al.* LM-GVP: An extensible sequence and structure informed deep learning framework for protein property prediction. *Sci. Rep.* **12**, 6832 (2022).

63. Yang, K. K., Zanichelli, N. & Yeh, H. Masked inverse folding with sequence transfer for protein representation learning. *Protein Eng. Des. Sel.* **36**, gzad015 (2023).

64. Su, J. *et al.* SaProt: Protein language modeling with structure-aware vocabulary. 2023.10.01.560349 Preprint at https://doi.org/10.1101/2023.10.01.560349 (2024).

65. Hu, M. *et al.* Exploring evolution-aware & -free protein language models as protein function predictors. *Adv. Neural Inf. Process. Syst.* **35**, 38873–38884 (2022).

66. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).

67. Xu, M., Yuan, X., Miret, S. & Tang, J. ProtST: Multi-modality learning of protein sequences and biomedical texts. in *Proceedings of the 40th International Conference on Machine Learning* 38749–38767 (PMLR, 2023).

68. Goldman, S., Das, R., Yang, K. K. & Coley, C. W. Machine learning modeling of family wide enzyme-substrate specificity screens. *PLOS Comput. Biol.* **18**, e1009853 (2022).

69. Chen, B. *et al.* xTrimoPGLM: Unified 100B-scale pre-trained transformer for deciphering the language of protein. Preprint at https://doi.org/10.48550/arXiv.2401.06199 (2024).

Chapter 3

MACHINE LEARNING-ASSISTED DIRECTED EVOLUTION ACROSS DIVERSE COMBINATORIAL LANDSCAPES

Material from this chapter appears in: "**Li, F.-Z.**, Yang, J., Johnston, K. E., Gürsoy, E., Yue, Y. & Arnold, F. H. Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes. *bioRxiv* (2024). doi: 10.1101/2024.10.24.619774."

ABSTRACT

Various machine learning-assisted directed evolution (MLDE) strategies have been shown to identify high-fitness protein variants more efficiently than typical directed evolution approaches. However, limited understanding of the factors influencing MLDE performance across diverse proteins has hindered optimal strategy selection for wet-lab campaigns. To address this, we systematically analyzed multiple MLDE strategies, including active learning and focused training using six distinct zero-shot predictors, across 16 diverse protein fitness landscapes. By quantifying landscape navigability with six attributes, we found that MLDE offers a greater advantage on landscapes which are more challenging for directed evolution, especially when focused training is combined with active learning. Despite varying levels of advantage across landscapes, focused training with zero-shot predictors leveraging distinct evolutionary, structural, and stability knowledge sources consistently outperforms random sampling for both binding interactions and enzyme activities. Our findings provide practical guidelines for selecting MLDE strategies for protein engineering.

3.1 Introduction for Chapter 3

Engineered proteins are indispensable across myriad applications, serving as effective therapeutics to combat diseases, non-toxic agents to enhance crops, and green biocatalysts to synthesize chemicals.[1] The development of such useful proteins often involves directed evolution (DE), a method for accumulating beneficial mutations using iterations of mutagenesis and functional assessment by selection or screening.[2–4] DE is an empirical, greedy hill climbing process on a high-dimensional fitness landscape that maps protein sequence to function (**Figure 3.1a**).[5,6] Despite its widespread use, DE remains time-consuming and resource-intensive: screening is expensive, and multiple rounds of mutation and screening may be needed to generate the desired improvements.

Fitness landscapes are more rugged and difficult to traverse when they are rich in epistatic, or non-additive, effects of amino acid substitutions.[7,8] Epistasis is often observed between mutations in close structural proximity[9] and is enriched at binding surfaces or enzyme active sites, due to direct interactions between residues, substrates, and/or cofactors.[8] Protein engineers frequently target mutations to these interacting sites to enhance a function, often using simultaneous site-saturation mutagenesis (SSM) to make libraries in which the targeted amino acids are mutated to many or all 19 other possible amino acids.[10] Combining the beneficial mutations found at these sites often reveals epistatic effects. For example, beneficial mutations in the context of the initial sequence may not be beneficial in combination with other mutations. Therefore, epistasis can present a significant challenge for DE.[3]

Compared to DE, machine learning-assisted DE (MLDE) has shown promise for exploring a broader scope of sequence space and more effectively navigating epistatic landscapes.[11–13] MLDE utilizes supervised ML models trained on sequence-fitness data to capture non-additive effects. The trained models can then be used to predict high-fitness variants across the entire landscape in a single evaluation round (MLDE)[11,12,14] or iteratively in an active learning (ALDE) fashion.[15–17]

The choice of the training set can greatly influence the performance of the ML models. One can randomly sample the full combinatorial space for training the model (MLDE) or alternatively do focused training (ftMLDE)[12] by selectively sampling to avoid low-fitness variants. In the latter, the quality of the training set can be enriched with more-informative variants using zero-shot (ZS) predictors to reach high-fitness variants more effectively.[12] ZS predictors estimate protein fitness without the need for experimental data: they are instead based on prior assumptions and leverage auxiliary information, such as protein stability, evolutionary data, or structural information.[18–26]

Although ML in protein engineering has been demonstrated in different case studies,[11,27–37] most MLDE[11,15,16] and ftMLDE[12] studies on epistatic landscapes have been benchmarked against a single dataset on the B1 domain of an immunoglobulin-binding protein G (GB1).[38] Thus, two key issues persist: first, the effectiveness of different MLDE strategies on proteins with complex functions, such as enzymes, remains uncertain and, second, the principles that guide successful use of MLDE strategies across diverse protein properties are not understood. Furthermore, despite a growing array of ZS options,[25] there is no definitive guideline for selecting predictors for a given application. This challenge is particularly pronounced for combinatorial epistatic landscapes.[23,39]

Recent experimental studies[40–45] provide a wealth of data on a broader array of protein fitness landscapes, enabling us to start establishing best practices and generalizable guidelines for practitioners working with various proteins. To contextualize the benefits of MLDE, ALDE, and focused training, we conducted a comprehensive study of 16 diverse combinatorial protein fitness landscapes. They span six protein systems and two function types (protein binding and enzyme activity). Consisting of variants that are simultaneously mutated at three or four residues, these landscapes vary in landscape attributes (**Box 3.1**), such as statistical measures (including the number of active variants and fitness distribution properties) as well as ruggedness (a measure of the prevalence of fitness interactions among variants,[46] including pairwise epistasis and the number of local optima). This study focuses on two questions using comprehensive computational analyses: (1) When do MLDE, multiple rounds (such as in ALDE), and focused training (ft) offer a significant advantage compared to DE? (2) How can we best select and utilize the ZS predictor(s) for focused training?

Box 3.1. *Definitions of important terms used in this study.*

Term	Definition
N-site (n_{site}) combinatorial landscape	A mapping of $20^{n_{site}}$ possible sequences to their corresponding fitness values. Specific sites can be selected based on prior experiments, system-specific insights, or computational predictions.
Library	A set of experimentally generated protein variants covering a portion of the landscape.
Navigability	The ability to traverse the fitness landscape from lower- to higher-fitness sequences through single amino acid substitutions in a sequence.
Landscape attribute	Empirically derived values used to quantify a fitness landscape's navigability.
Fitness statistics	Landscape attributes derived from the fitness distribution, including the percent of active variants and distribution shape (i.e., tail behavior, peak location, and multimodality).
Ruggedness[48,50]	Landscape attributes quantifying (1) non-magnitude epistasis, challenging for DE to navigate and (2) local optima, where no neighboring variant with a single amino acid substitution has higher fitness.
Rounds (n_{round})	The number of generations of wet-lab experiments conducted to collect fitness values.
Total number of unique variants (n_{total})	The total number of unique variants sampled by a given strategy across all rounds.
Site-saturation mutagenesis (SSM)[10]	An experimental method for generating libraries by substituting targeted amino acids with many or all 19 other possible amino acids.
Directed evolution (DE)	An iterative protein engineering process that begins with an active variant ("parent") and accumulates amino acid substitutions to improve protein fitness. DE strategies include recombination ("recomb" and "top96 recomb", both with n_{round} = 2 rounds, which consist of a round of simultaneous SSM at all n_{site} sites and a test round) and greedy hill climbing ("single-step", with $n_{round} = n_{site}$ rounds, which consists of successive rounds of SSM at each site). The total number of unique variants is given by $n_{total} = n_{sample} + n_{test}$, where $n_{sample} = 19 \times n_{site} + 1$ for all strategies. The number of test variants n_{test} is 1 for recomb DE, 96 for top96 recomb DE, and 0 for single-step DE.
Machine learning-assisted DE (MLDE)	Implemented in two rounds: training-validation round (with $n_{train} = n_{total} - n_{test}$ variants) and testing round (with n_{test} = 96 variants). We report models trained with boosting ensembles using one-hot sequence encoding in the main text. See the Discussion and Supplemental information for alternative models (ridge regression), sequence representations (ESM-2 with three pooling options), and strategies (ESM-2 fine-tuning).
Active learning-assisted DE (ALDE)	Implemented in multiple rounds (n_{round} = 2, 3, or 4), resembling batch Bayesian optimization. Each round samples $n_{batch} = n_{total} / n_{round}$ variants. We report boosting ensembles with greedy acquisition using one-hot sequence encodings in the main text. See the Discussion and Supplemental information for alternative models (deep neural network ensembles) and acquisition functions (upper confidence bound and Thompson sampling).
Zero-shot (ZS) predictor	A computational approach to estimating protein fitness based on prior assumptions and auxiliary information (e.g., protein stability calculations, evolutionary data, or structural information).
Zero-shot (ZS) score	A numerical score generated by a ZS predictor approximating protein fitness.
Focused training (ft)	A variation of MLDE (ftMLDE) and ALDE (ftALDE) where the initial round sampling is enriched with variants whose ZS scores are in the top 12.5% of the full library (referred to also as the focused training library). See the Discussion and Supplemental information for alternative cutoffs.

3.2 Results for Chapter 3

3.2.1 Overview of landscapes

For this study, we selected 16 experimental combinatorial landscapes covering a range of binding interactions and enzyme activities (**Tables 3.1** and **B.3.1**; **Box 3.1**; **Appendix B.2 Methods**).[38,41–45] All landscapes feature mutations at binding interaction points, in active sites, or at positions previously shown to modulate fitness, all of which are often targeted for engineering tasks (**Figure 3.1c**). For binding, we examined two three-site bacterial toxin-antitoxin ParD-ParE landscapes[41] and the GB1 landscape for immunoglobulin binding.[38] For enzyme activity, we analyzed a three-site dihydrofolate reductase (DHFR) landscape,[42] a three-site T7 RNA polymerase landscape,[43,44] a four-site TEV protease landscape,[43,44] and ten three- or four-site landscapes of the thermostable β-subunit of tryptophan synthase (TrpB).[45]

Table 3.1. *Combinatorial landscapes analyzed in the main text (see* ***Table B.3.1, Box 3.1****, and* ***Appendix B.2 Methods*** *for details).*[38,41–45]

Landscape	Protein	Function	Fitness	MSA depth	Variant space	Sites	Percent active	Fraction of local optima	Fraction of non-magnitude epistasis
ParD2	Bacterial toxin-antitoxin	Binding interactions	ParE toxin neutralization	6789	20^3 (8,000)	I61, L64, K80	82.89	0.001	0.34
ParD3				6784		D61, K64, E80	91.96	0.001	0.31
GB1	Protein G B1 domain	Binding interactions	Binding affinity for IgG-Fc	29	20^4 (160,000)	V39, D40, G41, V54	23.13	0.005	0.40
DHFR	Dihydrofolate reductase	Enzyme activity	Trimethoprim resistance	16042	20^3 (8,000)	A26, D27, L28	10.68	0.004	0.42
T7	T7 RNA polymerase	Enzyme activity	T7 RNA polymerase activity	309	20^3 (8,000)	N748, R756, Q758	3.48	0.368	0.52
TEV	TEV protease	Enzyme activity	TEV protease activity	164	20^4 (160,000)	T146, D148, H167, S170	11.5	0.060	0.56
TrpB3D	β-subunit of tryptophan synthase	Enzyme activity	Tryptophan formation	5816	20^3 (8,000)	T117, A118, A119	9.26	0.043	0.50
TrpB3E						F184, G185, S186	2.02	0.348	0.63
TrpB3F						L162, I166, Y301	1.06	0.232	0.54
TrpB3G						V227, S228, Y301	1.37	0.213	0.52
TrpB3I						Y182, V183, F184	32.04	0.006	0.43
TrpB4					20^4 (160,000)	V183, F184, V227, S228	6.15	0.057	0.46

We first characterized the landscapes to provide interpretation for further evaluations. To minimize the misalignment between theoretical landscape modeling and experimental applications, we selected two groups of empirical and interpretable attributes derived directly from the datasets: (1) fitness statistics (which do not incorporate sequence information) and (2) landscape ruggedness (which involves mapping sequences to fitness) (**Figure 3.1a**; **Box 3.1**; **Appendix B.2 Methods**).[47–54] We used the following statistics to indirectly infer the complexity of the fitness landscape: the percentage of active variants, the fitness value corresponding to the Cauchy peak location, the kurtosis of the fitness distribution ("tailedness"), and the number of kernel density estimation (KDE) peaks (**Appendix B.2 Methods**). The Cauchy distribution is known for its heavy tails. We sought to use the fitness corresponding to its peak location as a landscape attribute to capture the majority of the variant finesses. KDE is a non-parametric method for estimating the probability density function of fitness distribution. KDE does not assume any specific underlying distribution for fitness and is useful for smoothing out noise. We reasoned that the number of KDE peaks, reflecting the distribution modalities of fitness, could serve as a proxy for the underlying landscape navigability, which impacts the outcome of DE.

To quantify ruggedness, which poses navigation challenges for DE,[48,50] we included the number of local optima and the percentage of pairwise non-magnitude epistasis (**Figure 3.1a**). We defined a local optimum as a variant that possesses higher fitness than all its active neighbors differing by a single amino acid substitution (**Appendix B.2 Methods**).[42,45,47,48,50] Recent studies have also highlighted the impact of epistasis on DE[3,45] and emphasized that the majority of epistasis is pairwise.[55] Thus, we also included the amount of non-magnitude pairwise epistasis (challenging for DE to navigate) as a relevant landscape attribute (**Appendix B.2 Methods**).

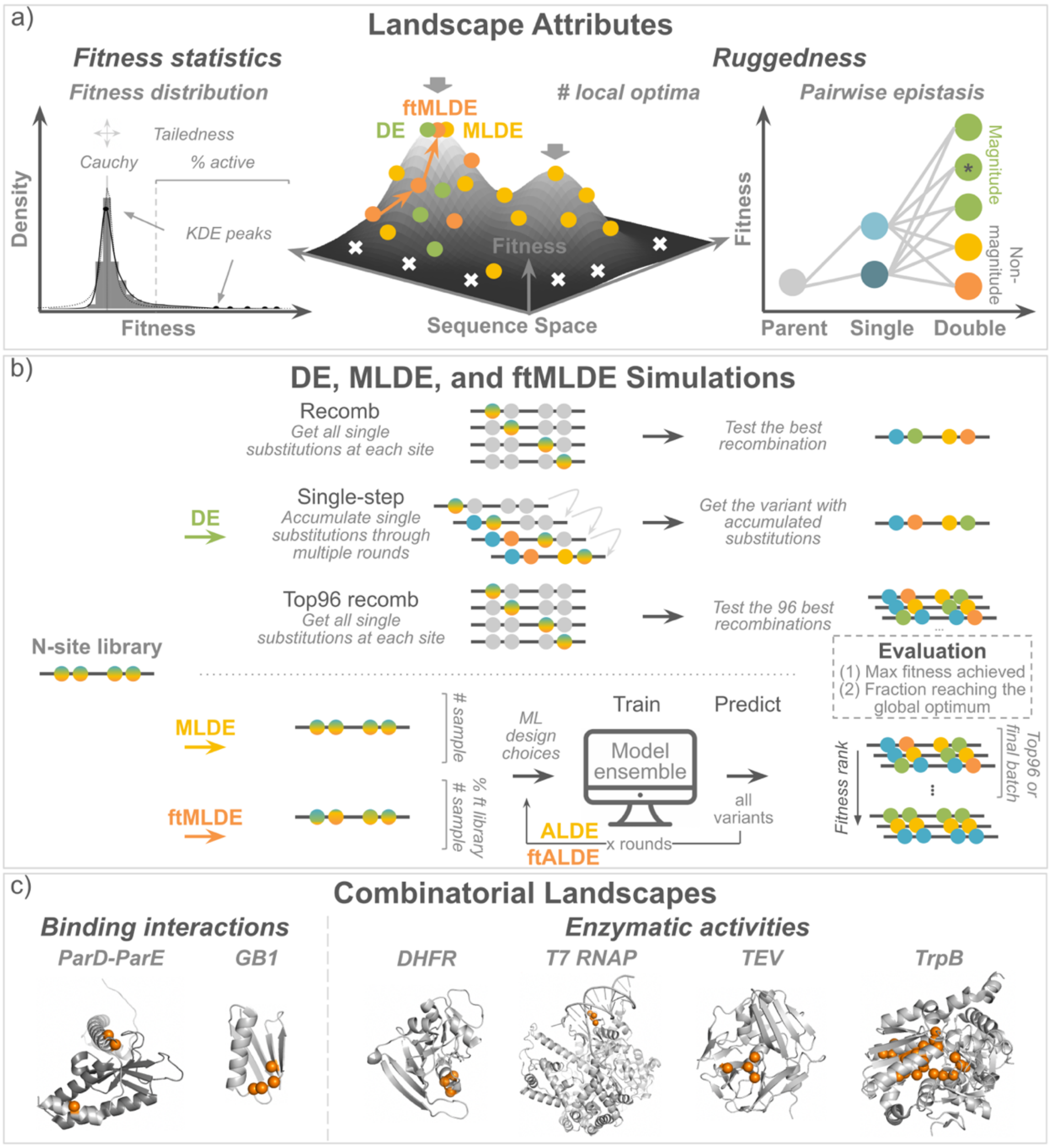


Figure 3.1. *Summary of landscape attributes, simulations, and combinatorial landscapes. a) Landscape attributes include (1) fitness statistics (percent of active variants, tailedness, location of Cauchy peak, and number of peaks for kernel density estimation) and (2) ruggedness (number of local optima and percent of non-magnitude pairwise epistasis) (**Box 3.1**; **Appendix B.2 Methods**). b)* In silico *simulations include three types of DE strategies (recomb, single-step, and top96 recomb), MLDE, multiple rounds of MLDE (ALDE), and focused training for both MLDE (ftMLDE) and ALDE (ftALDE) (**Box 3.1**; **Appendix B.2 Methods**). All DE and ML simulations were performed for each N-site library of a given landscape, where n_{site} is the number of sites targeted for mutation. All strategies were evaluated using two metrics: (1) average maximum fitness achieved and (2) fraction reaching the global optimum. For ML simulations, a range of total unique variants screened (n_{total}) was considered. MLDE and ftMLDE divided n_{total} into a training-validation round (n_{train}) and an*

*evaluation round ($n_{test} = 96$, $n_{round} = 2$). Evaluation metrics were calculated based on the fitness of the top 96 ranked variants from the evaluation round. ALDE and ftALDE split n_{total} across multiple rounds ($n_{round} = 2, 3, or\ 4$), with each round sampling $n_{batch} = n_{total}\ /\ n_{round}$ variants. Evaluation metrics were calculated based on the fitness of the variants sampled in the final batch. MLDE and ALDE sampled the initial round randomly from the full N-site library, while ftMLDE and ftALDE sampled the initial round randomly from a focused library containing variants ranked in the top 12.5% of ZS scores from the full N-site library. Different models, sequence encodings, and focus training library cutoffs were tested (**Box 3.1**; **Appendix B.2 Methods**). c) Combinatorial landscapes studied, categorized by the number of targeted sites ($n_{site} = 3\ or\ 4$) and function types (binding interactions and enzyme activities) on six protein systems (ParD-ParE toxin-antitoxin, GB1 immunoglobulin binding, dihydrofolate reductase, T7 RNA polymerase, TEV protease, and tryptophan synthase).[57][38,41,43–45] Targeted sites are highlighted in orange. We focus on libraries with at least 1% active variants in the main text, while the remaining landscapes are detailed in the Supplemental information. The 1% threshold was extrapolated from the expected occurrence of one active variant in a traditional DE screening method of the largest landscape in this study ($1\ /\ (4 \times 20) \times 100\%$; **Appendix B.2 Methods**).*

3.2.2 All MLDE strategies consistently outperform DE, particularly as landscape navigability decreases

We first assessed how landscape attributes influence the efficacy of protein engineering strategies. Specifically, we evaluated the outcomes of a protein engineering campaign using two metrics: (1) "average maximum fitness achieved" which is the fitness of the final variant achieved by each method on average and (2) "fraction reaching the global optimum", which measures how frequently the true maximum fitness is reached. We explored these measures across MLDE, ALDE, focused training, and three different DE strategies (**Box 3.1**). The DE strategies are "recomb", a recombination of the best SSM variant at each site ($n_{total} = 19 \times n_{site} + 2$ variants over $n_{round} = 2$ rounds); "single-step", an iterative process starting from any site with subsequent variants built on the best variant found ($n_{total} = 19 \times n_{site} + 1$ variants over $n_{round} = n_{site}$ rounds); and "top96 recomb", where SSM is performed at each position, all substitution combinations are calculated based on additive recombination, and the top 96 variants are selected ($n_{total} = 19 \times n_{site} + 97$ variants over $n_{round} = 2$ rounds, where 96 is the number of wells in plates commonly used for screening; **Figure 3.1b**; **Box 3.1**; **Appendix B.2 Methods**).[11,12,45] MLDE and ftMLDE trained an ensemble of models, employing either random or ZS predictor-guided training sample selection. The trained models were then used to predict fitness for all variants, where the top 96 predicted variants were then used for evaluation (**Figure 3.1b**; **Box 3.1**; **Appendix B.2

Methods).[12] ALDE divided the total sample size equally into multiple rounds ($n_{round} = 2, 3, or$ 4), with each round of sampling guided by the acquisition function (**Figure 3.1b**; **Box 3.1**; **Appendix B.2 Methods**).[17] Similar to how ftMLDE improved upon MLDE,[12] ftALDE used ZS predictors to select a more informative initial training set, instead of the random sampling used in ALDE.

Considering the variability in throughput and expense of experimental screens, we explored a range of total number of unique variants screened (n_{total}), from 120 to 2,016 samples (**Figure 3.2a**; **Table B.3.2**). On average across landscapes, MLDE (dashed light blue line) required 48 training samples (n_{total} = 144) to outperform recomb DE and 96 (n_{total} = 192) to surpass single-step DE for both metrics. It took 96 training samples (n_{total} = 192) for MLDE to match the average maximum fitness and 384 (n_{total} = 480) to achieve a comparable fraction reaching the global optimum to the most competitive DE strategy, “top96 recomb”. By incorporating various ZS predictors, ftMLDE (solid dark blue line) consistently outperformed MLDE with random sampling (showing a 4–12% improvement in average maximum fitness achieved for up to 960 training samples (n_{total} = 1,056) and a 9–77% improvement in fraction reaching the global optimum across all training sample sizes; **Table B.3.3**); ftMLDE achieved the same levels of average maximum fitness and global optimum fraction as MLDE but with fewer training samples required (**Figure 3.2a**). These results suggest that MLDE can identify high-fitness variants more effectively than DE, and its performance improves with more training data. Focused training with ZS predictors can further improve performance compared to MLDE with random sampling.

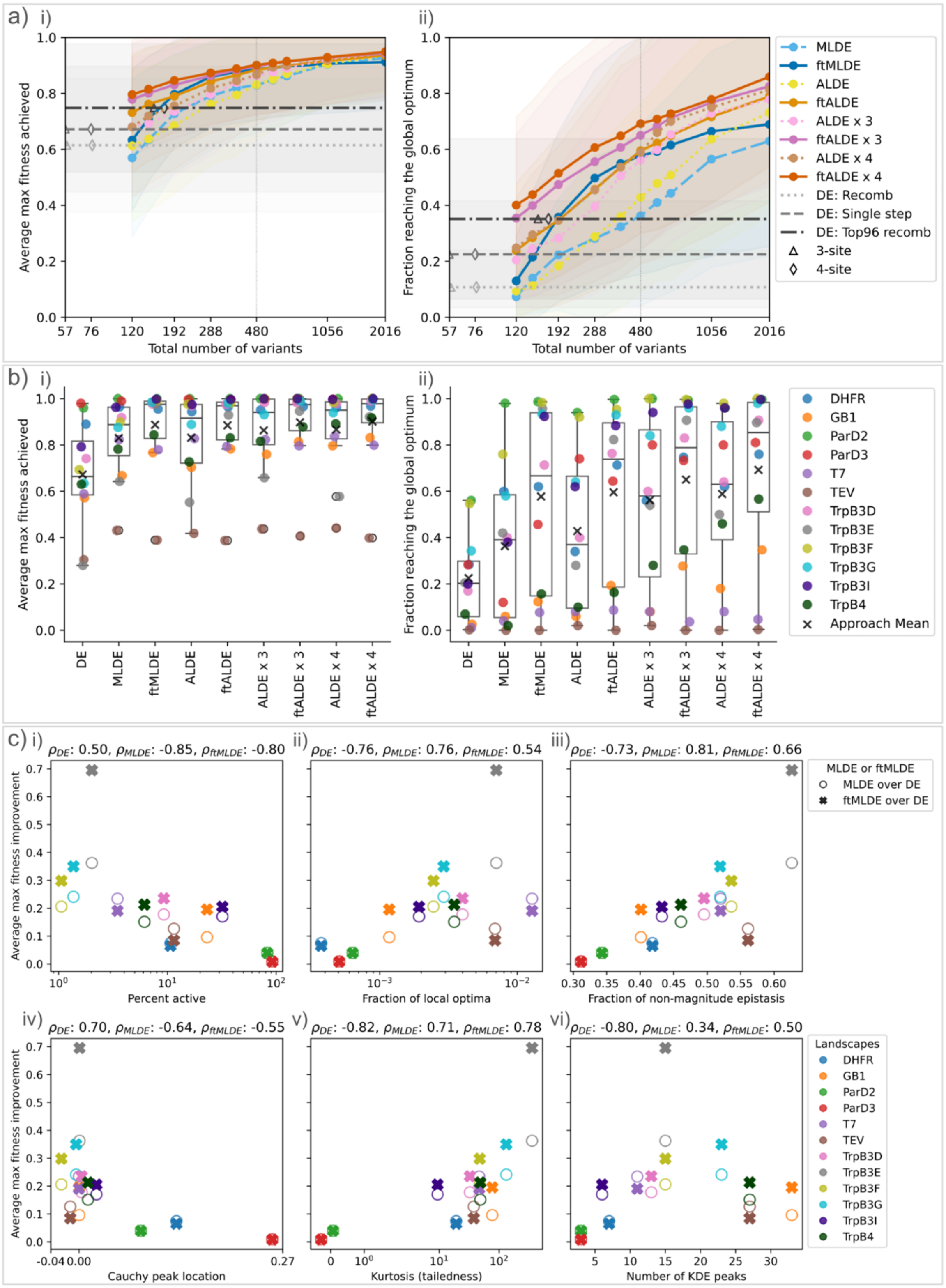

a) i)
Average max fitness achieved
Total number of variants
57 76 120 192 288 480 1056 2016
ii)
Fraction reaching the global optimum
Total number of variants
MLDE
ftMLDE
ALDE
ftALDE
ALDE x 3
ftALDE x 3
ALDE x 4
ftALDE x 4
DE: Recomb
DE: Single step
DE: Top96 recomb
3-site
4-site
b) i)
Average max fitness achieved
DE
MLDE
ftMLDE
ALDE
ftALDE
ALDE x 3
ftALDE x 3
ALDE x 4
ftALDE x 4
ii)
Fraction reaching the global optimum
DHFR
GB1
ParD2
ParD3
T7
TEV
TrpB3D
TrpB3E
TrpB3F
TrpB3G
TrpB3I
TrpB4
Approach Mean
c) i) ρ_{DE}: 0.50, ρ_{MLDE}: -0.85, ρ_{ftMLDE}: -0.80
Average max fitness improvement
Percent active
ii) ρ_{DE}: -0.76, ρ_{MLDE}: 0.76, ρ_{ftMLDE}: 0.54
Fraction of local optima
iii) ρ_{DE}: -0.73, ρ_{MLDE}: 0.81, ρ_{ftMLDE}: 0.66
Fraction of non-magnitude epistasis
MLDE or ftMLDE
MLDE over DE
ftMLDE over DE
iv) ρ_{DE}: 0.70, ρ_{MLDE}: -0.64, ρ_{ftMLDE}: -0.55
Cauchy peak location
v) ρ_{DE}: -0.82, ρ_{MLDE}: 0.71, ρ_{ftMLDE}: 0.78
Kurtosis (tailedness)
vi) ρ_{DE}: -0.80, ρ_{MLDE}: 0.34, ρ_{ftMLDE}: 0.50
Number of KDE peaks
Landscapes

Figure 3.2. *Performance of MLDE, ALDE, and focused training, compared with DE and correlated to six landscape attributes. a) Comparison of DE, MLDE, ALDE, ftMLDE, and ftALDE performance, averaged across 12 landscapes with at least 1% active variants. Shading indicates the standard deviation. Performance is shown for i) the maximum fitness achieved and ii) the fraction of campaigns reaching the global optimum, for different numbers of total unique variants sampled (n_{total}). Three DE methods were included: recomb, single-step, and top96 recomb. DE simulations started from all active variants (**Appendix B.2 Methods**). The triangle and diamond indicate the total number of unique variants sampled for DE, where $n_{total} = n_{sample} + n_{test}$ and $n_{sample} = 19 \times n_{site} + 1$ (**Box 3.1**; **Appendix B.2 Methods**). The vertical line marks a total sample size of 480, where 384 variants were sampled for training and the top 96 predicted variants were used for testing in MLDE and ftMLDE; two rounds of ALDE (ALDE) sampled 240 variants per round; three rounds of ALDE (ALDE x 3) sampled 160 variants per round; or four rounds of ALDE (ALDE x 4) sampled 120 variants per round (i.e., $n_{total} = 480 = n_{train} + n_{test} = 384 + 96 = n_{round} \times n_{batch} = 2 \times 240 = 3 \times 160 = 4 \times 120$). Subsequent results expand on this $n_{total} = 480$ setting. See Figure S1 for landscapes with fewer than 1% active variants, Figure S2 for Elo ratings, and Figures S3–S4 for individual landscapes. b) Single-step DE, MLDE, ALDE, and focused training results broken down by landscape, with a total sample size of 480 ($n_{total} = 480$) for both metrics. See Figure S5 for landscapes with fewer than 1% active variants. c) Spearman's rank correlation of ML strategy performance improvement (the average maximum fitness of the top 96 predicted variants by ML methods over single-step DE, y-axis) with six landscape attributes (x-axis): i) percentage of active variants, ii) fraction of local optima, iii) fraction of non-magnitude epistasis, iv) Cauchy peak location, v) kurtosis (tailedness), and vi) number of kernel density estimation (KDE) peaks (**Appendix B.2 Methods**). See Figure S6 for ALDE and ftALDE, Table S4 for attribute-performance correlation and Table S5 for fold improvements. For each MLDE and ftMLDE simulation, boosting models were trained on a range of n_{train} random samples from the entire or ZS-focused library using one-hot sequence encoding, with five-fold cross-validation. For each ALDE and ftALDE simulation, boosting ensembles with greedy acquisition function were trained with $n_{batch} = n_{total}/n_{round}$ variants per round for n_{round}=2, 3, or 4, respectively. The top 96 or final batch variants were evaluated. Each ML simulation was averaged across 50 replicates (**Box 3.1**; **Appendix B.2 Methods**). ftMLDE and ftALDE performance were further averaged across six ZS predictors (details in the next section).*

Next, we compared ALDE (MLDE with multiple rounds of training and testing guided by the acquisition function)[17] to MLDE (a single round of training and testing, equivalent to two rounds) with the same total number of variants screened. With two rounds, ALDE (dotted bright yellow line) began to outperform MLDE (dashed light blue line) after 480 total samples for average maximum fitness achieved and 288 total samples for fraction reaching the global optimum but did not outperform ftMLDE (solid dark blue line) until 1,056 samples for both metrics. With four rounds, ALDE (dotted light brown line) matched or exceeded ftMLDE performance. With focused training, ftALDE matched or surpassed ftMLDE with the same number of rounds ($n_{round} = 2$) and showed further improvement with additional rounds (ftALDE x 3 and ftALDE x 4; **Figure 3.2a**). However, for libraries with fewer than 1% active variants, even four rounds of ALDE (without focused training) consistently underperformed compared to ftMLDE (**Figure B.3.1**). Our observations underscore the utility of focused training using ZS predictors here, enabling MLDE to match multi-round ALDE performance and offering further improvement to ALDE.

Given the large standard deviations in the performance of different approaches across landscapes, we first validated our comparisons using the Elo rating system, a method widely used in model benchmarking and competitive games (**Figure B.3.2**; **Appendix B.2 Methods**).[56,57] We then examined how each approach performed on individual landscapes and found that some landscapes exhibited more significant improvements than others (**Figure 3.2b** and **Figures B.3.3–B.3.5**). We quantified the improvements of ML strategies over single-step DE and found that ML strategies offered a greater advantage on landscapes which were more challenging for DE to navigate. To better understand when DE struggled, we calculated six different attributes to provide insights into landscape navigability (**Figure 3.2c**; **Appendix B.2 Methods**). Specifically, the mean maximum fitness achieved by DE correlated positively with the fraction of active variants (Spearman's $\rho = 0.50$) and the fitness distribution's Cauchy peak location (Spearman's $\rho = 0.70$), indicating improving navigability by DE (**Table B.3.4**). Consequently, the improvements resulted from all ML methods were anti-correlated with percent active (**Figure 3.2c–i**) and Cauchy peak location (**Figure 3.2c–iv**). Greater kurtosis (tailedness) and number of KDE peaks of the fitness distribution hindered DE navigability (Spearman's $\rho = -0.82$ and -0.80, respectively; **Table B.3.4**). ML methods thus improved performance most significantly for landscapes with high tailedness and more KDE peaks (**Figures 3.2c–v** and **vi**). Similarly, increased landscape ruggedness decreased DE navigability, yielding greater benefit of using ML methods over DE for such landscapes. DE navigability was anti-correlated with the fraction of local optima and the fraction of non-magnitude epistasis (Spearman's $\rho = -0.76$ and -0.73, respectively; **Table B.3.4**), and thus the net improvement of ML methods over DE was positively correlated with both more local optima (**Figure 3.2c–ii**) and more non-magnitude epistasis (**Figure 3.2c–iii**). Indeed, ftMLDE demonstrated the most substantial performance improvements (3.5-fold) for one of the least navigable landscapes (TrpB3E; **Table B.3.5**). The performance improvements from different rounds of ALDE and ftALDE were also correlated with landscape navigability defined by the six attributes (Figure **B.3.6**; Tables **B.3.4** and **B.3.5**).

3.2.3 ZS predictors provide orthogonal priors on protein fitness that improve focused training performance

Next, we sought to understand the effectiveness of different ZS predictors for fitness prediction and their ability to improve ftMLDE and ftALDE performance across landscapes. ZS predictors could be useful for (1) effectively ranking variants to sample the fittest mutants (measured by Spearman's correlation, Methods) and (2) classifying active/inactive variants to avoid sampling non-viable variants, especially in landscapes dominated by inactive variants (measured by active/inactive accuracy ROC-AUC, **Appendix B.2 Methods**). To evaluate the effectiveness and limitations of various ZS predictors under different assumptions and priors, we selected six distinct predictors across two axes: calculation vs. learning-based and sequence vs. structure-based. These predictors are Hamming distances, EVmutation, ESM-2, ESM-IF, CoVES, and Triad (**Figure 3.3a**; **Appendix B.2 Methods**).[18–22,59] To differentiate our work from comprehensive ZS predictor benchmarks that are largely limited to measuring the effects of single amino acid substitutions,[25] we emphasized their utilities for focused training applications in epistatic landscapes.

As a baseline, we used the Hamming distance as a ZS predictor, which counts the number of amino acid substitutions from the parent, a variant already exhibiting some activity. By setting a Hamming distance threshold, we essentially confined the sampling space to the vicinity of the parent to enrich the training set with more viable variants on average, since most mutations are deleterious.[5] Indeed, we observed that the Hamming distance (indicated in blue) showed a weak correlation with fitness ranking (**Figure 3.3b–i**) and classified active/inactive variants better than random (**Figure 3.3b–ii**) as a ZS predictor. Notably, Hamming distance relied on the parent defined by the authors of each landscape, rather than a randomly sampled active variant as in the DE simulations. Since these landscapes were designed to have variant activity levels both higher and lower than the parent, the parent and its neighboring variants were likely to be more fit and active than those around a randomly selected active variant (**Figure B.3.7**). Although the landscape parent was one of the most active variants, contributing to strong Hamming distance performance, Hamming distance still outperformed random predictions in fitness ranking (**Figure B.3.8**) and active/inactive classification (**Figure B.3.9**) on average, when different active variants were used as the parent. In the focused training setting, Hamming distance-guided training set sampling outperformed random selection, improving both the average maximum fitness achieved across all total sample sizes (**Figure 3.3e–i**) and fraction reaching the global optimum, up to total sample sizes of 1,056 for ftMLDE (**Figure 3.3e–ii**) and 480 for ftALDE (**Figure B.3.10**).

Various ZS predictors can incorporate implicit evolutionary conservation based on the distribution of naturally occurring sequences. The EVmutation score predicts the fitness effect of a given set of substitutions based on conservation and evolutionary couplings through multiple-sequence alignment (MSA).[18,60] We observed that EVmutation (indicated in green) outperformed the Hamming distance for both ranking fitness values (**Figure 3.3b–i**) and classifying active/inactive variants (**Figure 3.3b–ii**). Moreover, it was one of the best ZS predictors for focused training across all sample sizes on both metrics (**Figure 3.3e**). Similarly, protein language models (PLMs) can capture these evolutionary conservations by learning to predict the original identity of masked or corrupted amino acids.[61,26,62–66] The likelihood of filling such amino acids can be thought of as a predictor for different amino acid substitutions given the sequence context.[19] The ESM-2 score (Evolutionary Scale Modeling,[26] indicated in purple) from one of such state-of-the-art PLMs performed similarly to EVmutation as a ZS predictor for both fitness ranking (**Figure 3.3b–i**) and active/inactive classification (**Figure 3.3b–ii**). It also did not further improve upon EVmutation in the focused training setting (**Figure 3.3e**).

Incorporating structural context can also be useful for ZS predictions. ESM-IF (ESM inverse-folding, indicated in yellow) is an inverse-folding model trained to predict a protein sequence from its backbone atom coordinates, where effects of substituting amino acids can be approximated with the likelihoods of each possible sequence for this reconstruction task.[20]

We observed that ESM-IF was one of the best ZS predictors for fitness ranking (**Figure 3.3b–i**) and active/inactive classification (**Figure 3.3b-ii**), alongside EVmutation and ESM-2. In the focused training setting, ESM-IF score did offer a consistent advantage over the sequence-only ESM-2, but only offered a slight advantage over EVmutation at either low or high number of samples (i.e., n_{total}=120, 144, 1,056, and 2,016, **Figure 3.3e**). CoVES (Combinatorial Variant Effects from Structure, indicated in brown) learns to predict a masked amino acid identity from its surrounding atomic-level structural microenvironments but does not account for epistasis.[21] Compared to ESM-IF, we observed that CoVES was a slightly less effective ZS predictor for fitness estimation (**Figure 3.3b**) and in the focused training setting (**Figure 3.3e**), but it was still one of the most effective predictors for improving over random sampling.

An alternative local structure-based ZS score utilizes physics-informed stability calculations. Stability is an important prior for protein function, as an unfolded or misfolded protein is less likely to be functional.[12,22] The Triad score estimates mutant stability by calculating the change in its free energy of folding relative to the parent ($\Delta\Delta G_f$) using a Rosetta energy function.[12,59] While Triad (indicated in orange) was the weakest predictor for variant fitness ranking (**Figure 3.3b–i**), it classified active/inactive variants fairly well (**Figure 3.3b–ii**) as a ZS predictor. Triad-guided training set sampling outperformed random selection in the ftMLDE setting, up to a total sample size of 1,056 for both metrics (**Figure 3.3e**). In the ftALDE setting, it outperformed random selection up to a total sample size of 576 for average maximum fitness and 384 for the fraction reaching the global optimum (**Figure B.3.10**). The relative differences between ZS predictors in focused training remained consistent across different rounds of ftALDE. However, in libraries with fewer than 1% active variants, these differences were minimized, and all ZS-guided focused training approaches showed a significant advantage over random sampling (**Figures B.3.11– B.3.12**).

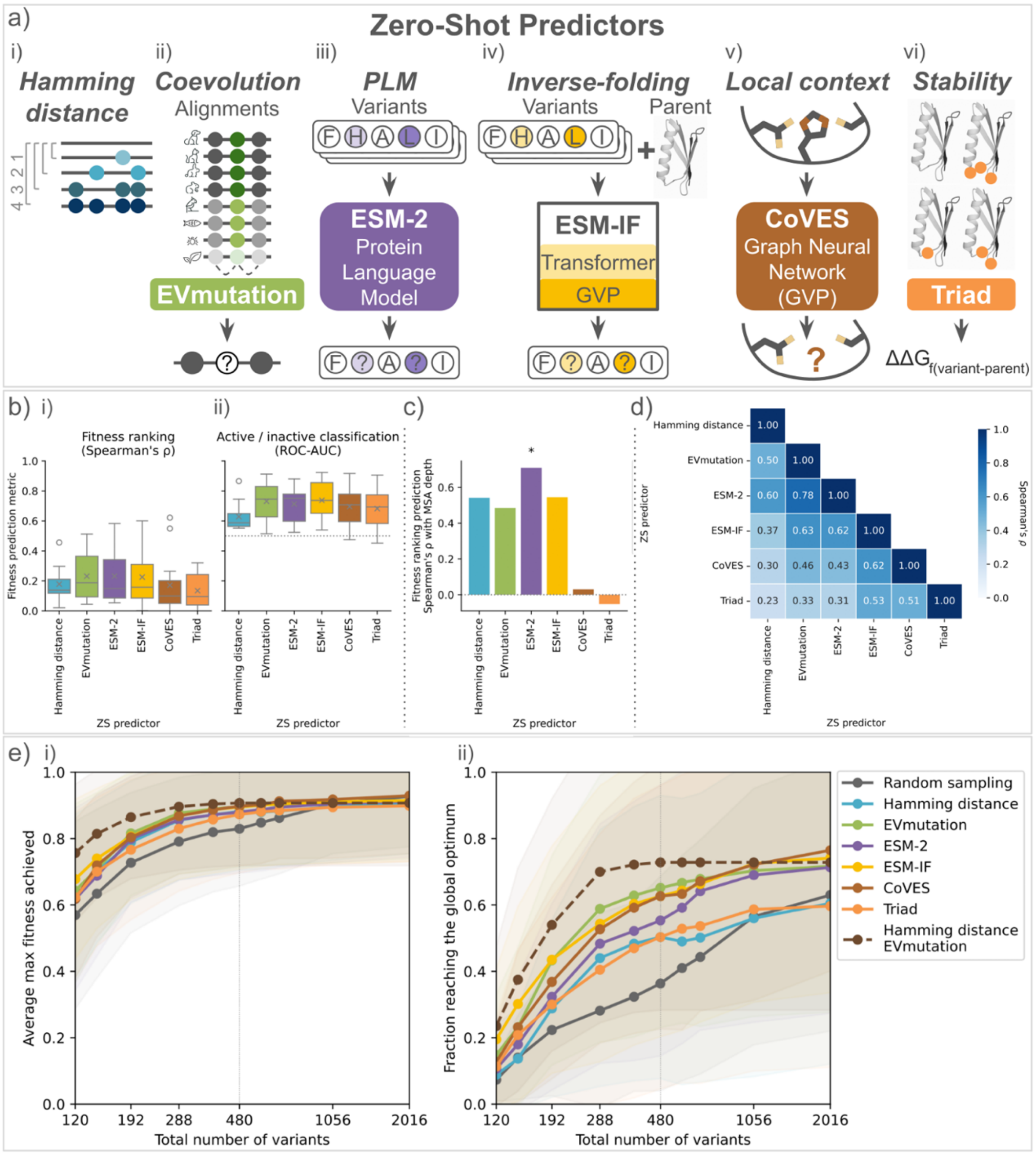


Figure 3.3. *Summary of different ZS predictors and their impacts on focused training across landscapes. a) Six ZS predictors: i) Hamming distance, ii) EVmutation (coevolutionary conservation),*[18,60] *iii) ESM-2 (mutant likelihood from pretrained protein-language model),*[19,26] *iv) ESM-IF (mutant likelihood from pretrained inverse-folding models based on sequence and structure information),*[20] *and v) Triad (mutant stability $\Delta\Delta G_f$).*[12,59] *b) ZS predictor performances in terms of correlation between ZS score ranking and fitness ranking (Spearman's ρ) and active/inactive classification accuracy (ROC-AUC) across 12 landscapes with at least 1% active variants. The cross and central line represent the mean and median, respectively. Boxes show the interquartile range (IQR), and whiskers extend up to 1.5× IQR. Outliers (gray circles) represent values beyond this range. Dotted gray line indicates random classification. c) Correlation of the fitness ranking Spearman's ρ*

*with MSA depth for each ZS predictor. Statistical significance (p-value <0.05) is indicated as * (**Table B.3.6**). Dotted gray line indicates no correlation. d) Pairwise Spearman's correlation among six ZS predictors averaged across 12 landscapes. e) Performance of focused training with different ZS predictors including the best Hamming distance ensemble, averaged across 12 landscapes. Shading indicates the standard deviation. It assesses i) the maximum fitness achieved and ii) the fraction reaching the global optimum in relation to the number of samples used by ftMLDE. For each MLDE and ftMLDE simulation, boosting models were trained on 384 random samples from the entire or ZS-focused library using one-hot sequence encoding, with five-fold cross-validation. The top 96 predicted variants were evaluated. Each ML simulation was averaged across 50 replicates (**Appendix B.2 Methods**). The vertical line marks a total sample size of 480 (e.g., 384 sampled variants for training and top 96 predicted variants for testing). See **Figure B.3.10** for ALDE and ftALDE, **Figures B.3.11**, **B.3.12**, and **B.3.15** for landscapes with fewer than 1% active variants, **Figure B.3.16** for Elo ratings, and **B.3.17–B.3.22** for individual landscapes.*

To facilitate ZS predictor selection and ensembling, we first examined how the fitness ranking performance of ZS predictors correlated with the depth of multiple sequence alignments (MSAs) (**Figure 3.3c**; **Table B.3.6**; **Appendix B.2 Methods**). We found that the performance of the physics-based Triad and the structure-only CoVES did not correlate with MSA depth, confirming their independence from evolutionary data. In contrast, the three sequence-based predictors, Hamming distance, EVmutation, and ESM-2 did show correlation with MSA depth. Despite being a hybrid sequence-structure model, ESM-IF captured evolutionary information to a similar extent as EVmutation, likely because over 99% of its structures were predicted from similar sequence databases (the UniRef family).[20]

We then investigated the relationship between different ZS predictors using pairwise correlations (**Figure 3.3d**; **Appendix B.2 Methods**).[67,68] Within each modality, sequence-based (Hamming distance, EVmutation, and ESM-2) or structure-based (CoVES and Triad), all ZS predictors exhibited at least a 0.5 Spearman's correlation with each other. ESM-2 and EVmutation showed the strongest correlation (Spearman's $\rho = 0.78$), suggesting PLMs like ESM-2 may capture similar coevolutionary information as MSAs used by EVmutation.[26,69,70] ESM-IF showed similar correlations with both the structure-only CoVES (Spearman's $\rho = 0.62$) and the sequence-only ESM-2 (Spearman's $\rho = 0.62$) and EVmutation (Spearman's $\rho = 0.63$). Despite being distinct approaches, all four learning-based predictors captured related information. However, Triad had only moderate correlations with the other two structure-based predictors (Spearman's $\rho = 0.5$) and weak correlations with the three sequence-based predictors (Spearman's $\rho < 0.4$). Hamming distance showed moderate correlations with

ESM-2 and EVmutation (Spearman's ρ = 0.6 and 0.5, respectively) but it was weakly correlated with the structurally inclined predictors (Spearman's $\rho < 0.4$). This underscores the orthogonality between learning-based models, a naive protein engineering prior, and a physics-based approach. Thus, we hypothesized that ensembling orthogonal ZS predictors may further enhance focused training by synergizing complementary information sources.

We evaluated whether ensembling Hamming distance with other ZS predictors enhanced focused training performance compared to each predictor alone. Prefiltering with a Hamming distance (by restricting variants to those within two amino acid substitutions of the parent sequence) boosted focused training performance from each ZS predictor up to 192 total samples (left vertical gray line in **Figure B.3.13**). This benefit extended to 480 total samples for ESM-2 and ESM-IF (right vertical gray line in **Figure B.3.13**) and continued to 1,056 for EVmutation for both average max fitness achieved (**Figure 3.3e–i**) and fraction reaching the global optimum (**Figure 3.3e–ii**). However, the benefits diminished beyond a total sample size of 480, due to this pre-constrained sampling space. In contrast, ZS ensembles with Triad did not show significant improvement (**Figures B.3.14– B.3.15**). This suggests that, despite its orthogonal information, the physics-based Triad predictor offered no further benefit to focused training performance. Similarly, naively ensembling the two top-performing predictors, ESM-IF and EVmutation, yielded no additional improvements nor did naive ensembling of other high-performing but orthogonal predictors (i.e., CoVES with EVmutation or with ESM-2, **Figures B.3.14– B.3.15**). Elo ratings for ZS predictors and ensembles, along with individual landscape analysis, validated the advantage of ZS-guided focused training and the additional benefit of combining Hamming distance with other informative ZS predictors, especially for low total sample numbers (e.g., $n_{total} \leq 480$, **Figures B.3.16–B.3.22**; **Appendix B.2 Methods**).

3.2.4 Landscape and functional attributes affect ZS predictability

We next examined how ZS predictability differed across landscapes, specifically comparing those measuring binding interactions vs. enzyme activities (**Figure 3.4a**). All six ZS predictors ranked fitness values substantially better for binding interactions than for enzyme activity (**Figure 3.4a–i**), with Triad showing a statistically significant difference (p-value = 0.001, **Table B.3.7**). The structure-based predictors (CoVES and Triad) were better at classifying active/inactive variants for binding datasets than for enzymatic ones, while the sequence-based predictors (Hamming distance, EVmutation, and ESM-2) performed better for the enzyme activity datasets. Hamming distance showed a statistically significant difference in the context of classification (p-value = 0.042, **Figure 3.4a–ii**; **Table B.3.7**). However, the differences between binding interactions and enzyme activities were no longer statistically significant for any ZS predictors in the focused training setting across different ML strategies (**Figure 3.4b**; **Tables B.3.8– B.3.9**).

For 10 out of the 12 landscapes, all six ZS predictors successfully focused the training set to be more informative than random sampling, leading to improved ftMLDE performance for both metrics (**Figure 3.4b**). Harder-to-navigate landscapes and the libraries with fewer than 1% active variants benefited more from focused training, provided the ZS predictor for active/inactive variant classification was better than random (ROC-AUC > 0.5, **Figures 3.4**, **B.3.23**, and **B.3.29**). TrpB3E (indicated in gray), one of the hardest-to-navigate landscapes (**Figure 3.2c**) with one of the lowest but still above-random active/inactive variant classification ROC-AUC (**Figure 3.4a–ii**), gained the most from all ZS predictors compared to randomly sampled MLDE training sets for both performance metrics (**Figure 3.4b**). Similar improvements in hard-to-navigate landscapes were consistently observed when comparing ftALDE with ALDE in the same round, and when considering different total sample sizes for each of the focused training approaches (**Figures B.3.24–B.3.28**).

A falsely biased training set could negatively impact focused training performance. For DHFR (dihydrofolate reductase, indicated in blue), the structure-based predictions, Triad $\Delta\Delta G_f$ and CoVES, performed poorly, with worse-than-random active/inactive classification (**Figure 3.4a–ii**, dashed gray line) and harmed ftMLDE performance for both metrics (**Figure 3.4b**).

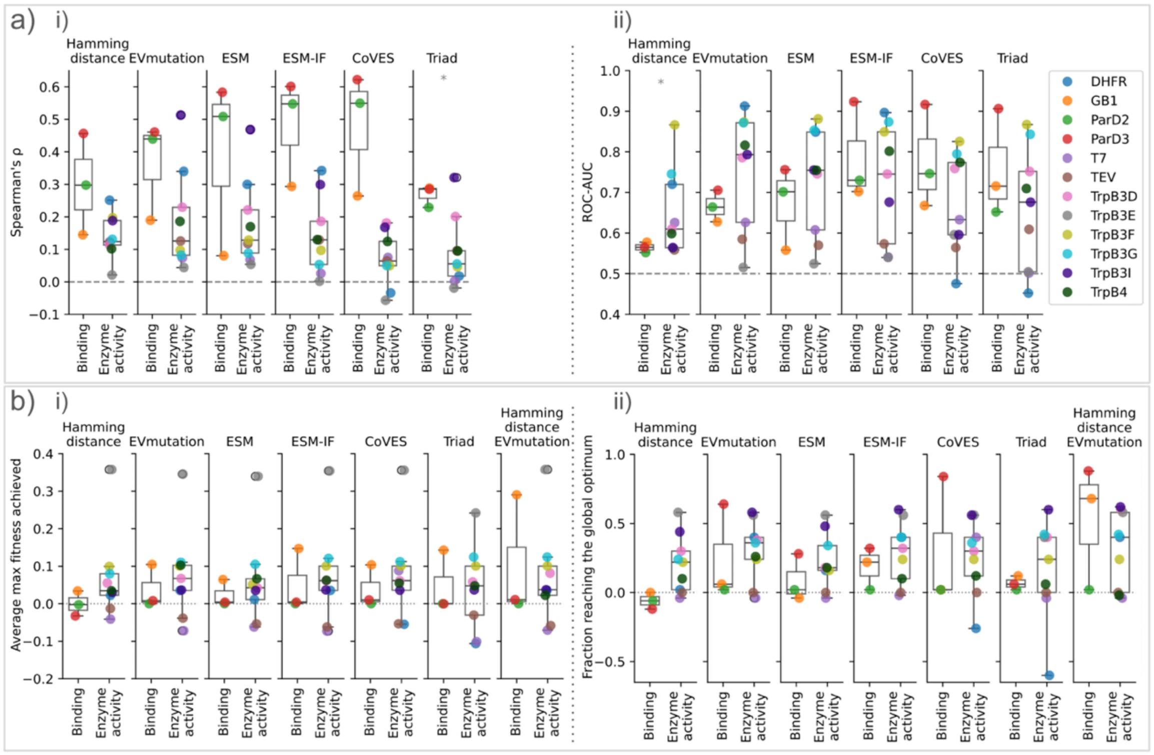


Figure 3.4. *Summary of different ZS scores and their impact on individual landscapes, grouped by function types. a) Six ZS predictor performance for each individual landscape in terms of i) Spearman's correlation of ZS score ranking and fitness values ranking and ii) ROC-AUC of active/inactive classification. Random predictions are indicated in horizontal gray dashed lines. Statistical significance (p-value <0.05) is indicated as *. b) A breakdown of the ftMLDE results with a total sample size of 480 from* ***Figure 3.3e****, categorized by six ZS predictors and two functions (binding interactions and enzyme activities) for each landscape. Focused training improvement over randomly sampled training set (MLDE) is quantified by i) average maximum fitness and ii) fraction reaching the global optimum. See Supplemental information for landscapes with fewer than 1% active variants (****Figures B.3.24–B.3.25****), ftALDE with different rounds (****Figures B.3.25–B.3.26****) and ZS predictor impacts on focused training with 192 total samples (****Figures B.3.24****,* ***B.3.27****, and* ***B.3.28****).*

3.3 Discussion for Chapter 3

Our findings confirmed that all the MLDE strategies tested exceeded or at least matched DE performance across 16 protein fitness landscapes, with the advantages becoming more pronounced as landscape attributes posed greater obstacles for DE (e.g., fewer active variants and more local optima). Using ZS predictors, which leverage various prior knowledge, enriched training sets in focused training enable further performance improvements. Overall, this computational study suggests that MLDE strategies are highly generalizable and can unlock better protein engineering outcomes than DE, and we present key considerations for the effective deployment of these approaches. We expect that these findings will encourage and facilitate the adoption of ML-assisted directed evolution for efficient protein engineering.

We note that making an equitable comparison of traditional DE methods and ML strategies is challenging. Iterative SSM-based DE methods inherently have a limited search space. The recombination methods we assessed only combined variants that contained single amino acid substitutions relative to the starting sequence. Single-step DE requires more rounds of engineering as the number of targeted sites increases, slowing the process. Furthermore, while we evaluated DE using the unique variants, in practice, ensuring adequate coverage of the variant space (e.g., 95%) typically requires close to three-fold oversampling (**Figures B.3.3–B.3.4**; **Appendix B.2 Methods**).[71] In contrast, ML-assisted methods can explore a broader sequence space and continue to improve with more data, but they often rely on require direct synthesis of predicted variants or designed libraries.

As a general recommendation for the implementation of ML strategies to guide protein engineering objectives, we introduce a decision-making process for selecting a campaign strategy (**Figure 3.5**). The first step is to assess whether the landscape is hard to navigate. This typically involves a low percentage of active variants and high percentage of pairwise non-magnitude epistasis, which can be inferred from prior knowledge (e.g., the percentage of active variants from single-site SSM experiments) or from the structural proximity of residues of interest to functional (binding or active) sites and to each other. We observed a weak negative correlation (Spearman's $\rho = -0.34$, **Table B.3.10**) between the percentage of pairwise non-magnitude epistasis (where higher values indicate harder-to-navigate landscapes) and the average pairwise C-alpha distance of substituted residues (the smaller the distance, the closer the central carbon atoms of the two amino acids at the targeted sites, **Appendix B.2 Methods**). Next, determine whether there is a "good-enough" ZS prior–a ZS predictor that can classify active/inactive variants better than random, for example an evolutionary-based predictor when deep MSAs are available and fitness corresponds to largely native activity. Meanwhile, assess whether the number of total variants is low or evolvability is low (e.g., the protein cannot tolerate multiple mutations at once). For hard-to-navigate landscapes without prior information but low n_{total}, using a Hamming distance threshold of two (i.e., constructing double-site libraries) can effectively enrich informed variant sampling for the training sets. Additionally, ZS predictor classification performance on single-site libraries can help identify predictors that may fail on larger combinatorial libraries (**Figures B.3.29–B.3.30**; **Table B.3.11**). For example, CoVES and Triad performed worse than random in classifying active/inactive variants for DHFR single amino acid substitutions (**Figure B.3.29**) and they also classified the full DHFR landscape worse than random (**Figure 3.4a–i**), which ultimately ablated the benefit of focused training (**Figure 3.4a–ii**). Finally, consider whether the search space is large (e.g., four-site libraries) and whether the screening budget allows for multiple rounds. A decision tree is provided to assist users in selecting the appropriate strategy (**Figure 3.5**).

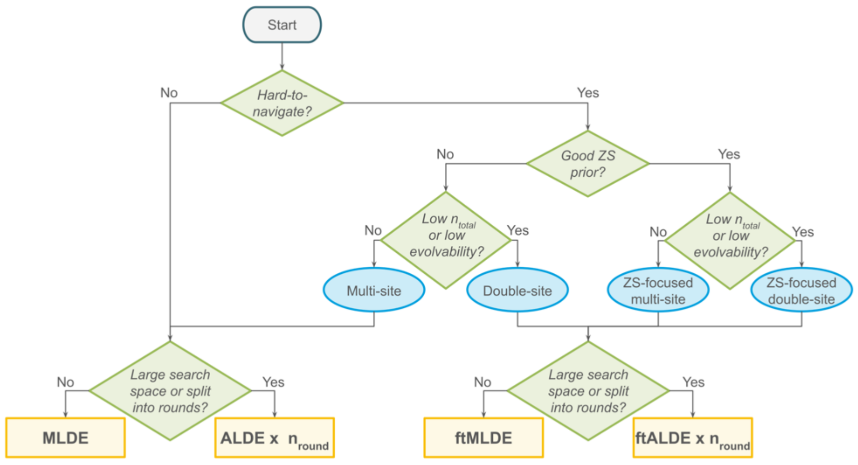


Figure 3.5. *Decision tree summarizing recommended ML strategies based on landscape navigability (e.g., percentage of active variants, pairwise epistasis), ZS prior (i.e., ROC-AUC > 0.5, evolution relevance), total number of variants screened experimentally (n_{total}) and the number of available screening rounds (n_{round}). We define a landscape as "hard-to-navigate" if the average pairwise C-alpha distance of substituted residues was ≤10 Å*[72] *or if the sites were in an enzyme active site. A "good ZS prior" was defined as using EVmutation if the MSA contained more than 1,000 sequences, using ESM-IF for binding interactions, and having no good ZS prior otherwise. Low n_{total} was defined as $n_{total} \leq 480$, for which ZS predictors were ensembled with Hamming distance. A "large search space" was defined as a landscape targeting four or more sites (**Appendix B.2 Methods**).*

Following this decision tree (**Figure 3.5**), we mimicked a prospective decision-making process to select an ML strategy and ZS for a given landscape. For a total of 480 unique variants screened, we would reach 0.93 ± 0.15 average maximum fitness and 0.75 ± 0.42 fraction reaching the global optimal, averaged across all 16 landscapes (**Table B.3.12**; **Appendix B.2 Methods**). For a total of 2016 unique variants screened, we would reach 0.97 ± 0.12 and 0.85 ± 0.32, respectively (**Table B.3.13**; **Appendix B.2 Methods**). Although our analyses provide valuable insights about strategy selection, most decisions in this tree were retrospective in nature and may not fully capture campaign-specific nuances. The real test of generalization lies in applying these approaches to a broader range of landscapes and, ultimately, to real-world DE campaigns.

We focused this study on combinatorial landscapes typically generated using SSM, which are often enriched in epistasis and present challenges for DE. In contrast, random mutations spread across a protein generally exhibit little epistasis, and thus beneficial mutations can often be combined to generate variants with higher fitness with great success using laboratory methods such as staggered extension process (StEP) recombination.[73,74] Hence, we expect simple additive models to be competitive with more complicated frameworks. Hamming distance has been demonstrated to have a weak correlation with variant fitness in this context.[23] Similarly, the ZS predictor benchmark has also been performed predominantly on datasets with random mutations spread across a protein.[25] We expect our conclusions would generalize to other few-site studies and alternative active-learning frameworks.[75] However, identifying critical target sites in an entirely new system remains a challenge; identification could benefit from computational tools but often requires some initial experimental screening.

We also evaluated several additional design choices for ML strategies that had a more limited impact on MLDE performance. First, we explored more informative ways to represent protein sequences compared to a categorical encoding (one-hot, which has no learned information and treats all amino acids equally). Learned representations from PLMs (e.g., frozen ESM-2 embeddings) showed minimal to no improvement over one-hot encoding for landscapes with at least 1% active variants (**Figure B.3.31**), including in the focused training setting (**Figure B.3.32**). However, they did exhibit improvements for landscapes with fewer than 1% active variants (**Figure B.3.33**). While not as beneficial as focused training (**Figure B.3.11**), learned representations may still enhance performance for particularly challenging landscapes when combined with focused training (**Figure B.3.34**). To leverage learned representations beyond frozen embeddings and assess the impact of focused training on fine-tuning, we fine-tuned ESM-2 with either randomly sampled or ZS-guided variants. To mitigate overfitting risks,[62] we adopted Low Rank Adaptation (LoRA), a parameter-efficient approach effective on diverse, non-combinatorial datasets.[76] While LoRA fine-tuning with randomly sampled variants failed to outperform MLDE, focused training-based fine-tuning outperformed both fine-tuning and MLDE with randomly sampled variants (**Figure B.3.35**). For challenging landscapes with fewer than 1% active variants, focused training-based fine-tuning was among the most effective strategies, alongside ftMLDE and ftALDE (**Figure B.3.36**). More advanced strategies like meta-transfer learning or learning to rank could further enhance performance, especially with limited training data.[77] Additionally, we used boosting models to facilitate a direct comparison between MLDE and ALDE. Different model choices and ensembles, such as ridge regression for MLDE or deep neural network ensembles for ALDE (**Figures B.3.37–B.3.38**), or hyperparameter tuning, could offer further improvements. Meanwhile, upper confidence bound and Thompson sampling acquisition functions performed similarly to the greedy acquisition function for ALDE (**Figures B.3.39–B.3.40**). For MLDE, we used top 96 variants for evaluation across all training sizes. However, different splits between training and testing variants could be explored to achieve more optimal outcomes given a fixed total number of variants. We provide a codebase, SSMuLA (Site-Saturation Mutagenesis Landscape Assistant), which includes options for these granular design choices. While MLDE and ALDE can be run on CPUs, fine-tuning is much more computationally expensive and less feasible without GPUs. ZS predictions vary in runtime, ranging from seconds to a few hours (**Appendix B.2 Methods**). The computational cost scales linearly with number of rounds or number of samples for training and exponentially with number of sites for evaluation. For larger search spaces, more efficient methods such as generative modeling may be necessary to reduce computational overhead and accelerate exploration.

While we streamlined focused training design choices, we also identified areas for improvement. Based on testing individual ZS predictors across different thresholds of the original search space, we set the ZS-focused library threshold to the top 12.5%, ranked by ZS scores, across all landscapes (**Figure B.3.41**). We then naively ensembled ZS predictors to demonstrate the benefits of combining orthogonal priors (**Appendix B.2 Methods**). The current Hamming distance ensembled ZS-focused libraries inherently had a size cutoff (i.e., 12.5% of a double-site library on a four-site landscape is 300). More sophisticated approaches, such as MODIFY,[78] which balance ZS selection with training set diversity and manage the exploration-exploitation trade-off, may offer a more comprehensive and autonomous method for selecting and ensembling ZS predictors.

There are also signs that ZS predictors are intrinsically limited for certain prediction tasks. For instance, the enabling K227 substitution in TrpB, which enables high-fitness variants under engineering conditions but is nearly undetectable in natural sequences, might not be adequately captured by EVmutation.[45] Additionally, the performance of different ZS predictors varied even within the same protein, as observed in TrpB across different landscapes. This indicates that while natural evolutionary information can be predictive, it may fall short in capturing evolution in the laboratory. Furthermore, all the enzyme systems we studied primarily involved native or near-native functions, with a majority being TrpB landscapes. When applied to non-native functions, we expect that the usefulness of evolutionary-based ZS predictors will decrease, perhaps significantly so. Enzyme activities involve intricate catalytic mechanisms, including substrate recognition, transition state stabilization, and conformational changes.[79] Some of the enzymes surveyed also involve multi-step reactions.[80] These complexities make enzyme activities inherently more challenging to predict than binding interactions, which are often dominated by non-covalent intermolecular interactions. In the case of the TEV and T7 landscapes, none of the ZS predictors consistently improved focused training performance from random sampling, suggesting room for the development of new ZS predictors.

We selected ZS predictors to cover diverse modalities and methodologies, rather than focusing on exhaustiveness or the latest models. Benchmarks such as ProteinGym have evaluated a wealth of models, and new ZS predictors continue to emerge rapidly.[25] Future work could explore hybrid and novel approaches. For example, TranceptEVE integrates family-specific alignments with a family-agnostic PLM,[81] and ProSST and SaProt incorporate structural features distinct from inverse folding models that learn sequence distributions conditional on an input structure.[82,83] Even within a single category like PLM-based ZS predictors, differences in architecture and methodology exist. Unlike ESM-2, which uses masked language modeling objective,[26] ProGen uses an autoregressive approach to predict amino acids sequentially,[84] while PoET employs retrieval-augmented language modeling to incorporate external contextual information.[85] These distinctions may affect performances, warranting further investigation to determine their relative advantages in different contexts.

In summary, our study lays the groundwork for a ML-assisted protein engineering framework leveraging the strengths of multiple ML approaches, including MLDE, ftMLDE, ALDE, and, introduced here, ftALDE. With our growing ability to read[86,87] and write sequences,[88] along with improved tools for constructing libraries,[89] we believe our findings will demystify the application of ML-based DE strategies and encourage their broader adoption in protein engineering.

3.4 Bibliography for Chapter 3

1. Lutz, S. & Iamurri, S. M. Protein engineering: Past, present, and future. *Methods Mol. Biol. Clifton NJ* **1685**, 1–12 (2018).
2. Arnold, F. H. Directed evolution: Bringing new chemistry to life. *Angew. Chem. Int. Ed.* **57**, 4143–4148 (2018).
3. Packer, M. S. & Liu, D. R. Methods for the directed evolution of proteins. *Nat. Rev. Genet.* **16**, 379–394 (2015).
4. Wang, Y. *et al.* Directed evolution: Methodologies and applications. *Chem. Rev.* **121**, 12384–12444 (2021).
5. Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat. Rev. Mol. Cell Biol.* **10**, 866–876 (2009).
6. Maynard Smith, J. Natural selection and the concept of a protein space. *Nature* **225**, 563–564 (1970).
7. Starr, T. N. & Thornton, J. W. Epistasis in protein evolution. *Protein Sci. Publ. Protein Soc.* **25**, 1204 (2016).
8. Miton, C. M., Buda, K. & Tokuriki, N. Epistasis and intramolecular networks in protein evolution. *Curr. Opin. Struct. Biol.* **69**, 160–168 (2021).
9. Anishchenko, I., Ovchinnikov, S., Kamisetty, H. & Baker, D. Origins of coevolution between residues distant in protein 3D structures. *Proc. Natl. Acad. Sci.* **114**, 9122–9127 (2017).
10. Bell, E. L. *et al.* Biocatalysis. *Nat. Rev. Methods Primer* **1**, 1–21 (2021).
11. Wu, Z., Kan, S. B. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci.* **116**, 8852–8858 (2019).
12. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* (2021) doi:10.1016/j.cels.2021.07.008.
13. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).

14. Johnston, K. E. *et al.* Machine learning for protein engineering. *ArXiv* arXiv:2305.16634v1 (2023).

15. Qiu, Y., Hu, J. & Wei, G.-W. Cluster learning-assisted directed evolution. *Nat. Comput. Sci.* **1**, 809–818 (2021).

16. Qiu, Y. & Wei, G.-W. CLADE 2.0: Evolution-driven cluster learning-assisted directed evolution. *J. Chem. Inf. Model.* **62**, 4629–4641 (2022).

17. Yang, J. *et al.* Active learning-assisted directed evolution. *Nat. Commun.* **16**, 714 (2025).

18. Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).

19. Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. in *Advances in Neural Information Processing Systems* vol. 34 29287–29303 (Curran Associates, Inc., 2021).

20. Hsu, C. *et al.* Learning inverse folding from millions of predicted structures. Preprint at https://doi.org/10.1101/2022.04.10.487779 (2022).

21. Ding, D. *et al.* Protein design using structure-based residue preferences. *Nat. Commun.* **15**, 1639 (2024).

22. Bloom, J. D., Labthavikul, S. T., Otey, C. R. & Arnold, F. H. Protein stability promotes evolvability. *Proc. Natl. Acad. Sci.* **103**, 5869–5874 (2006).

23. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* 1–9 (2022) doi:10.1038/s41587-021-01146-5.

24. Yang, K. K., Zanichelli, N. & Yeh, H. Masked inverse folding with sequence transfer for protein representation learning. *Protein Eng. Des. Sel.* **36**, gzad015 (2023).

25. Notin, P. *et al.* ProteinGym: Large-scale benchmarks for protein fitness prediction and design. *Adv. Neural Inf. Process. Syst.* **36**, 64331–64379 (2023).

26. Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).

27. Bedbrook, C. N., Yang, K. K., Rice, A. J., Gradinaru, V. & Arnold, F. H. Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic

expression and plasma membrane localization. *PLOS Comput. Biol.* **13**, e1005786 (2017).

28. Bedbrook, C. N. *et al.* Machine learning-guided channelrhodopsin engineering enables minimally invasive optogenetics. *Nat. Methods* **16**, 1176–1184 (2019).
29. Thomas, N. *et al.* Engineering highly active nuclease enzymes with machine learning and high-throughput screening. *Cell Syst.* **16**, (2025).
30. Rapp, J. T., Bremer, B. J. & Romero, P. A. Self-driving laboratories to autonomously navigate the protein fitness landscape. *Nat. Chem. Eng.* **1**, 97–107 (2024).
31. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl. Acad. Sci.* **110**, E193–E201 (2013).
32. Bryant, D. H. *et al.* Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39**, 691–696 (2021).
33. Freschlin, C. R., Fahlberg, S. A. & Romero, P. A. Machine learning to navigate fitness landscapes for protein engineering. *Curr. Opin. Biotechnol.* **75**, 102713 (2022).
34. Yang, J., Li, F.-Z. & Arnold, F. H. Opportunities and Challenges for Machine Learning-Assisted Enzyme Engineering. *ACS Cent. Sci.* **10**, 226–241 (2024).
35. Angermueller, C. *et al.* High-throughput ML-guided design of diverse single-domain antibodies against SARS-CoV-2. 2023.12.01.569227 Preprint at https://doi.org/10.1101/2023.12.01.569227 (2023).
36. Guo, J. *et al.* Computationally guided AAV engineering for enhanced gene delivery. *Trends Biochem. Sci.* **49**, 457–469 (2024).
37. Gelman, S. *et al.* Biophysics-based protein language models for protein engineering. 2024.03.15.585128 Preprint at https://doi.org/10.1101/2024.03.15.585128 (2024).
38. Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife* **5**, e16965 (2016).
39. Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).
40. Jalal, A. S. B. *et al.* Diversification of DNA-binding specificity by permissive and specificity-switching mutations in the ParB/Noc protein family. *Cell Rep.* **32**, 107928 (2020).

41. Lite, T.-L. V. *et al.* Uncovering the basis of protein-protein interaction specificity with a combinatorially complete library. *eLife* **9**, e60924 (2020).

42. Papkou, A., Garcia-Pastor, L., Escudero, J. A. & Wagner, A. A rugged yet easily navigable fitness landscape. *Science* **382**, eadh3860 (2023).

43. Tu, B., Sundar, V. & Esvelt, K. M. An ultra-high-throughput method for measuring biomolecular activities. Preprint at https://doi.org/10.1101/2022.03.09.483646 (2024).

44. Sundar, V., Tu, B., Guan, L. & Esvelt, K. FLIGHTED: Inferring fitness landscapes from noisy high-throughput experimental data. Preprint at https://doi.org/10.1101/2024.03.26.586797 (2024).

45. Johnston, K. E. *et al.* A combinatorially complete epistatic fitness landscape in an enzyme active site. *Proc. Natl. Acad. Sci.* **121**, e2400439121 (2024).

46. Van Cleve, J. & Weissman, D. B. Measuring ruggedness in fitness landscapes. *Proc. Natl. Acad. Sci.* **112**, 7345–7346 (2015).

47. de Visser, J. A. G. M. & Krug, J. Empirical fitness landscapes and the predictability of evolution. *Nat. Rev. Genet.* **15**, 480–490 (2014).

48. Szendro, I. G., Schenk, M. F., Franke, J., Krug, J. & Visser, J. A. G. M. de. Quantitative analyses of empirical fitness landscapes. *J. Stat. Mech. Theory Exp.* **2013**, P01005 (2013).

49. Aita, T. & Husimi, Y. Fitness spectrum among random mutants on Mt. Fuji-type fitness landscape. *J. Theor. Biol.* **182**, 469–485 (1996).

50. Kauffman, S. & Levin, S. Towards a general theory of adaptive walks on rugged landscapes. *J. Theor. Biol.* **128**, 11–45 (1987).

51. Kingman, J. F. C. A simple model for the balance between selection and mutation. *J. Appl. Probab.* **15**, 1–12 (1978).

52. Crona, K., Greene, D. & Barlow, M. The peaks and geometry of fitness landscapes. *J. Theor. Biol.* **317**, 1–10 (2013).

53. Kondrashov, D. A. & Kondrashov, F. A. Topological features of rugged fitness landscapes in sequence space. *Trends Genet.* **31**, 24–33 (2015).

54. Thomas, N., Agarwala, A., Belanger, D., Song, Y. S. & Colwell, L. Tuned fitness landscapes for benchmarking model-guided protein design. 2022.10.28.514293 Preprint at https://doi.org/10.1101/2022.10.28.514293 (2022).
55. Park, Y., Metzger, B. P. H. & Thornton, J. W. The simplicity of protein sequence-function relationships. *Nat. Commun.* **15**, 7953 (2024).
56. Chiang, W.-L. *et al.* Chatbot Arena: An open platform for evaluating LLMs by human preference. Preprint at https://doi.org/10.48550/arXiv.2403.04132 (2024).
57. Boubdir, M., Kim, E., Ermis, B., Hooker, S. & Fadaee, M. Elo uncovered: Robustness and best practices in language model evaluation. *Adv. Neural Inf. Process. Syst.* **37**, 106135–106161 (2024).
58. Triad. https://triad.protabit.com/.
59. Hopf, T. A. *et al.* The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35**, 1582–1584 (2019).
60. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci.* **118**, (2021).
61. Yang, K. K., Fusi, N. & Lu, A. X. Convolutions are competitive with transformers for protein sequence pretraining. *Cell Syst.* S2405471224000292 (2024) doi:10.1016/j.cels.2024.01.008.
62. Elnaggar, A. *et al.* ProtTrans: Towards cracking the language of life's code through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
63. Elnaggar, A. *et al.* Ankh: Optimized protein language model unlocks general-purpose modelling. Preprint at https://doi.org/10.48550/arXiv.2301.06568 (2023).
64. Ofer, D., Brandes, N. & Linial, M. The language of proteins: NLP, machine learning & protein sequences. *Comput. Struct. Biotechnol. J.* **19**, 1750–1758 (2021).
65. Bepler, T. & Berger, B. Learning the protein language: Evolution, structure, and function. *Cell Syst.* **12**, 654-669.e3 (2021).
66. Johnson, S. R. *et al.* Computational scoring and experimental evaluation of enzymes generated by neural networks. *Nat. Biotechnol.* 1–10 (2024) doi:10.1038/s41587-024-02214-2.

67. Lu, W., Zhang, J., Rao, J., Zhang, Z. & Zheng, S. AlphaFold3, a secret sauce for predicting mutational effects on protein-protein interactions. 2024.05.25.595871 Preprint at https://doi.org/10.1101/2024.05.25.595871 (2024).
68. Li, F.-Z., Amini, A. P., Yue, Y., Yang, K. K. & Lu, A. X. Feature reuse and scaling: Understanding transfer learning with protein language models. in *Proceedings of the 41st International Conference on Machine Learning* 27351–27375 (PMLR, 2024).
69. Chowdhury, R. *et al.* Single-sequence protein structure prediction using a language model and deep learning. *Nat. Biotechnol.* **40**, 1617–1623 (2022).
70. Kille, S. *et al.* Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. *ACS Synth. Biol.* **2**, 83–92 (2013).
71. Duarte, J. M., Sathyapriya, R., Stehr, H., Filippis, I. & Lappe, M. Optimal contact definition for reconstruction of Contact Maps. *BMC Bioinformatics* **11**, 283 (2010).
72. Zhao, H., Giver, L., Shao, Z., Affholter, J. A. & Arnold, F. H. Molecular evolution by staggered extension process (StEP) in vitro recombination. *Nat. Biotechnol.* **16**, 258–261 (1998).
73. Almhjell, P. J. *et al.* The β-subunit of tryptophan synthase is a latent tyrosine synthase. *Nat. Chem. Biol.* **20**, 1086–1093 (2024).
74. Vornholt, T. *et al.* Enhanced sequence-activity mapping and evolution of artificial metalloenzymes by active learning. *ACS Cent. Sci.* (2024) doi:10.1021/acscentsci.4c00258.
75. Schmirler, R., Heinzinger, M. & Rost, B. Fine-tuning protein language models boosts predictions across diverse tasks. *Nat. Commun.* **15**, 7407 (2024).
76. Zhou, Z. *et al.* Enhancing efficiency of protein language models with minimal wet-lab data through few-shot learning. *Nat. Commun.* **15**, 5566 (2024).
77. Ding, K. *et al.* Machine learning-guided co-optimization of fitness and diversity facilitates combinatorial library design in enzyme engineering. *Nat. Commun.* **15**, 6392 (2024).
78. Nam, K., Shao, Y., Major, D. T. & Wolf-Watz, M. Perspectives on computational enzyme modeling: From mechanisms to design and drug development. *ACS Omega* **9**, 7393–7412 (2024).

79. Buller, A. R. *et al.* Directed evolution of the tryptophan synthase β-subunit for stand-alone function recapitulates allosteric activation. *Proc. Natl. Acad. Sci.* **112**, 14599–14604 (2015).

80. Notin, P. *et al.* Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. in *Proceedings of the 39th International Conference on Machine Learning* 16990–17017 (PMLR, 2022).

81. Li, M. *et al.* ProSST: Protein language modeling with quantized structure and disentangled attention. *Adv. Neural Inf. Process. Syst.* **37**, 35700–35726 (2024).

82. Su, J. *et al.* SaProt: Protein language modeling with structure-aware vocabulary. 2023.10.01.560349 Preprint at https://doi.org/10.1101/2023.10.01.560349 (2024).

83. Nijkamp, E., Ruffolo, J. A., Weinstein, E. N., Naik, N. & Madani, A. ProGen2: Exploring the boundaries of protein language models. *Cell Syst.* **14**, 968-978.e3 (2023).

84. Truong Jr, T. & Bepler, T. PoET: A generative model of protein families as sequences-of-sequences. *Adv. Neural Inf. Process. Syst.* **36**, 77379–77415 (2023).

85. Wittmann, B. J., Johnston, K. E., Almhjell, P. J. & Arnold, F. H. evSeq: Cost-effective amplicon sequencing of every variant in a protein library. *ACS Synth. Biol.* (2022) doi:10.1021/acssynbio.1c00592.

86. Long, Y. *et al.* LevSeq: Rapid generation of sequence-function data for directed evolution and machine learning. *ACS Synth. Biol.* **14**, 230–238 (2025).

87. Hoose, A., Vellacott, R., Storch, M., Freemont, P. S. & Ryadnov, M. G. DNA synthesis technologies to close the gene writing gap. *Nat. Rev. Chem.* **7**, 144–161 (2023).

88. Yang, J. *et al.* DeCOIL: Optimization of degenerate codon libraries for machine learning-assisted protein engineering. *ACS Synth. Biol.* **12**, 2444–2454 (2023).

89. Virtanen, P. *et al.* SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272 (2020).

90. Berman, H. M. *et al.* The protein data bank. *Nucleic Acids Res.* **28**, 235–242 (2000).

91. Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L. & Dror, R. Learning from protein structure with geometric vector perceptrons. Preprint at https://doi.org/10.48550/arXiv.2009.01411 (2021).

92. Pines, G., Pines, A. & Eckert, C. A. Highly efficient libraries design for saturation mutagenesis. *Synth. Biol.* **7**, ysac006 (2022).

93. Chen, T. & Guestrin, C. XGBoost: A scalable tree boosting system. in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 785–794 (Association for Computing Machinery, New York, NY, USA, 2016). doi:10.1145/2939672.2939785.

94. Buitinck, L. *et al.* API design for machine learning software: experiences from the scikit-learn project. Preprint at https://doi.org/10.48550/arXiv.1309.0238 (2013).

95. Balandat, M. *et al.* BoTorch: A framework for Efficient Monte-Carlo Bayesian Optimization. Preprint at https://doi.org/10.48550/arXiv.1910.06403 (2020).

96. Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q. & Wilson, A. G. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. Preprint at https://doi.org/10.48550/arXiv.1809.11165 (2021).

97. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. Preprint at https://doi.org/10.48550/arXiv.1912.01703 (2019).

Chapter 4

SUBSTRATE-AWARE ZERO-SHOT PREDICTORS FOR NON-NATIVE ENZYME ACTIVITIES

Material from this chapter appears in: "**Li, F.-Z.**, Radtke, L. A., Johnston, K. E., Liu, C.-H., Yue, Y. & Arnold, F. H. Substrate-aware zero-shot predictors for non-native enzyme activities. *GEM Workshop, ICLR* (2025)."

ABSTRACT

Enzymes can be engineered to catalyze reactions with non-native substrates or even perform entirely new reactions unknown in nature. However, developing such novel activities through wet-lab engineering is time- and resource-intensive. By estimating enzyme activity without new experimental data, zero-shot (ZS) predictors can accelerate enzyme engineering. While ZS predictors have been demonstrated in various contexts, they have yet to be evaluated on non-native substrates and *new-to-nature* chemistry. Critically, many existing predictors do not explicitly encode substrate or transition-state properties, which we propose are essential for predicting *new-to-nature* chemistry. Here, we systematically studied two types of mechanistically distinct enzymes using 16 ZS predictors—including six general and ten substrate-aware scores derived from generative modeling, molecular docking, and active-site heuristics. We curated new experimental and literature mutation datasets spanning 11 non-native substrates and three *new-to-nature* reactions with 11 additional substrates. The six general ZS predictors could generalize to non-native substrates but failed for *new-to-nature* chemistries. In contrast, certain substrate-aware approaches could predict *new-to-nature* chemistries, with AlphaFold 3's chain-predicted aligned error being the most predictive of both activity and stereoselectivity. A weighted ensemble of AlphaFold 3 and EVmutation scores generalized to all chemistries that we tested. Our findings highlight the potential of ZS predictors to accelerate enzyme engineering, advancing the expansion of the chemical universe beyond nature's repertoire.

4.1 Introduction for Chapter 4

Enzymes, nature's catalysts, perform life-sustaining chemistry. Due to their exquisite specificity and selectivity, engineered enzymes have applications in therapeutics, bioremediation, and biocatalysis, where they can offer greener and more sustainable alternatives to conventional chemical methods.[1,2] Beyond merely enhancing their native functions, significant efforts have focused on expanding enzyme repertoires to catalyze reactions with non-native substrates—or even perform entirely new chemistries unknown in biological systems, termed *new-to-nature*.[3–6] The development of such enzymes often starts by identifying a promiscuous or side activity, which is then optimized for desired functions (termed "fitness", **Figure 4.1a**).[7,8] Fitness optimization typically employs directed evolution, a widely used method for accumulating beneficial mutations through iterative rounds of mutagenesis (to generate variants) and functional assessment by selection or screening.[9,10] However, this process is labor- and resource-intensive, particularly for challenging chemistries constrained by low-throughput data collection.

Emerging computational tools, especially machine learning (ML)-based methods, have shown promise in accelerating enzyme engineering, from starting point discovery, to *de novo* designs and fitness optimization.[11–18] A particularly appealing avenue is zero-shot (ZS) prediction: estimating variant fitness without relying on experimental data. ZS predictors leverage auxiliary information such as protein stability, evolutionary patterns, or structural features. These predictors have augmented supervised models to identify higher-fitness variants, guided experimental data collection for ML model training, and scored *in silico* designs for reinforcement learning or experimental validation.[19–24] Recent benchmarks highlight the broad applicability of ZS predictors.[25] However, these methods have yet to demonstrate capability across diverse enzyme-substrate pairs, particularly for out-of-distribution non-native substrates and *new-to-nature* chemistries. Few studies also assess the ZS predictability of the reaction product stereoselectivity, a key factor influencing their structural and functional properties.[26] Furthermore, existing enzyme-substrate datasets primarily focus on native or near-native activities and rarely address active-site mutations that are critical to enabling fundamentally new chemistries.[27,28] Most importantly, many existing ZS methods do not explicitly encode substrate or transition-state properties, which we hypothesize are essential for predicting *new-to-nature* chemistry.

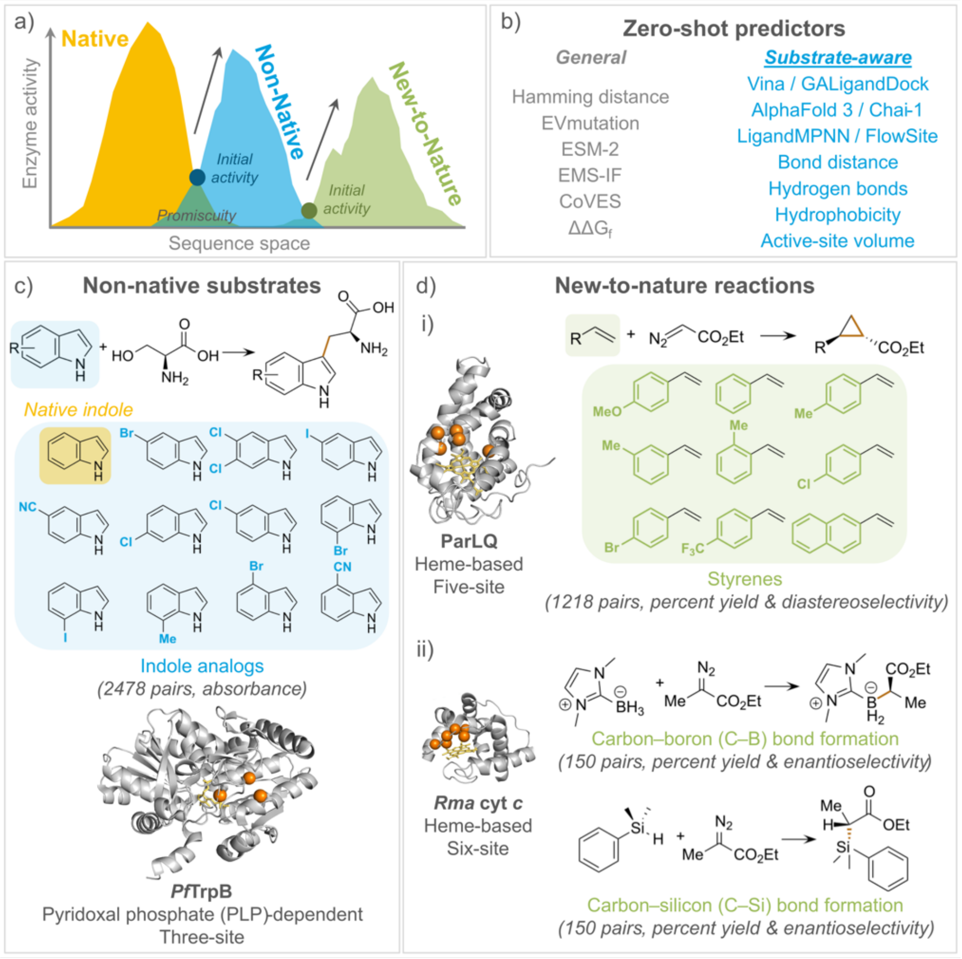


Figure 4.1. *Overview of datasets and zero-shot (ZS) predictors. a) Enzyme engineering for reactions with* nonnative *substrates and* new-to-nature *chemistries. b) Six general ZS predictors, covering different modalities and auxiliary information, alongside ten substrate-aware predictors derived from generative modeling, molecular docking, and active-site properties. c) The* Pf*TrpB reaction with the native substrate, indole, in a pyridoxal phosphate (PLP)-dependent manner, along with 11 non-native indole analogs. d) Heme-based new-to-nature carbene transfer reactions with activity and stereoselectivity labels: i) ParLQ: protoglobin-catalyzed olefin cyclopropanation with 9 styrenes and ii)* Rma *cyt* c*-mediated formation of C–B (*Rma*-CB) and C–Si (*Rma*-CSi) bonds. Crystal structures illustrate the cofactors and active-site residues targeted for engineering. The number of unique enzyme-substrate pairs is listed under each reaction.*

Here, we curated new experimental and literature datasets for two types of mechanistically distinct enzymes and benchmarked six general and ten newly proposed substrate-aware ZS predictors. Specifically, we evaluated their performance across 11 non-native substrates and three *new-to-nature* chemistries, covering 11 additional substrates (**Figure 4.1**). Each dataset includes active-site mutations designed to enhance specificity and selectivity, and introduce new chemistries.[6] Our study addresses three key questions: 1) Do general ZS predictors generalize to non-native substrates and *new-to-nature* chemistries for both activity and stereoselectivity? 2) Are novel substrate-aware ZS scores, derived from generative modeling, molecular docking, and active-site properties, more generalizable than the general predictors? 3) Which combination of ZS predictors can best generalize across these chemistries?

4.2 Methods and Datasets

We examined the predictability of ZS predictors on activity, defined by absorbance or percent yield of the major product. Where applicable, we also studied stereoselectivity, defined as enantiomeric excess of the desired chiral isomer or diastereomeric excess of the desired diastereomer. We analyzed the effects of mutations in key active-site residues across two types of mechanistically distinct enzymes: *Pf*TrpB, which catalyzes reactions with 11 non-native substrates, and heme-binding proteins (protoglobin and *Rma* cyt *c*), which catalyze three different *new-to-nature* chemistries (ParLQ, *Rma*-CB, *Rma*-CSi). In ParLQ, we further examined the effects of 9 different substrates. The *Pf*TrpB dataset is presented for the first time, while the rest were previously reported.[15,16] The 11 *Pf*TrpB datasets and ParLQ-a contain more low-activity variants than the other datasets (**Appendix C.4.1**). The similarity of the non-native substrate to the native substrate was calculated using the Tanimoto similarity of atom-pair fingerprints.[29] The energy barriers of new-to-nature chemistries were determined using density functional theory (DFT) calculations. We tested predictor ensembles using families of linear models. See **Appendix C.1** and **Appendix C.3** for more details. Spearman's correlation is reported in the main text. For recall (true positives of the top 25% ranked variants) and additional results, see **Appendix C.4.2** and **C.4.3**. Data supporting this study are deposited on Zenodo: *https://zenodo.org/records/15226690*. The code is available on GitHub: *https://github.com/fhalab/substrate_aware_zs*.

4.3 Results

4.3.1 General ZS predictors are predictive of activates on non-native substrates but do not generalize to new-to-nature chemistries

As baselines, we evaluated six general ZS predictors for variant activity scoring. Each ZS predictor leveraged distinct auxiliary information (**Figure 4.2a**). Hamming distance assumes that most mutations are deleterious.[3,30] It counts the number of amino acid substitutions from the parent, a variant with initial activity, aiming to perform a local search for viable variants. Given similar reaction mechanisms and conserved catalytic residues, we expected the assumption to hold true for substrates similar to their native counterparts. However, new-to-nature chemistry may require exploring a more diverse sequence space. Indeed, we observed a weak positive correlation for non-native activities ($\rho \sim 0.3$), but very little to weak negative correlation for the new-to-nature reactions ($\rho \sim -0.2 – 0.1$).

EVmutation and ESM-2 estimate mutation effects using evolutionary patterns, either through a Potts model applied to multiple-sequence alignments (MSAs) or a mask-prediction protein language models.[31,32] Both predictors generalized well to non-native substrates ($\rho \sim 0.5$) but not to new-to-nature chemistries ($\rho \sim -0.2 – 0.2$). Notably, their predictability decreased with shallower MSAs (**Table C6**) and was statistically correlated with substrate similarity to the native substrate ($p < 0.05$, **Figure 4.2c**). ESM-IF and CoVES incorporate structural context for ZS predictions. ESM-IF assigns residue likelihoods based on a backbone structure (inverse folding) and CoVES predicts masked residues based on their local atomic environments.[33,34] However, neither predictor outperformed EVmutation or ESM-2, though CoVES scores exhibited a weaker correlation with substrate similarity to the native substrate (**Table C10**).

Stability is crucial for protein function, as misfolded proteins will be less likely to retain activity.[15,35] We estimated variant stability by calculating the change in its free energy of folding relative to the parent ($\Delta\Delta G_f$) using a Rosetta energy function.[19] However, stability did not predict new-to-nature activities ($\rho \sim -0.1 – 0.1$). We reason that once a minimal stability threshold is met, factors such as substrate recognition and transition-state stabilization become the dominant determinants of activity. Moreover, excessive stability may limit the structural flexibility needed to accommodate new substrates or reaction mechanisms.[36]

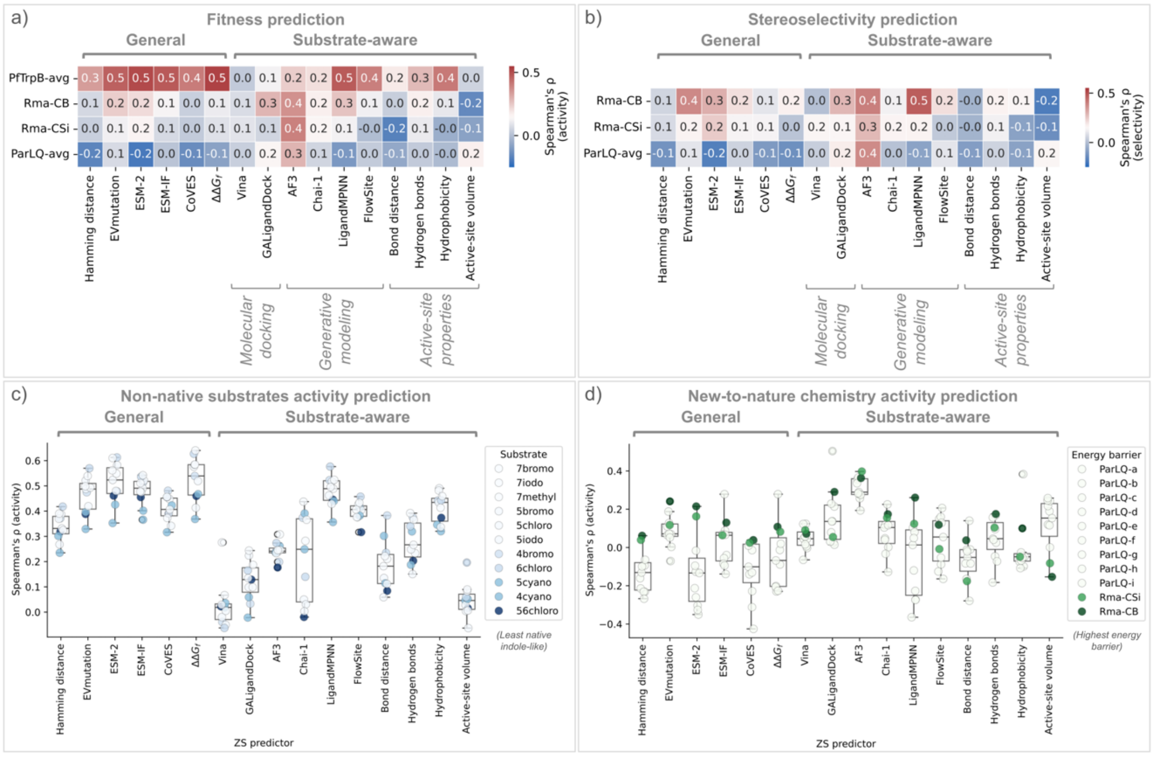


Figure 4.2. *General and substrate-aware ZS predictors. Spearman's ρ for a) activity, b) stereoselectivity, c)* Pf*TrpB non-native substrate activities (**Table C10**), and d) heme-based new-to-nature activities (**Table C11**).*

For the new-to-nature chemistries, we also evaluated the stereoselectivity of the major products (**Figure 4.2b**). For the general ZS predictors, activity and stereoselectivity predictions were generally correlated ($\rho \sim 0.68 - 0.95$, **Table C7**). The chemical mechanisms for the non-native substrates in this study are conserved, thus we postulate that this makes the predictions easier to generalize (**Figure C1**). In contrast, new-to-nature chemistry involves mechanisms distinct from an enzyme's native chemistry and may require beneficial mutations that are rare in MSAs or occur at conserved residues, thus demanding deeper structural insights into the substrate and/or the active site.

4.3.2 Substrate-aware predictors offer insights for new-to-nature chemistries

Enzyme catalysis involves complex mechanistic steps, including substrate binding and transition-state stabilization. We hypothesize that substrate-aware ZS predictors can better describe substrate recognition and transition-state stabilization for more diverse molecular systems, enhancing the predictability of new-to-nature chemistries. To capture the full enzyme-substrate-cofactor interaction, we considered the cofactors in their catalytically relevant states for each ZS predictor (**Appendix C.2**).

We first explored enzyme-substrate binding energy as a potential ZS score using physics-based molecular docking with GALigandDock and AutoDock Vina.[37,38] Both methods had weak to no correlation on non-native substrates ($\rho \sim 0 - 0.3$), but we noted GALigandDock was slightly better ($\Delta\rho \sim 0.1 - 0.2$) than AutoDock Vina for new-to-nature chemistries (**Figure 4.2a**). Correlations of both scores were independent of substrates (**Table C10**).

Recent advances in generative modeling have significantly advanced molecular docking and structure prediction.[39–41] We hypothesized that the scores pertaining to enzyme-substrate/cofactor binding may predict interactions impacting activity. While not outperforming general ZS predictors for non-native substrates, AlphaFold 3 (AF3) and Chai-1's interface predicted TM-scores (iPTM) for the enzyme-cofactor were predictive of new-to-nature activities (**Figure 4.2a**). Interestingly, AF3's chain-predicted aligned error (PAE) for the enzyme-cofactor interaction was the most predictive for both activity and stereoselectivity ($\rho \sim 0.4$), independent of the substrate (**Figure 4.2**). In contrast, despite adopting similar algorithmic approaches but without MSAs or templates, Chai-1 exhibited lower predictability, particularly for substrates more dissimilar to the native one (**Figure 4.2c**, **Table C10**). While MSA quality may impact prediction accuracy, further investigation is needed. Generative models can also facilitate binding site design via substrate-aware inverse folding or simultaneous docking and backbone redesign.[42–45] We studied using probability scores from variant generation, conditioned on the parent structure, as a ZS predictor. LigandMPNN and FlowSite were predictive of non-native reactions, performing comparably to existing predictors like ESM-2, but were less effective for new-to-nature chemistries other than *Rma*-CB.

Beyond docking scores, we reasoned that a docked pose can be distilled into key components that reflect enzyme-substrate interactions. We hypothesized that bond-forming atom proximity could indicate higher activity–for instance, the distance between Glu104 and N1-hydrogen in *Pf*TrpB, or between the carbene carbon and boron, silicon, or the styrene double bond (**Appendix C.2**). However, bond distance was a poor predictor, likely due to noise in docking poses. Stabilization forces, particularly hydrogen bonding, can lower reaction energy barriers in the enzyme's active site.[46] In *Pf*TrpB, the number of active-site hydrogen bonds correlated with activity, though it was less evident for heme-based new-to-nature chemistries. Instead, the highly reactive carbene intermediate would be stabilized by the iron in the heme.[47] The combined hydrophobicity of the substrate and active site affects their interaction, with optimal binding occurring when their hydrophobicity levels match.[48,49] This offered some predictive power for *Pf*TrpB, but not for heme-based chemistries. Lastly, active-site volume has been linked to enzyme promiscuity,[50] but it showed little predictive power for non-native or new-to-nature chemistries. We reasoned this may only exclude overly large residues, while failing to account for exposed active sites in *Rma* cty *c* or the substrate tunnel in ParLQ.[51] Consistent with general ZS predictors, the predictability for activity in substrate-aware predictors generally correlated with its predictability for stereoselectivity ($\rho \sim 0.31 - 0.95$, **Table C7**). AF3 remained the best predictor for new-to-nature chemistries.

4.3.3 Predictors ensembles improve generalization across chemistries

We explored ensemble methods to improve generalization and found that many model and predictor combinations outperformed individual predictors (**Table C5**). The unweighted ensemble of EVmutation and AF3—the top two individual predictors averaged across all chemistries—was consistently predictive ($\rho \sim 0.3 - 0.5$). A learned linear combination of them achieved an average ρ of 0.39 across all chemistries in the test set (**Figure 4.3**). This generalization remained robust regardless the training dataset (**Figure C17**). EVmutation, ESM-IF and AF3 had the highest regression weights when averaged across all chemistries (**Figure C16**). However, incorporating ESM-IF, the third-best predictors averaged across all chemistries, into any combination did not further improve the generalization (**Figure C15**). Interestingly, EVmutation was one of the top predictors alongside AF3 in the top 25% recall analysis, whereas ESM-IF was not (**Figure C10**).

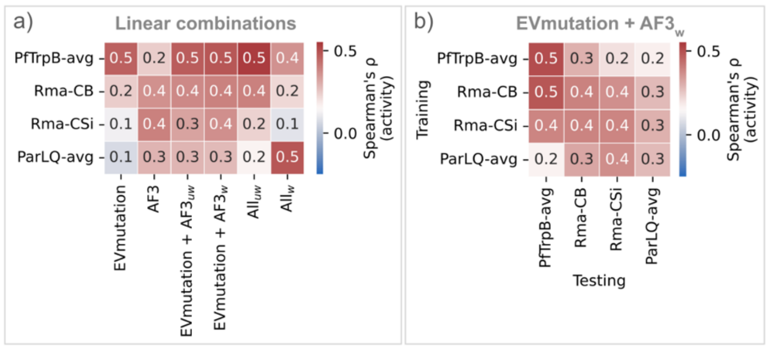


Figure 4.3. *Predictor ensembles. a) Different linear combinations of ZS predictors. w represents weighted linear combination trained on the* Rma-*CB dataset and tested on the rest. uw refers to an unweighted combination of EVmutation and AF3 rankings for each dataset. b) Linear models were trained on EVmutation and AF3 score for each dataset and tested on all the datasets.*

4.4 Discussion

We evaluated six general and ten substrate-aware ZS predictors using two types of enzymes with distinct mechanisms across 22 different substrates and four types of chemistries. General ZS predictors were effective for non-native substrates but failed for new-to-nature chemistries. Among substrate-aware ZS predictors, AF3 was the best for both activity and stereoselectivity prediction in new-to-nature chemistries. A linear combination of AF3 and EVmutation generalized across all studied reactions, which could complement current protein design pipelines that employ a series of logical filtering steps.[23,52] Physics-based docking methods and simple active-site heuristics did not consistently capture enzyme reactivity.

Enzyme engineering for new reactivities remains an out-of-distribution challenge, constrained by limited sequence-activity data. Although we generated new experimental data and curated literature datasets, their scope remains limited, especially since the literature datasets had largely active variants, which would not resemble distributions with mostly inactive variants. While the approaches studied here generalized well in active-site mutation datasets, expanding to more diverse new-to-nature reactions and testing datasets with mutations outside the active site remains a priority. With our growing ability to collect sequence-activity data,[53,54] more comprehensive datasets will be curated, and systematic benchmarks will be conducted. We are optimistic that increasingly generalizable substrate-aware ZS predictors will accelerate enzyme engineering, unlocking entirely new realms of biocatalysis.

4.5 Bibliography for Chapter 4

1. Buller, R. *et al.* From nature to industry: Harnessing enzymes for biocatalysis. *Science* **382**, eadh8615 (2023).
2. Lutz, S. & Iamurri, S. M. Protein engineering: Past, present, and future. *Methods Mol. Biol. Clifton NJ* **1685**, 1–12 (2018).
3. Arnold, F. H. Directed evolution: Bringing new chemistry to life. *Angew. Chem. Int. Ed.* **57**, 4143–4148 (2018).
4. Renata, H., Wang, Z. J. & Arnold, F. H. Expanding the enzyme universe: Accessing non-natural reactions by mechanism-guided directed evolution. *Angew. Chem. Int. Ed.* **54**, 3351–3367 (2015).
5. Chen, K. & Arnold, F. H. Engineering new catalytic activities in enzymes. *Nat. Catal.* **3**, 203–213 (2020).
6. Bell, E. L. *et al.* Biocatalysis. *Nat. Rev. Methods Primer* **1**, 1–21 (2021).
7. O'Brien, P. J. & Herschlag, D. Catalytic promiscuity and the evolution of new enzymatic activities. *Chem. Biol.* **6**, R91–R105 (1999).
8. Leveson-Gower, R. B., Mayer, C. & Roelfes, G. The importance of catalytic promiscuity for enzyme design and evolution. *Nat. Rev. Chem.* **3**, 687–705 (2019).
9. Packer, M. S. & Liu, D. R. Methods for the directed evolution of proteins. *Nat. Rev. Genet.* **16**, 379–394 (2015).
10. Wang, Y. *et al.* Directed evolution: Methodologies and applications. *Chem. Rev.* **121**, 12384–12444 (2021).
11. Yang, J., Li, F.-Z. & Arnold, F. H. Opportunities and challenges for machine learning-assisted enzyme engineering. *ACS Cent. Sci.* **10**, 226–241 (2024).
12. Mak, W. S. *et al.* Integrative genomic mining for enzyme function to enable engineering of a non-natural biosynthetic pathway. *Nat. Commun.* **6**, 10005 (2015).
13. Siegel, J. B. *et al.* Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction. *Science* **329**, 309–313 (2010).

14. Kalvet, I. *et al.* Design of heme enzymes with a tunable substrate binding pocket adjacent to an open metal coordination site. *J. Am. Chem. Soc.* **145**, 14307–14315 (2023).
15. Ding, K. *et al.* Machine learning-guided co-optimization of fitness and diversity facilitates combinatorial library design in enzyme engineering. *Nat. Commun.* **15**, 6392 (2024).
16. Yang, J. *et al.* Active learning-assisted directed evolution. *Nat. Commun.* **16**, 714 (2025).
17. Thomas, N. *et al.* Engineering highly active nuclease enzymes with machine learning and high-throughput screening. *Cell Syst.* **16**, (2025).
18. Rapp, J. T., Bremer, B. J. & Romero, P. A. Self-driving laboratories to autonomously navigate the protein fitness landscape. *Nat. Chem. Eng.* **1**, 97–107 (2024).
19. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026-1045.e7 (2021).
20. Li, F.-Z. *et al.* Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes. Preprint at https://doi.org/10.1101/2024.10.24.619774 (2024).
21. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* 1–9 (2022) doi:10.1038/s41587-021-01146-5.
22. Landwehr, G. M. *et al.* Accelerated enzyme engineering by machine-learning guided cell-free expression. *Nat. Commun.* **16**, 865 (2025).
23. Johnson, S. R. *et al.* Computational scoring and experimental evaluation of enzymes generated by neural networks. *Nat. Biotechnol.* 1–10 (2024) doi:10.1038/s41587-024-02214-2.
24. Stocco, F. *et al.* Guiding generative protein language models with reinforcement learning. Preprint at https://doi.org/10.48550/arXiv.2412.12979 (2024).
25. Notin, P. *et al.* ProteinGym: Large-scale benchmarks for protein fitness prediction and design. *Adv. Neural Inf. Process. Syst.* **36**, 64331–64379 (2023).

26. Reisenbauer, J. C., Sicinski, K. M. & Arnold, F. H. Catalyzing the future: recent advances in chemical synthesis using enzymes. *Curr. Opin. Chem. Biol.* **83**, 102536 (2024).
27. Goldman, S., Das, R., Yang, K. K. & Coley, C. W. Machine learning modeling of family wide enzyme-substrate specificity screens. *PLOS Comput. Biol.* **18**, e1009853 (2022).
28. Paton, A. *et al.* Generation of connections between protein sequence space and chemical space to enable a predictive model for biocatalysis. Preprint at https://doi.org/10.26434/chemrxiv-2024-w4dtr (2024).
29. Carhart, R. E., Smith, D. H. & Venkataraghavan, R. Atom pairs as molecular features in structure-activity studies: definition and applications. *J. Chem. Inf. Comput. Sci.* **25**, 64–73 (1985).
30. Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat. Rev. Mol. Cell Biol.* **10**, 866–876 (2009).
31. Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).
32. Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. in *Advances in Neural Information Processing Systems* vol. 34 29287–29303 (Curran Associates, Inc., 2021).
33. Hsu, C. *et al.* Learning inverse folding from millions of predicted structures. Preprint at https://doi.org/10.1101/2022.04.10.487779 (2022).
34. Ding, D. *et al.* Protein design using structure-based residue preferences. *Nat. Commun.* **15**, 1639 (2024).
35. Bloom, J. D., Labthavikul, S. T., Otey, C. R. & Arnold, F. H. Protein stability promotes evolvability. *Proc. Natl. Acad. Sci.* **103**, 5869–5874 (2006).
36. Teufl, M., Zajc, C. U. & Traxlmayr, M. W. Engineering strategies to overcome the stability–function trade-off in proteins. *ACS Synth. Biol.* **11**, 1030–1039 (2022).
37. Park, H., Zhou, G., Baek, M., Baker, D. & DiMaio, F. Force field optimization guided by small molecule crystal lattice data enables consistent sub-angstrom protein–ligand docking. *J. Chem. Theory Comput.* **17**, 2000–2010 (2021).

38. Eberhardt, J., Santos-Martins, D., Tillack, A. F. & Forli, S. AutoDock Vina 1.2.0: New docking methods, expanded force field, and Python bindings. *J. Chem. Inf. Model.* **61**, 3891–3898 (2021).

39. Yim, J. *et al.* Diffusion models in protein structure and docking. *WIREs Comput. Mol. Sci.* **14**, e1711 (2024).

40. Abramson, J. *et al.* Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630**, 493–500 (2024).

41. Discovery, C. *et al.* Chai-1: Decoding the molecular interactions of life. Preprint at https://doi.org/10.1101/2024.10.10.615955 (2024).

42. Krapp, L. F., Meireles, F. A., Abriata, L. A. & Peraro, M. D. Context-aware geometric deep learning for protein sequence design. Preprint at https://doi.org/10.1101/2023.06.19.545381 (2023).

43. Watson, J. L. *et al.* De novo design of protein structure and function with RFdiffusion. *Nature* **620**, 1–3 (2023) doi:10.1038/s41586-023-06415-8.

44. Dauparas, J. *et al.* Atomic context-conditioned protein sequence design using LigandMPNN. Preprint at https://doi.org/10.1101/2023.12.22.573103 (2023).

45. Stärk, H., Jing, B., Barzilay, R. & Jaakkola, T. Harmonic Self-conditioned flow matching for multi-ligand docking and binding site design. Preprint at https://doi.org/10.48550/arXiv.2310.05764 (2024).

46. Shan, S. & Herschlag, D. The change in hydrogen bond strength accompanying charge rearrangement: Implications for enzymatic catalysis. *Proc. Natl. Acad. Sci.* **93**, 14474–14479 (1996).

47. Chaturvedi, S. S., Vargas, S., Ajmera, P. & Alexandrova, A. N. Directed evolution of protoglobin optimizes the enzyme electric field. *J. Am. Chem. Soc.* **146**, 16670–16680 (2024).

48. Estell, D. A. *et al.* Probing steric and hydrophobic effects on enzyme-substrate interactions by protein engineering. *Science* **233**, 659–663 (1986).

49. Sriramulu, D. K. & Lee, S.-G. Combinatorial effect of ligand and ligand-binding site hydrophobicities on binding affinity. *J. Chem. Inf. Model.* **60**, 1678–1684 (2020).

50. Martínez-Martínez, M. *et al.* Determinants and prediction of esterase substrate promiscuity patterns. *ACS Chem. Biol.* **13**, 225–234 (2018).

51. Danelius, E., Porter, N. J., Unge, J., Arnold, F. H. & Gonen, T. MicroED structure of a protoglobin reactive carbene intermediate. *J. Am. Chem. Soc.* **145**, 7159–7165 (2023).

52. Bennett, N. *et al.* Improving de novo protein binder design with deep learning. Preprint at https://www.biorxiv.org/content/10.1101/2022.06.15.495993v1 (2022).

53. Long, Y. *et al.* LevSeq: Rapid generation of sequence-function data for directed evolution and machine learning. *ACS Synth. Biol.* **14**, 230–238 (2025).

54. Wittmann, B. J., Johnston, K. E., Almhjell, P. J. & Arnold, F. H. evSeq: Cost-effective amplicon sequencing of every variant in a protein library. *ACS Synth. Biol.* **11**, 1313–1324 (2022).

Chapter 5

CONCLUSION AND OUTLOOK

5.1 Introduction for Chapter 5

Motivated by the gap between current machine learning (ML) capabilities and the practical demands of experimental protein engineering, this thesis investigates how ML methods can generalize across protein fitness landscapes to support the design of proteins with enhanced or novel functions. It focuses on the core challenges of real-world campaigns: scarce assay-labeled data, epistatic and rugged combinatorial landscapes, and the need to extrapolate beyond natural biology. Through three core projects, the thesis systematically evaluates strategies spanning the ML pipeline—from sequence representation and model training to data-efficient learning and zero-shot prediction—with an emphasis on methods that generalize across diverse engineering scenarios.

These findings not only provide practical strategies for navigating realistic protein design challenges, but also identify specific failure modes in current ML approaches—offering signals for ML developers to address these shortcomings. Together, they lay the groundwork for more targeted, application-aware tools that align better with the needs of experimental workflows. The remainder of this chapter outlines future opportunities and challenges in assay-labeled data curation, benchmarking, frontier models, closed-loop experimentation, and biosecurity considerations.

5.2 Data consolidation and benchmarking initiatives

The insights in this thesis are made possible by recent advances in sequencing and screening technologies, which now enable high-throughput mapping of protein fitness landscapes (**Section 1.2.1**).[1,2] As more laboratories adopt these methods, they are generating an ever-growing diversity of protein-engineering datasets that span distinct applications, degrees of epistasis, and non-native activities. Curating these data will not only enable more rigorous benchmarking and deeper insights into method generalization, but will also accelerate algorithmic and architectural innovations and develop more capable and generalizable foundation models.

5.2.1 Sequence-function databases

To make emerging datasets accessible and actionable for both experimental and computational researchers, a unified effort to consolidating assay-labeled data into structured, shareable resources is essential. In the Arnold Lab, we are building an interactive sequence-fitness database with cross-institutional collaborators. The platform is designed to enable experimentalists to effortlessly deposit their data and visualize results within and across experiments, and to provide ML scientists access to rich, standardized datasets for model training and benchmarking. Additionally, well-structured metadata can offer valuable experimental context and tailored retrieval.

Although our current focus is on enzyme-catalyzed new-to-nature chemistry—where no comparable repository yet exists—we envision a centralized, community-maintained resource analogous to the Protein Data Bank for protein structure data or UniRef for sequence data. Such a database would link protein sequences to quantitative functional measurements, annotated with rich metadata and spanning multiple domains and applications.

5.2.2 Fitness prediction benchmarking

On the evaluation front, benchmark frameworks such as ProteinGym provide a structured suite for assessing fitness predictors.[3] Although its coverage of challenging real-world engineering scenarios—especially highly epistatic landscapes and non-native functions—remains limited, ProteinGym has garnered notable community engagement. Researchers continue to contribute new zero-shot predictors and supervised models, making it one of the few adopted evaluation frameworks and a promising foundation for a standardized benchmark platform in the field.

5.3 Emerging ML frontiers in protein engineering

Recent advances at the interface of machine learning and protein science reveal both limitations to overcome and opportunities to exploit. **Section 5.3.1** reviews how scaling behavior and dataset composition reveal limitations in current foundation models and highlights opportunities for improvement. **Section 5.3.2** surveys generative frameworks that combine natural-sequence priors with assay-labeled data to expand and optimize design space. **Sections 5.3.3** and **5.3.4** discuss the potential of multimodal and AI-agent-assisted approaches to improve protein design and engineering. Collectively, these avenues aim to capture the information in vast, heterogeneous datasets into practical design and engineering capabilities.

5.3.1 Foundation-model scaling and pretraining datasets

Protein language models (PLMs), protein sequence foundation models such as Evolutionary Scale Modeling (ESM) covered in **Chapter 2**, have become indispensable across applications. Inspired by natural language processing (NLP), developers have pursued ever-larger models; yet scaling-law studies show that performance gains taper once model size reaches the high-hundreds-of-millions to low-billions of parameters, suggesting that the "bigger-is-better" rule from NLP does not directly translate to proteins.[4–6] Uneven taxonomic representation in the pretraining data further biases likelihoods and fitness predictions toward over-sampled clades, sometimes at odds with engineering objectives such as thermostability.[7]

Fundamentally, improvement of such foundation models may depend less on parameter count alone and more on a combination of strategies, such as curating diverse, taxonomically balanced datasets and making model design choices better suited to the structure and sparsity of biological data. Metagenomic sequencing is now adding billions of previously unseen sequences,[8] promising richer signal but also raising new curation challenges. There is still a lack of principled guidelines for which sequence subsets, or which levels of taxonomic, structural, or functional diversity, most effectively improve a given downstream task. Addressing these gaps will require close collaboration among ML researchers, bioinformaticians, and protein engineers to assemble balanced training sets and benchmark models on tasks that mirror real engineering contexts.

5.3.2 Generative modeling for design and optimization

Advances in generative modeling are opening new avenues for protein design and engineering. First, generative models can propose diverse, high-quality starting points for optimization—offering a principled alternative to traditional approaches like mutagenesis or recombination, which are often limited to local searches.[9] Second, these models enable broad exploration of sequence space and can support fitness prediction when guided by zero-shot scoring or fine-tuned with assay-labeled data.[10–12] By capturing the underlying rules of natural proteins while incorporating functional signals, generative models offer a unified framework for both design and optimization—expanding the reach of protein engineering beyond what evolution or supervised ML alone can achieve.

5.3.3 Multimodal integration

The combination of sequence, structure, evolutionary context and nature language text descriptions has emerged as a powerful strategy for protein modeling. For example, leveraging both multiple sequence alignments and structure, predictors such as VenusREM[13] and S3F-MSA[14], consistently outperforms larger single-sequence models like ESM on zero-shot fitness prediction. In the context of generative modeling, efforts have included conditioning protein language models on protein structures or functional descriptors.[15] Future work may include a unified framework that can bridge sequence, structure, natural language, and assay data, and support transfer learning across tasks.

5.3.4 Agent-driven platforms

Recent advancements in AI-driven agent platforms are transforming protein engineering workflows by enabling autonomous research pipelines. For example, the Virtual Lab is an AI–human research framework that uses a team of specialized LLM agents—led by a principal investigator agent and guided by a human—for interdisciplinary scientific discovery. It operates through simulated team meetings and task delegation.[18] When applied to nanobody design against SARS-CoV-2 variants, the system generated 92 candidates through a pipeline integrating ESM, AlphaFold-Multimer, and Rosetta. Experimental validation confirmed several functional binders, including two with improved binding to recent variants, demonstrating the Virtual Lab's potential to accelerate real-world, cross-domain research.[18] Similar multi-agent systems, such as Google's AI Co-Scientist, extend this paradigm to hypothesis generation and experimental planning across diverse scientific domains.[19]

5.4 Closing the loop in the wet lab

Ultimately, the value of machine learning-assisted protein engineering lies not only in improved retrospective prediction accuracy or proven generalizability across benchmark performance, but in its ability to prospectively guide successful individual experimental outcomes. Closing the loop—by implementing model-driven predictions and designs in the laboratory—is essential for realizing the practical impact of computational methods.

5.4.1 Ring expansion case study

Building on the substrate-aware zero-shot predictors developed in **Chapter 4**, current work in the lab applies these tools to guide enzyme engineering for a new-to-nature reaction: enantioselective oxetane ring expansion. This transformation enables access to valuable five-membered tetrahydrofuran heterocycles—motifs prevalent in drugs and natural products—from strained oxetane precursors.[16,17] Achieving enantioselective control for oxetane ring expansion has remained elusive with traditional chemical synthesis. Inspired by the laboratory-evolved variant of heme-containing enzyme P450BM3, P411-AzetS, which catalyzes aziridine ring expansion with high enantioselectivity,[16] the goal is to engineer an enzyme capable of expanding oxetanes.

A variant of *Aeropyrum pernix* protoglobin (*Ape*Pgb) was recently identified with trace activity for oxetane ring expansion in our lab. Three rounds of site-saturation mutagenesis (SSM) at the active site improved the yield to 5%. However, the campaign stalled as the variant resisted further improvement over multiple rounds of site-saturation mutagenesis and error-prone mutagenesis. To overcome this, *in silico* SSM guided by the substrate-aware zero-shot predictors identified in **Chapter 4** was performed, and preliminary experiments have identified several yield-improving variants. Ongoing efforts focus on optimizing over a broader sequence space using sampling methods such as Markov chain Monte Carlo, guided by these predictors, to identify higher-performing variants for downstream experimental validation.

5.5 Community initiatives and open tournaments

Beyond individual lab efforts, non-profit and community-driven organizations are emerging to build a more integrated ecosystem that connects data generation, benchmarking, and experimental validation. For example, the Align Foundation, whose mission is to enable “predictable biology,” promotes open data sharing, standardized benchmarking, and automation platforms for reproducible, large-scale experimentation. Inspired by the success of the Critical Assessment of Structure Prediction (CASP) in advancing protein structure prediction, initiatives such as the Protein Engineering Tournament aspire to play a similar role for protein fitness prediction and design.[20] These efforts chart a promising path toward centralized, transparent, and collaborative infrastructure to accelerate both model development and real-world impact. As the synergy between machine learning and experimental protein engineering deepens, we are beginning to see the emergence of workshops—and, potentially soon, dedicated conferences—designed to bridge these communities and foster the interdisciplinary collaboration essential for innovation.

5.6 Biosecurity and ethical considerations

As machine learning becomes increasingly integrated into protein engineering, it introduces not only powerful capabilities but also potential dual-use risks. One recent study identified the theoretical potential for AI-generated protein sequences to evade current biosecurity screening measures.[21] Although the researchers proposed mitigation strategies, the actual severity of the threat remained unclear due to the lack of experimental validation. A follow-up study addressed this by introducing a testing, evaluation, validation, and verification (TEVV) framework using safe biological proxies to empirically assess these risks.[22] The study found that, while current models can generate structurally similar sequences, they are not yet capable of reliably producing functional proteins that escape detection. Importantly, it demonstrated that experimental validation of AI-assisted protein design risks is not only essential but also achievable, with the TEVV framework offering a blueprint for future studies.

These studies highlight the need for proactive, ongoing assessment of biosecurity safeguards as AI capabilities evolve. This includes the responsible development and rigorous stress-testing of foundation models, as well as the creation of standardized evaluation protocols. While many open questions remain, several organizations are already working to address them, such as the International Gene Synthesis Consortium and the International Biosecurity and Biosafety Initiative for Science. Meanwhile, broader initiatives like the AI Safety Institutes across the globe and the Frontier Model Forum aim to evaluate and mitigate risks from frontier AI models, including those relevant to biosecurity. Together, these organizational, regulatory, and governmental efforts form a growing ecosystem of safeguards—essential for ensuring the responsible development and safe deployment of AI-driven biological design.

5.7 Closing thoughts

Looking ahead, machine learning will increasingly shape the future of protein engineering—not only by accelerating optimization but by reimagining how we discover and design new proteins altogether. With advances in high-throughput sequencing, synthesis, and screening, we are approaching an era where rich, functionally labeled datasets—once scarce—can drive model development at scale. Generative models, especially when informed by assay data and mechanistic priors, promise to explore vast regions of sequence space unreachable by directed evolution or local search. Coupled with zero-shot predictors, these models can prioritize viable, high-function variants and navigate toward truly novel chemistries. As cloud labs and automated experimentation systems mature, ML-guided "design–build–test–learn" cycles will become increasingly integrated and autonomous. To ensure these powerful tools are used responsibly, technological safeguards and policy frameworks must evolve in parallel. Ultimately, by combining data-driven prediction with biophysical insight and experimental feedback, ML has the potential not only to accelerate protein engineering but to unlock fundamentally new functions—reshaping how we interact with biology and build with it.

5.8 Bibliography for Chapter 5

1. Long, Y. *et al.* LevSeq: Rapid Generation of Sequence-Function Data for Directed Evolution and Machine Learning. *ACS Synth. Biol.* **14**, 230–238 (2025).
2. Wittmann, B. J., Johnston, K. E., Almhjell, P. J. & Arnold, F. H. evSeq: Cost-Effective Amplicon Sequencing of Every Variant in a Protein Library. *ACS Synth. Biol.* **11**, 1313–1324 (2022).
3. Notin, P. *et al.* ProteinGym: Large-Scale Benchmarks for Protein Fitness Prediction and Design. *Adv. Neural Inf. Process. Syst.* **36**, 64331–64379 (2023).
4. Serrano, Y., Ciudad, Á. & Molina, A. Are Protein Language Models Compute Optimal? Preprint at https://doi.org/10.48550/arXiv.2406.07249 (2024).
5. Notin, P. Have We Hit the Scaling Wall for Protein Language Models? *Pascal Notin* https://pascalnotin.substack.com/p/have-we-hit-the-scaling-wall-for (2025).
6. Fournier, Q. *et al.* Protein Language Models: Is Scaling Necessary? 2024.09.23.614603 Preprint at https://doi.org/10.1101/2024.09.23.614603 (2024).
7. Ding, F. & Steinhardt, J. Protein language models are biased by unequal sequence sampling across the tree of life. 2024.03.07.584001 Preprint at https://doi.org/10.1101/2024.03.07.584001 (2024).
8. Richardson, L. *et al.* MGnify: the microbiome sequence data analysis resource in 2023. *Nucleic Acids Res.* **51**, D753–D759 (2023).
9. Yang, J., Li, F.-Z. & Arnold, F. H. Opportunities and Challenges for Machine Learning-Assisted Enzyme Engineering. *ACS Cent. Sci.* **10**, 226–241 (2024).
10. Stocco, F. *et al.* Guiding Generative Protein Language Models with Reinforcement Learning. Preprint at https://doi.org/10.48550/arXiv.2412.12979 (2024).
11. Widatalla, T., Rafailov, R. & Hie, B. Aligning protein generative models with experimental fitness via Direct Preference Optimization. Preprint at https://doi.org/10.1101/2024.05.20.595026 (2024).
12. Blalock, N., Seshadri, S., Babber, A., Fahlberg, S. & Romero, P. Functional Alignment of Protein Language Models via Reinforcement Learning with Experimental Feedback.

13. Tan, Y., Wang, R., Wu, B., Hong, L. & Zhou, B. Retrieval-Enhanced Mutation Mastery: Augmenting Zero-Shot Prediction of Protein Language Model. Preprint at https://doi.org/10.48550/arXiv.2410.21127 (2024).
14. Zhang, Z. *et al.* Multi-Scale Representation Learning for Protein Fitness Prediction. *Adv. Neural Inf. Process. Syst.* **37**, 101456–101473 (2024).
15. Hayes, T. *et al.* Simulating 500 million years of evolution with a language model. *Science* **0**, eads0018 (2025).
16. Miller, D. C., Lal, R. G., Marchetti, L. A. & Arnold, F. H. Biocatalytic One-Carbon Ring Expansion of Aziridines to Azetidines via a Highly Enantioselective [1,2]-Stevens Rearrangement. *J. Am. Chem. Soc.* **144**, 4739–4745 (2022).
17. Bull, J. A., Croft, R. A., Davis, O. A., Doran, R. & Morgan, K. F. Oxetanes: Recent Advances in Synthesis, Reactivity, and Medicinal Chemistry. *Chem. Rev.* **116**, 12150–12233 (2016).
18. Swanson, K., Wu, W., Bulaong, N. L., Pak, J. E. & Zou, J. The Virtual Lab: AI Agents Design New SARS-CoV-2 Nanobodies with Experimental Validation. 2024.11.11.623004 Preprint at https://doi.org/10.1101/2024.11.11.623004 (2024).
19. Gottweis, J. *et al.* Towards an AI co-scientist. Preprint at https://doi.org/10.48550/arXiv.2502.18864 (2025).
20. Armer, C. *et al.* The Protein Engineering Tournament: An Open Science Benchmark for Protein Modeling and Design. Preprint at http://arxiv.org/abs/2309.09955 (2023).
21. Wittmann, B. J. *et al.* Toward AI-Resilient Screening of Nucleic Acid Synthesis Orders: Process, Results, and Recommendations. 2024.12.02.626439 Preprint at https://doi.org/10.1101/2024.12.02.626439 (2024).
22. Ikonomova, S. P. *et al.* Experimental Evaluation of AI-Driven Protein Design Risks Using Safe Biological Proxies. 2025.05.15.654077 Preprint at https://doi.org/10.1101/2025.05.15.654077 (2025).

Appendix A

SUPPLEMENTARY INFORMATION FOR CHAPTER 2

A.1 Additional tables and figures

Table A.1.1. *Downstream functional and structural tasks.*

Task	Task type	Split type	n Train	n Val	n Test	Model type (n classes)
SS3 – CB513	Residue-level secondary structure	Minimal homology	8678	2170	513	PyTorch linear classifier (3)
SS3 – TS115	Residue-level secondary structure	Released in 2016, from 43 to 1085 residues	8678	2170	115	PyTorch linear classifier (3)
SS3 – CASP12	Residue-level secondary structure	CASP12 targets, mostly more than 400 residues	8678	2170	21	PyTorch linear classifier (3)
Thermostability	Global property	In-distribution	22335	2482	3134	Scikit-learn ridge regression
Subcellular localization (scl)	Global property	In-distribution	9503	1678	385	PyTorch linear classifier (10)
GB1 – sampled	Local property	In-distribution	6289	699	1745	Scikit-learn ridge regression
GB1 – low vs high	Local property	Out-of-distribution	4580	509	3644	Scikit-learn ridge regression
GB1 – two vs rest	Local property	Out-of-distribution (fewer training samples)	381	43	8309	Scikit-learn ridge regression
AAV – two vs many	Local property	Out-of-distribution	28626	3181	50776	Scikit-learn ridge regression
AAV – one vs many	Local property	Out-of-distribution (fewer training samples)	1053	117	81413	Scikit-learn ridge regression

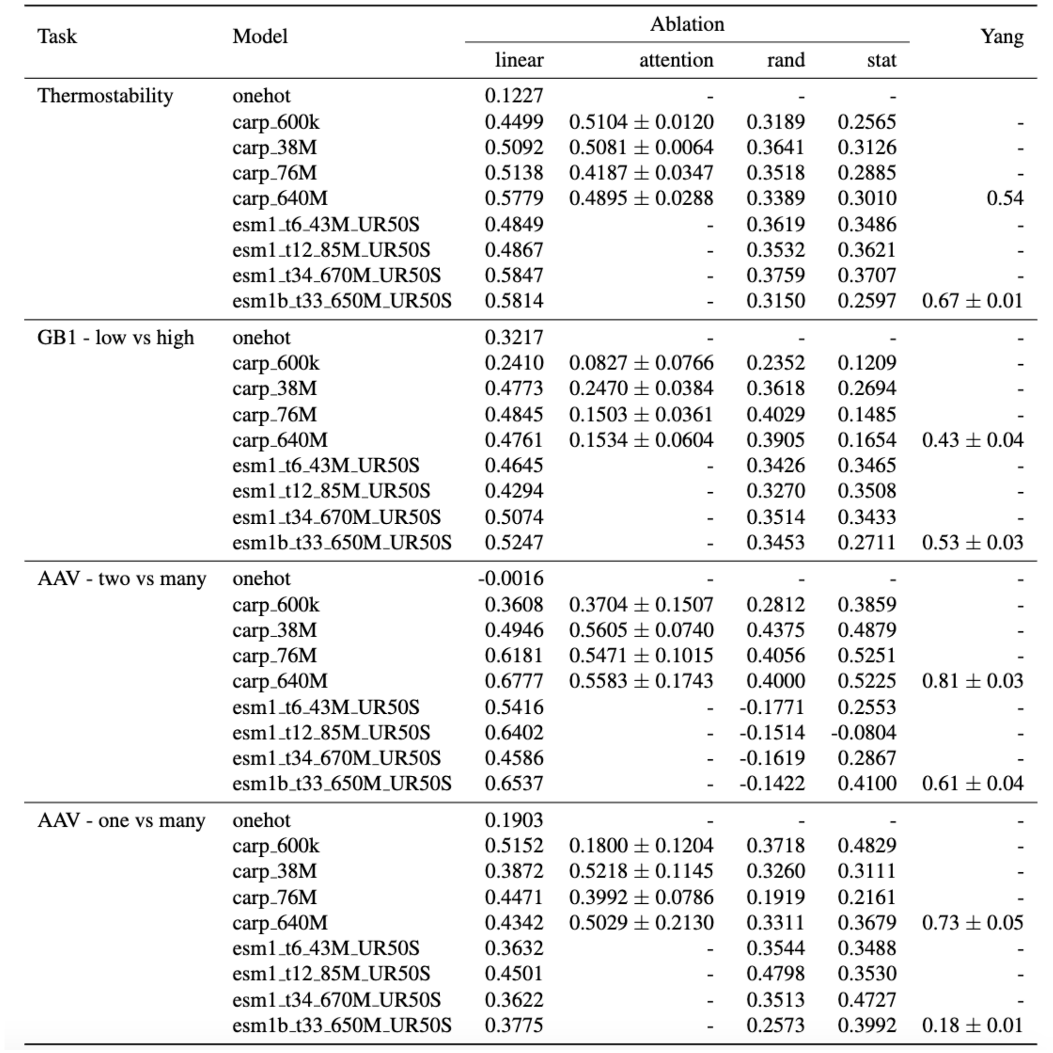
Table A.1.2. *Pretrained models.*

Name	Size	Name in code	Layers	Parameters	Embedding dimension
ESM-43M	Small	esm1_t6_43M_UR50S	6	43M	768
ESM-85M	Medium	esm1_t12_85M_UR50S	12	85M	768
ESM-670M	-	esm1_t34_670M_UR50S	34	670M	1280
ESM-650M	Large	esm1b_t33_650M_UR50S	33	650M	1280
CARP-600k	Tiny	carp_600k	16	600k	128
CARP-38M	Small	carp_38M	16	38M	1024
CARP-76M	Medium	carp_76M	32	76M	1024
CARP-640M	Large	carp_640M	56	640M	1280

Table A.1.3. *Last layer transfer learning performance for tasks that are aligned with MLM pretraining. Values are accuracy.*

Task	Model	Ablation		
		pretrain	rand	stat
SS3 - CASP12	onehot	0.481946	-	-
	carp_600k	0.660695	0.478914	0.452729
	carp_38M	0.688671	0.563534	0.497244
	carp_76M	0.699283	0.543137	0.485254
	carp_640M	0.725055	0.517503	0.487321
	esm1_t6_43M_UR50S	0.675441	0.552508	0.537486
	esm1_t12_85M_UR50S	0.681505	0.551130	0.540656
	esm1_t34_670M_UR50S	0.713892	0.545755	0.528528
	esm1b_t33_650M_UR50S	0.717337	0.498208	0.509234
SS3 - CB513	onehot	0.488168	-	-
	carp_600k	0.711180	0.495941	0.451330
	carp_38M	0.761277	0.573476	0.479191
	carp_76M	0.792297	0.549633	0.443703
	carp_640M	0.820865	0.530729	0.452383
	esm1_t6_43M_UR50S	0.742595	0.518412	0.507689
	esm1_t12_85M_UR50S	0.770913	0.518973	0.495643
	esm1_t34_670M_UR50S	0.802556	0.517013	0.504350
	esm1b_t33_650M_UR50S	0.815163	0.449030	0.476766
SS3 - TS115	onehot	0.508551	-	-
	carp_600k	0.739395	0.506632	0.466368
	carp_38M	0.779659	0.594533	0.504915
	carp_76M	0.802989	0.571236	0.452633
	carp_640M	0.824030	0.556827	0.459433
	esm1_t6_43M_UR50S	0.766227	0.568543	0.547502
	esm1_t12_85M_UR50S	0.789018	0.566826	0.527404
	esm1_t34_670M_UR50S	0.810127	0.565412	0.544607
	esm1b_t33_650M_UR50S	0.821539	0.467715	0.501885

Table A.1.4. *Last layer transfer learning performance for tasks where transfer learning improves performance but the pretrain and downstream tasks are not aligned. Values are Spearman rank correlation. We include linear and attention probes for the pretrained models. The "Yang" column indicates results for the best-performing baseline for the PLM from Yang et al. (2024).*[10]



Task	Model	Ablation				Yang
		linear	attention	rand	stat	
Thermostability	onehot	0.1227	-	-	-	
	carp_600k	0.4499	0.5104 ± 0.0120	0.3189	0.2565	-
	carp_38M	0.5092	0.5081 ± 0.0064	0.3641	0.3126	-
	carp_76M	0.5138	0.4187 ± 0.0347	0.3518	0.2885	-
	carp_640M	0.5779	0.4895 ± 0.0288	0.3389	0.3010	0.54
	esm1_t6_43M_UR50S	0.4849	-	0.3619	0.3486	-
	esm1_t12_85M_UR50S	0.4867	-	0.3532	0.3621	-
	esm1_t34_670M_UR50S	0.5847	-	0.3759	0.3707	-
	esm1b_t33_650M_UR50S	0.5814	-	0.3150	0.2597	0.67 ± 0.01
GB1 - low vs high	onehot	0.3217	-	-	-	-
	carp_600k	0.2410	0.0827 ± 0.0766	0.2352	0.1209	-
	carp_38M	0.4773	0.2470 ± 0.0384	0.3618	0.2694	-
	carp_76M	0.4845	0.1503 ± 0.0361	0.4029	0.1485	-
	carp_640M	0.4761	0.1534 ± 0.0604	0.3905	0.1654	0.43 ± 0.04
	esm1_t6_43M_UR50S	0.4645	-	0.3426	0.3465	-
	esm1_t12_85M_UR50S	0.4294	-	0.3270	0.3508	-
	esm1_t34_670M_UR50S	0.5074	-	0.3514	0.3433	-
	esm1b_t33_650M_UR50S	0.5247	-	0.3453	0.2711	0.53 ± 0.03
AAV - two vs many	onehot	-0.0016	-	-	-	-
	carp_600k	0.3608	0.3704 ± 0.1507	0.2812	0.3859	-
	carp_38M	0.4946	0.5605 ± 0.0740	0.4375	0.4879	-
	carp_76M	0.6181	0.5471 ± 0.1015	0.4056	0.5251	-
	carp_640M	0.6777	0.5583 ± 0.1743	0.4000	0.5225	0.81 ± 0.03
	esm1_t6_43M_UR50S	0.5416	-	-0.1771	0.2553	-
	esm1_t12_85M_UR50S	0.6402	-	-0.1514	-0.0804	-
	esm1_t34_670M_UR50S	0.4586	-	-0.1619	0.2867	-
	esm1b_t33_650M_UR50S	0.6537	-	-0.1422	0.4100	0.61 ± 0.04
AAV - one vs many	onehot	0.1903	-	-	-	-
	carp_600k	0.5152	0.1800 ± 0.1204	0.3718	0.4829	-
	carp_38M	0.3872	0.5218 ± 0.1145	0.3260	0.3111	-
	carp_76M	0.4471	0.3992 ± 0.0786	0.1919	0.2161	-
	carp_640M	0.4342	0.5029 ± 0.2130	0.3311	0.3679	0.73 ± 0.05
	esm1_t6_43M_UR50S	0.3632	-	0.3544	0.3488	-
	esm1_t12_85M_UR50S	0.4501	-	0.4798	0.3530	-
	esm1_t34_670M_UR50S	0.3622	-	0.3513	0.4727	-
	esm1b_t33_650M_UR50S	0.3775	-	0.2573	0.3992	0.18 ± 0.01

Table A.1.5. *Last layer transfer learning performance for tasks where transfer learning does not improve performance. Values are Spearman rank correlation for the GB1 tasks and accuracy for subcellular localization. We include linear and attention probes for the pretrained models. The "Yang" column indicates results for the best-performing baseline for the PLM from Yang et al. (2024).*[10]

Task	Model	Ablation				Yang
		linear	attention	rand	stat	
GB1 - two vs rest	onehot	0.5428	-	-	-	-
	carp_600k	0.5623	-0.1611 ± 0.1816	0.5151	0.1754	-
	carp_38M	0.5398	0.2421 ± 0.2000	0.4202	0.4140	-
	carp_76M	0.5341	0.0701 ± 0.2850	0.4444	0.2934	-
	carp_640M	0.5817	0.3311 ± 0.0667	0.5185	0.1919	0.73 ± 0.03
	esm1_t6_43M_UR50S	0.4806	-	0.6093	0.6295	-
	esm1_t12_85M_UR50S	0.3969	-	0.5852	0.5978	-
	esm1_t34_670M_UR50S	0.5065	-	0.5882	0.5664	-
	esm1b_t33_650M_UR50S	0.5437	-	0.5421	0.4112	0.67 ± 0.07
GB1 - sampled	onehot	0.7885	-	-	-	-
	carp_600k	0.7856	0.4347 ± 0.0781	0.7528	0.6554	-
	carp_38M	0.8598	0.7811 ± 0.0320	0.8259	0.7760	-
	carp_76M	0.8506	0.6899 ± 0.0491	0.8315	0.7153	-
	carp_640M	0.8679	0.7366 ± 0.0264	0.8312	0.6752	-
	esm1_t6_43M_UR50S	0.8524	-	0.7977	0.8133	-
	esm1_t12_85M_UR50S	0.8575	-	0.7930	0.8132	-
	esm1_t34_670M_UR50S	0.8660	-	0.8003	0.8164	-
	esm1b_t33_650M_UR50S	0.8846	-	0.7899	0.7939	-
Subcellular localization	onehot	0.3740	-	-	-	-
	carp_600k	0.4494	0.5210 ± 0.0095	0.4338	0.4312	-
	carp_38M	0.4883	0.5164 ± 0.0170	0.5481	0.5013	-
	carp_76M	0.5429	0.4836 ± 0.0224	0.5377	0.5506	-
	carp_640M	0.5740	0.5397 ± 0.0410	0.5584	0.5662	-
	esm1_t6_43M_UR50S	0.5584	-	0.5558	0.5766	-
	esm1_t12_85M_UR50S	0.5688	-	0.5714	0.5688	-
	esm1_t34_670M_UR50S	0.6156	-	0.6104	0.6104	-
	esm1b_t33_650M_UR50S	0.6052	-	0.6104	0.6130	-

Table A.1.6. *Spearman's rank correlation (ρ) between downstream task performance and layer depth.*

Task	CARP-640M		ESM-650M	
	ρ	p	ρ	p
SS3 - CB513	0.989	5.850×10^{-47}	0.954	2.511×10^{-18}
SS3 - TS115	0.985	6.109×10^{-44}	0.953	4.197×10^{-18}
SS3 - CASP12	0.991	2.136×10^{-49}	0.957	1.063×10^{-18}
Thermostability	-0.090	5.042×10^{-1}	-0.432	1.068×10^{-2}
GB1 - low vs high	0.289	2.922×10^{-2}	0.814	4.817×10^{-9}
AAV - two vs many	0.809	2.583×10^{-14}	0.014	9.378×10^{-1}
AAV - one vs many	0.853	3.966×10^{-17}	0.757	2.160×10^{-7}
Subcellular localization	0.694	2.085×10^{-9}	0.621	8.896×10^{-5}
GB1 - sampled	0.325	1.362×10^{-2}	0.850	1.961×10^{-10}
GB1 - two vs rest	0.436	7.023×10^{-4}	-0.267	1.266×10^{-1}

Table A.1.7. *Pretrained CARP checkpoints.*

Name	Fraction	Loss	Accuracy	Step
carp_600k	1	2.505	0.240	4.889×10^5
carp_600k	0.5	2.512	0.239	2.393×10^5
carp_600k	0.25	2.518	0.237	1.143×10^5
carp_600k	0.125	2.527	0.234	5.204×10^4
carp_38M	1	2.303	0.300	1.027×10^6
carp_38M	0.5	2.319	0.295	5.176×10^5
carp_38M	0.25	2.339	0.289	2.569×10^5
carp_38M	0.125	2.363	0.282	1.296×10^5
carp_76M	1	2.206	0.328	6.545×10^5
carp_76M	0.5	2.225	0.322	3.280×10^5
carp_76M	0.25	2.248	0.315	1.630×10^5
carp_76M	0.125	2.278	0.307	8.318×10^4
carp_640M	1	2.019	0.382	6.220×10^5
carp_640M	0.5	2.054	0.372	3.118×10^5
carp_640M	0.25	2.094	0.360	1.547×10^5
carp_640M	0.125	2.146	0.345	7.881×10^4

Table A.1.8. *Spearman's rank correlation (ρ) between downstream task performance and CARP pretrain loss.*

Task	ρ	p
SS3 - CB513	1.000	0.000
SS3 - TS115	1.000	0.000
SS3 - CASP12	0.949	2.000×10^{-6}
Thermostability	0.552	6.251×10^{-2}
GB1 - low vs high	-0.392	2.081×10^{-1}
AAV - two vs many	0.483	1.121×10^{-1}
AAV - one vs many	0.727	7.355×10^{-3}
Subcellular localization	0.832	7.980×10^{-4}
GB1 - sampled	0.441	1.517×10^{-1}
GB1 - two vs rest	-0.084	7.954×10^{-1}

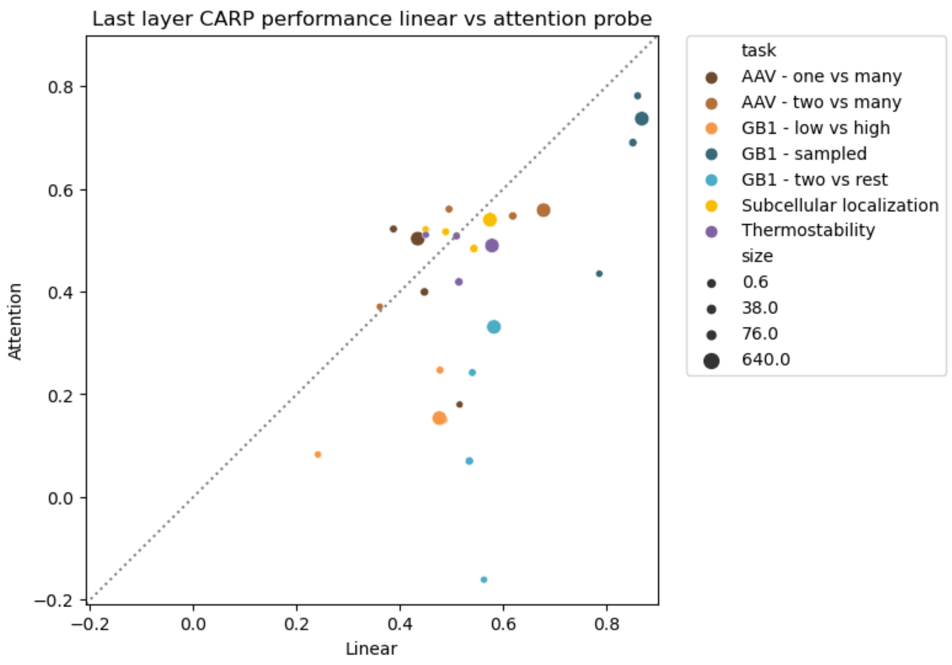


Figure A.1.1. *Compare linear and attention probes for last layer performance on downstream tasks. For the attention probe, a shallow neural net with learned aggregation is applied with 5 random seeds on 3-5 checkpoints, which is found to be inferior to the linear models across almost all downstream tasks.*

Appendix B

SUPPLEMENTARY INFORMATION FOR CHAPTER 3

B.1 Data and code availability

All data and results that support this study are deposited at https://doi.org/10.5281/zenodo.13910505. All code is available at https://github.com/fhalab/SSMuLA and https://github.com/fhalab/alde4ssmula.

B.2 Methods

B.2.1 Key resource table

REAGENT or RESOURCE	**SOURCE**	**IDENTIFIER**
Deposited data		
Bacterial toxin-antitoxin (ParD-ParE) fitness landscape	Lite, *et al.,* 2020	https://doi.org/10.7554/eLife.60924
Bacterial toxin-antitoxin (ParD-ParE) structure	Protein Data Bank	PDB: 6X0A; PDB: 5CEG
Protein G B1 domain (GB1) fitness landscape	Wu, *et al.*, 2016	https://doi.org/10.7554/eLife.16965
Protein G B1 domain (GB1) structure	Protein Data Bank	PDB: 2GI9
Dihydrofolate reductase (DHFR) fitness landscape	Papkou, *et al.,* 2023	https://doi.org/10.1126/science.adh3860
Dihydrofolate reductase (DHFR) structure	Protein Data Bank	PDB: 6XG5
T7 RNA polymerase fitness landscape	Tu, *et al.,* 2024	https://doi.org/10.1101/2022.03.09.483646
T7 RNA polymerase structure	Protein Data Bank	PDB: 1CEZ
TEV protease fitness landscape	Tu, *et al.,* 2024	https://doi.org/10.1101/2022.03.09.483646
TEV protease structure	Protein Data Bank	PDB: 1LVM
β-subunit of tryptophan synthase (TrpB) fitness landscape	Johnston, *et al.*, 2024	https://doi.org/10.1073/pnas.2400439121
β-subunit of tryptophan synthase (TrpB) structure	Protein Data Bank	PDB: 8VHH
Compiled data and results	This chapter	10.5281/zenodo.13910505
Software and algorithms		
SSMuLA code and conda environment	This chapter	https://github.com/fhalab/SSMuLA
Active Learning-Assisted Directed Evolution for SSMuLA (ALDE4SSMuLA)	This chapter	https://github.com/fhalab/alde4ssmula

EVmutation webserver	Hopf, e*t al.*, 2017	https://v2.evcouplings.org/
EVmutation GitHub	Hopf, e*t al.*, 2017	https://github.com/debbiemarkslab/EVmutation
Evolutionary Scale Modeling (ESM-2)	Rives, *et al.*, 2021; Meier, *et al.*, 2021; Lin, *et al.*, 2023	https://github.com/facebookresearch/esm
ESM inverse folding (ESM-IF)	Hsu, *et al.*, 2022	https://github.com/facebookresearch/esm
Combinatorial Variant Effects from Structure (CoVES)	Ding, *et al.*, 2024	https://github.com/ddingding/CoVES
Triad	Protabit, Pasadena, CA, USA	https://triad.protabit.com/
Machine Learning-Assisted Directed Evolution (MLDE)	Wittmann, *et al.*, 2021	https://github.com/fhalab/MLDE
Fine-tuning protein language models boosts predictions across diverse tasks	Schmirler, *et al.*, 2024	https://github.com/RSchmirler/data-repo_plm-finetune-eval

B.2.2 Landscape preparation

To avoid biases and misrepresentations, especially for deriving landscape attributes, we chose essentially complete datasets and did not perform any imputation. We assumed that missing values followed the same distribution as the existing data and therefore did not affect attribute calculations. To reduce bias from noisy data or less reliable landscape attribute calculations, which could lead to non-generalizable conclusions, we focused on landscapes with at least 1% active variants in the main text. However, to ensure that our conclusions comparing different methods remain valid across all landscapes, we provided extensive analyses of landscapes with fewer than 1% active variants in the Supplemental information, addressing differences where applicable. The 1% threshold was derived based on the expected occurrence of one active variant in a traditional DE screening of the largest landscape in this study, calculated as (1 / (4 × 20)) × 100%.

All variant fitness values were normalized within each landscape such that the variant with the maximum fitness was assigned a value of one:

$$\omega' = \frac{\omega}{\omega_{max}}$$

where ω is the original fitness value of a variant, ω_{max} is the maximum observed fitness within the landscape, and ω' is the normalized fitness value used in all analyses.

B.2.3 Landscape attributes

To provide comprehensive context for the model outcomes, we considered two groups of attributes for this analysis: 1) fitness statistics, which included percent of active variants and parameters derived from simple statistical modeling, and 2) ruggedness, which included pairwise epistasis and the number of local optima. All values are empirically derived and calculated. We did not impute missing values, assuming that they follow the same distribution as the existing data and therefore do not affect attribute calculations. All values can be found with data deposit and all implementations can be found in the SSMuLA codebase.

B.2.3.1 Definition of active variants

For landscapes containing fitness data for variants with stop codons, "active" variants were defined as those 1.96 standard deviations above the mean fitness of all sequences containing stop codons, which are expected to be inactive.[1] For GB1, T7, and TEV we followed the cutoffs set by the authors, based on the detection limit of their fitness measurement system.[2–4]

B.2.3.2 Fitness statistics

We used the "statistical functions" (`scipy.stats`) and signal (`scipy.signal`) modules from the SciPy Python package[5] to calculate kurtosis, estimate the Cauchy peak location, and determine the number of kernel density estimation (KDE) peaks. Specifically, kurtosis was calculated using the `kurtosis` function with default settings from the `stats` module. Cauchy peak location was estimated using the `fit` method from the `cauchy` distribution object in the `stats` module. The number of KDE peaks was determined by estimating the probability density function with the `gaussian_kde` function from the `stats` module and then identifying local optima using the `argrelextrema` function from the signal module.

B.2.3.3 Pairwise epistasis calculation

We classified pairwise epistasis into three categories: magnitude, sign, and reciprocal sign. For each active variant, we assigned an epistasis type for each possible double substitution at chosen sites.[1] We then calculated the fraction of epistasis type for each starting variant in the landscape. To enhance relevance to DE navigability, we incorporated additive interactions into magnitude epistasis, and merged sign and reciprocal sign epistasis into non-magnitude epistasis. Missing values were omitted.

For each unique starting variant, ab, let ω_{ab} be the fitness value of the starting variant, ω_{Ab} be fitness value of the variant with an amino acid substitution at position A only, ω_{aB} be fitness value of the variant with an amino acid substitution at position B only, and ω_{AB} be fitness value of the variant with amino acid substitutions at both positions A and B.

<u>*Magnitude epistasis*</u>

The combined effect of two amino acid substitutions is larger than or equal to their additive effects in the same direction as each individual mutation. This is navigable through single-step or recombination-based DE methods. The fraction of magnitude epistasis was calculated by simply counting the number of instances classified as magnitude epistasis divided by the total number of pairwise interactions.

<u>*Non-magnitude epistasis*</u>

Non-magnitude epistasis includes sign epistasis and reciprocal sign epistasis. For sign epistasis, the direction of the effect of one amino acid substitution in the presence of the other such that the substitution order impacts single-step DE navigability. For reciprocal sign epistasis, the combined effect changes the direction of both mutations in the presence of each other. This is not accessible with single-step DE that is inherently a greedy uphill walk. The fraction of the non-magnitude epistasis was simply calculated using one minus the fraction of magnitude epistasis.

B.2.3.4 Pairwise epistasis correlation with C-alpha distances

The pairwise C-alpha distances of mutated residues were calculated based on each of the parent structure (PDB: 6X0A, 5CEG, 2GI9, 6XG5, 1CEZ, 1LVM, and 8VHH) and then averaged for each landscape. The Spearman's correlation was then calculated between the average C-alpha distance between residues in the landscape and the fraction of pairwise non-magnitude epistasis.

For each residue pair (i, j) within the landscape, the C-alpha distance d_{ij} was computed as:

$$d_{ij} = \|\boldsymbol{r}_i^{C\alpha} - \boldsymbol{r}_j^{C\alpha}\|$$

where $\boldsymbol{r}_i^{C\alpha}$ and $\boldsymbol{r}_j^{C\alpha}$ are the Cartesian coordinates of the C-alpha (C_α) atoms for residues i and j, $\|\cdot\|$ denotes the Euclidean norm, computed simply as:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) are the 3D coordinates of the C-alpha atoms for residues i and j.

For each landscape, the averaged C-alpha distance is then computed as

$$\bar{d} = \frac{1}{N} \sum_{(i,j)\in P} d_{ij}$$

where $\bar{d}$ is the mean pairwise C-alpha distance for the landscape, P is the set of all residue pairs in the landscape, and N is the total number of residue pairs.

B.2.3.4 Local optimum calculation

A local optimum is a variant with higher fitness than all its neighboring active variants differing by one amino acid substitution.[1,6–9]

A variant with sequence x is a local optimum if:

$$\omega_x > \omega_{x'} \ \ \forall x' \in \mathcal{H}(x)$$

where ω_x is the fitness of variant with sequence x, $\mathcal{H}(x)$ is the set of all neighbors of x with a single-amino acid substitution (Hamming distance of one), and x' represents any neighboring variant in $\mathcal{H}(x)$.

B.2.4 ZS calculations

We calculated six different ZS scores for each landscape. The ranking correlation between ZS scores and the ground-truth fitness values was calculated using Spearman's correlation and the active/inactive variant classification was quantified by ROC-AUC. All values can be found with data deposit and all implementations can be found in the SSMuLA codebase.

B.2.4.1 Hamming distance

Hamming distance reflects the number of amino acid differences between a variant and the parent sequence. The negation of Hamming distance was used as the score, meaning that a less negative score corresponded to fewer number of amino acid substitutions from the parent sequence and, therefore, a more fit variant. In the main text, the parent sequence was defined by the authors of each landscape. Simulations provided in the Supplemental information explored all possible parent sequences, starting from any active variant (Figures S7–S9). The Hamming distance from each given parent sequence was used for fitness ranking and active/inactive variant classification, with the final results averaged over all possible parent sequences.

B.2.4.2 EVmutation score

The EVmutation scores were generated using EVcouplings, which employs a Potts model framework to infer conservation and coevolutionary patterns from a multiple sequence alignment (MSA).[10,11] The probability $P(x)$ of a sequence x is given by:

$$P(x) = \frac{1}{Z}\exp\{E(x)\}$$

where Z is a normalization constant, and $E(x)$ represents the total energy of the sequence x, computed as:

$$E(x) = \sum_{i} h_i(x_i) + \sum_{i<j} J_{ij}(x_i, x_j)$$

where $h_i(x_i)$ captures site-specific constraints, and $J_{ij}(x_i, x_j)$ models coevolutionary interactions between amino acids. The coupling parameters h and J were estimated using a regularized maximum pseudolikelihood.

The effect of amino acid substitutions was quantified using the EVmutation score ($S_{EVmutation}$), calculated as the energy difference between the parent sequence (x^{parent}) and the variant sequence with amino acid substitutions ($x^{variant}$):

$$S_{EVmutation}(x^{variant}) = \Delta E(x^{variant}, x^{parent}) = E(x^{variant}) - E(x^{parent})$$
$$= log\frac{P(x^{variant})}{P(x^{parent})}$$

For each fitness landscape, the parent sequence was uploaded to the EVcouplings webserver (https://v2.evcouplings.org/) using the default parameters for MSA generation (Sequence database: UniRef90; Bitscore sequence inclusion thresholds: 0.10, 0.30, 0.50, and 0.70; Search iterations: 5; Position filter: 70%; Sequence fragment filter: 50%; Removing similar sequences: 90%; Downweighting similar sequences: 80%) and for evolutionary couplings inference (Statistical inference method: pseudo-likelihood maximization). The resulting evaluation parameters were also kept as default (Contact distance cutoff: 5.0 Å; PDB structure search method: conservative; Bitscore threshold for finding known homologous 3D structures: 0.50), but they are independent of the downstream analyses.

To ensure complete coverage of substituted sites, we prioritized EVcouplings models in the following order: the recommended model, if it covered all substitution sites; the model with the highest bitscore that covered all substitution sites; and if no model covered all substitution sites, the position filter was reduced from 70% to 50% to increase sequence coverage and the process was repeated. The final "EVcouplings model parameters" were downloaded from the webserver as ".model" binary files, available in the "Download" tab under the "Evolutionary couplings" section. The models used in the study are available on Zenodo. As an alternative to using the webserver, a local version of EVcouplings can be installed from the EVcouplings GitHub repository (https://github.com/debbiemarkslab/EVcouplings), which can then be used to obtain the EVcouplings models.

The EVmutation scores were computed using the selected EVcouplings model. The EVcouplings Python library (https://github.com/debbiemarkslab/EVcouplings) was used to parse the model output (regardless of whether the model was obtained via the webserver or local command-line execution). The `CouplingsModel` module was used to load the model file, and the `delta_hamiltonian` method was applied to compute EVmutation scores.

B.2.4.3 ESM-2 score

The ESM-2 score was based on the pretrained protein language model's masked language modeling objective, resulting in likelihoods for amino acid substitutions given their surrounding context.[12] Unlike EVmutation, ESM-2 does not explicitly use MSAs but is trained on UniRef sequences and extracts logits using a mask-filling protocol for each amino acid position. For each landscape, the parent sequence was tokenized using a batch converter. Each variant position was masked and a log-softmax operation was applied to obtain the probability distribution over all possible amino acid substitutions at the masked positions. The ESM-2 score for each position was calculated using the log-odds ratio between the variant sequence with amino acid substitutions ($x^{variant}$) and the parent sequence (x^{parent}). The ESM-2 score (S_{ESM-2}) for a given variant was computed by summing the contributions from individual substitutions:[12,13]

$$S_{ESM-2}(x^{variant}) = \sum_{i \in M} \log P(x_i = x_i^{variant} | x_{\setminus M}) - \log P(x_i = x_i^{parent} | x_{\setminus M})$$

where M denotes the set of substitution sites, $x_{\setminus M}$ represents sequence x with residues in M masked. The ESM-2 score for the parent sequence was assigned a value of 0.

We found that ESM-1v (esm1v_t33_650M_UR90S_1), ESM-1b (esm1b_t33_650M_UR50S), and ESM-2 (esm2_t33_650M_UR50D) yielded comparable results, and we chose to proceed with ESM-2 for all analyses reported in this study.

B.2.4.4 ESM-IF score

The ESM-IF score was calculated as log-likelihood ratio between the variant and parent sequences using the inverse folding model ESM-IF1 (esm_if1_gvp4_t16_142M_UR50),[14] conditioned on the experimentally determined parent protein structure from PDB.[15] This model assigns likelihoods to sequences based on a given backbone structure. It incorporates a Geometric Vector Perceptron (GVP) module that ensures invariance to transformations such as rotations.[16]

For each fitness landscape, the parent PDB structure was provided as an input and its target protein chain (chain A) backbone atomic coordinates (B) were extracted. The corresponding parent sequence (x^{parent}) was extracted, and the full sequence for each variant ($x^{variant}$) was supplied as a FASTA file. The log-likelihoods for both the parent sequence and each variant were computed using `esm.inverse_folding.util.score_sequence(model, alphabet, coords, sequence)`. The ESM-IF score (S_{ESM-IF}) for a given variant was then computed as the log-likelihood ratio between the variant and parent sequences, given the fixed parent backbone structure (B):

$$S_{ESM-IF}(x^{variant}) = \log P(x^{variant} | B) - \log P(x^{parent} | B)$$

B.2.4.5 CoVES score

The CoVES score was calculated using pretrained weights (RES_1646945484.3030427_8.pt) from Ding *et al.* (2024), following the "Unsupervised protein variant scoring" methods section and the corresponding GitHub repository (https://github.com/ddingding/CoVES).[17] The scoring was applied to all the parent structures from the Protein Data Bank (PDB). Per-residue amino acid preference scores were calculated at each site throughout the substitution-containing chain (chain A). The scores were averaged across 100 replicates (`n_ave = 100`). Variant CoVES scores were than calculated by summing the log-probability-normalized classifier scores at each site, using a temperature of 0.1 (`t=0.1`).

Specifically, the normalized probability for the amino acid substitution i at a given site, given t, is:

$$P_t(s_i) = \frac{\exp\left(-\frac{|s_i|}{t} - max_j\left(-\frac{|s_j|}{t}\right)\right)}{\sum_j \exp\left(-\frac{|s_j|}{t} - max_j\left(-\frac{|s_j|}{t}\right)\right)}$$

where s_i is the preference score for amino acid substitution i at this site, s_j represents the preference scores for all 20 possible amino acid substitutions at this site, and t is the temperature parameter. $max_j\left(-\frac{|s_j|}{t}\right)$ is a log probability shift applied to prevent numerical underflow.

The variant CoVES score (S_{CoVES}) for a variant sequence with amino acid substitutions ($x^{variant}$) was then calculated by summing the log probabilities over all the sites covered in the landscape:

$$S_{CoVES}(x^{variant}) = \sum_{l=1}^{n_{site}} \log P_t(s_i)$$

where $P_t(s_i)$ is the normalized probability for an amino acid substitution i at site n given t and n_{site} is the total number of sites covered in the landscape. The CoVES scores were only compared within each landscape.

B.2.4.6 Triad score

The Triad score estimates mutant stability by calculating the change in its free energy of folding relative to the parent ($\Delta\Delta G_f$). Calculations were performed using the Rosetta energy function under a fixed-backbone assumption. All calculations were carried out on a local installation of the Triad software suite (version 2.1.3, Protabit, Pasadena, CA, USA: https://triad.protabit.com). Installation details are provided in the Triad User Manual

(https://triad.protabit.com/api/static/doc/user/userGettingStarted.html). The databases and dependencies were downloaded and built using the Makerfile, including `mpirun` (OpenMPI 1.6.5) and Python 2.7.6.

The parent protein crystal structure for each landscape was obtained from the PDB. For each fitness landscape, the structure processing and scoring were performed using the Triad suite following the method described by Wittmann *et al.* (2021)[18] and detailed below:

For each parent structure, preprocessing was performed by executing the following command inside the `${INPUT_PDB_DIR}` directory, which contained the input PDB file, `${TRIAD_DIR}/triad.sh ${TRIAD_DIR}/apps/preparation/proteinProcess.py -struct ${INPUT_PDB_DIR}/${NAME}.pdb -crosetta`, where `${TRIAD_DIR}` is the directory containing the `triad.sh` executable and `${NAME}` is the name of the input PDB file. The command generated the processed PDB file (`${INPUT_PDB_DIR}/${NAME}_prepared.pdb`) with an added constrained minimization, which was used for the downstream calculations.

A ".mut" file was generated separately to describe amino acid substitutions for all variants relative to the parent sequence in the landscape. Each row corresponded to a single variant and followed the format `Chain_SiteSubstitution` for single substitution (e.g., `A_26H`) or joined by a plus sign `+` for multiple substitutions (e.g., `A_26H+A_27I+A_28F`). The parent sequence should not be included in the ".mut" file.

The Triad scoring process was executed using the command `${TRIAD_DIR}/tools/openmpi/bin/mpirun -np ${NUM_CPUS} ${TRIAD_DIR}/triad.sh ${TRIAD_DIR}/apps/cleanSequences.py -struct ${PROC_PDB_FILE} -rosetta -inputSequenceFormat pid -inputSequences ${MUTANTS_FILE} -floatNearbyResidues -numPDBs=${numPDBs} -soft 2>&1 | tee $OUTPUT_TXT`, where `${PROC_PDB_ FILE}` is the processed parent PDB file (i.e., `${INPUT_PDB_DIR}/${NAME}_prepared.pdb`), `${MUTANTS_FILE}` is the ".mut" file containing variant substitutions, `${numPDBs}` is the total number of variants (number of lines in the ".mut" file), and `$OUTPUT_TXT` is the output summary ".txt" file containing the scoring results. The outputs consisted of the ".txt" file along with individual PDB file for each variant. These files were stored in the same directory where the command was executed.

The scoring output was located towards the end of ".txt" file under the "Solution summary" section. Each row contained the variant index, name, score, and sequence (of the concatenated substituted amino acids and "-" for the same amino acid as the parent). Variants were ranked from the most stable (most negative score) to the least stable (least negative score). The parent sequence was labeled as "WT".

Example output:

```
Solution summary:
All sequences:
Index                         Tags        Score    Seq      Muts
     0  A_26H+A_27I+A_28F,2544  -646.11885   HIF     A_26H+A_27I+A_28F
     1 A_26H+A_27V+A_28F,2744  -646.04873   HVF     A_26H+A_27V+A_28F
     2 A_26Q+A_27V+A_28F,5544  -645.85691   QVF     A_26Q+A_27V+A_28F
   …                            …            …       …       …
 1118                          WT  -639.45570   ---      WT
   …                            …            …       …       …
7999 A_26P+A_27P+A_28P,5052     -586.96179    PPP    A_26P+A_27P+A_28P
```

To ensure consistency with other ZS scores, the negation of the score was taken as the Triad score such that higher, more positive Triad scores indicate more fit variants. As an alternative to the command-line version, the web app is available at https://triad.protabit.com (version 3.0.2). The standardize structure and score variants apps are equivalent to the command-line `proteinProcess` and `cleanSequences`, and accept the same PDB and ".mut" files. We also noted that tools such as FoldX could be used as an alternative.[13]

B.2.5 ZS ensembles

For each landscape, the six different ZS scores were computed independently. Naive ensemble scores were then derived using one of two methods: Hamming distance-based down-selection or the sum of the individual ZS ranks.

Hamming distance-based ensemble:

A Hamming distance cutoff of two was first applied to preselect variants, meaning only variants differing by two or fewer amino acid substitutions from the reference parent sequence were considered (two-site libraries). Within this subset, the remaining five ZS scores were then used to rank the variants, determining the final ensemble score.

Other naive ensemble:

For all other ensembles, each ZS score was assigned equal weight. The ensemble score for a variant was computed as the sum of its rankings across the selected ZS predictors. Variants were then ranked based on this summed score, and the resulting ranking was used for ZS-focused library construction.

B.2.6 ZS analysis

B.2.6.1 ZS MSA depth correlation

The MSA depth referred to the number of sequences resulting from EVcouplings search, where all mutation sites were covered.

B.2.6.2 ZS pairwise correlation

The pairwise correlation was performed for each landscape and then averaged across the 12 landscapes with at least 1% active variants.

B.2.7 DE simulations

All DE simulations were performed for each N-site library of a given landscape, where N was the number of targeted sites in the landscape (n_{site}). Each DE simulation started from an active variant, which served as the "parent" for the simulated DE, regardless of its original parent. The maximum fitness achieved by each starting variant was recorded. All DE simulations were repeated for all active variants within a given landscape. Two evaluation metrics were then calculated across all active variants within the landscape: (1) average maximum fitness achieved and (2) fraction reaching the global optimum. The total number of unique variants was given by $n_{total} = n_{sample} + n_{test}$. Both three-site and four-site recomb DE share the same number of unique variants sampled (n_{sample}) as their corresponding top96 recomb DE. In single-step DE, the same n_{sample} is divided across $n_{round} = n_{site}$ rounds, where $n_{sample} = n_{total}$ since $n_{test} = 0$. To ensure adequate coverage of the variant space (e.g., 95%), the total number of screened variants, denoted $n_{total}{}^{\dagger}$, is given by $n_{total}{}^{\dagger} = n_{screen} + n_{test} = 66 \times\ n_{site} +\ n_{test}$ for each DE strategy. The number of variants screened per site to cover all 20 amino acids can be approximated by $n_{screen} = -n_{codon} \times \ln(1 - p)$, where n_{codon} represents the number of codons used to generate the SSM library and p is the desired library coverage.[19,20] For 95% coverage of a single-site SSM library using the 22-codon trick (n_{codon}= 22), n_{screen} is approximated as $66 \times\ n_{site}$. In practice, n_{codon} varies depending on the SSM library generation methods and a three-fold oversampling is often used as a rule of thumb.

B.2.7.1 Single-step DE

This is a greedy walk algorithm, where SSM was performed at each unique site sequentially. The process begins with selecting one of the possible substitution sites, evaluating the fitness impact of all possible amino acid substitutions at this position. The substitution yielding the highest fitness is fixed, and the position is restricted from further exploration. In the next round, one of the remaining positions is selected, with all mutants evaluated, and the best substitution is fixed again. This process repeats iteratively until all positions have been evaluated yielding the fitness of the best variant identified in the last ($n_{round} = n_{site}$) round. Consequently, each site is optimized once per simulation. Over $n_{round} = n_{site}$ rounds, a total of $n_{total} = (19\ \times\ n_{site} + 1) + 0$ unique variants were sampled and a total of $n_{total}{}^{\dagger} = (66\ \times\ n_{site} + 1) + 0$ variants were considered to achieve 95% variant space coverage.

Each single-step DE simulation for a given active variant was repeated $n_{site}!$ (factorial) times to sample all possible orders of the n_{site} sites. For example, a four-site library requires four rounds of single-step DE to reach the optimal variant and there is a total of 24 (4!) possible orders of sampling. This is a deterministic approach to navigate the fitness landscape as the best variant is always selected.[1,18,21]

B.2.7.2 Recombination strategies

Two recombination strategies (recomb DE and top96 recomb DE) performed simultaneous site-saturation mutagenesis (SSM) at each site in the initial round. The best variant at each site was additively recombined. Either the top one variant or the top 96 variants (matching the number of wells in plates commonly used for screening) were used in the evaluation round ($n_{round} = 2$).

Recomb DE

This is a naive recombination. This approach starts from an active variant in the combinatorial space, independently optimizing each site within the context of the initial sequence and then combining the best substitutions from each site into a new variant.[1,21] Over $n_{round} = 2$ rounds, a total of $n_{total} = (19 \times n_{site} + 1) + 1$ unique variants were sampled and a total of $n_{total}{}^{\dagger} = (66 \times n_{site} + 1) + 1$ variants were considered to achieve 95% variant space coverage.

Top96 recomb

This is an alternative recombination approach. All substitutions are made at each of the sites independently in the background of the initial sequence, calculating fitness for all combinations from single substitution over the initial sequence. The sequences are then ranked based on their fitness, and the top 96 variants are calculated *in silico* assuming perfect additivity. The reported maximum fitness reflects the highest observed among the initial sequence, any single substitutions, and the best of the top 96.[1] Over $n_{round} = 2$ rounds, a total of $n_{total} = (19 \times n_{site} + 1) + 96$ unique variants were sampled and a total of $n_{total}{}^{\dagger} = (66 \times n_{site} + 1) + 96$ variants were considered to achieve 95% variant space coverage.

B.2.8 MLDE, ALDE, fine-tuning, and focus-training simulations

Each simulation was performed on a given landscape. For all ML simulations, a range of total unique variants screened (n_{total} = 120, 144, 192, 288, 384, 480, 576, 672, 1056, and 2016) was considered. n_{total} variants were split across training-validation and testing for MLDE, ftMLDE, fine-tuning, and ZS-guided fine-tuning or multiple rounds of sampling for ALDE and ftALDE. All strategies were evaluated using two metrics: (1) average maximum fitness achieved and (2) fraction reaching the global optimum. All results were averaged across 50 seeded replicates for MLDE, ALDE, and focused training, and five seeded replicates for fine-tuning (constraint by computational resources).

B.2.8.1 Encoding strategies

One-hot and learned representations from ESM-2 (esm2_t33_650M_UR50D) were tested. One-hot sequence encodings were flattened over the targeted sites. Learned representations from ESM-2 (esm2_t33_650M_UR50D) were implemented in three ways, (1) flattened over the targeted sites, (2) mean pooled over the targeted sites, and (3) mean pooled over the full sequence.

B.2.8.2 MLDE simulations

For each MLDE simulation on a given landscape, we evaluated a range of total unique variants screened (n_{total} = 120, 144, 192, 288, 384, 480, 576, 672, 1056, and 2016). Each n_{total} was divided into a training-validation round (n_{train} = 24, 48, 96, 192, 288, 384, 480, 576, 960, and 1920) and an evaluation round (n_{test} = 96, n_{round} = 2). . The training-validation set was either randomly sampled from the full N-site library or from a focused library containing variants with ZS scores in the top 12.5%, 25%, or 50% of the full N-site library (12.5% for main results, while alternative cutoffs were discussed in the Discussion and Supplemental information; see also the focused-training simulations in Methods). Training was performed using five-fold cross-validation. Models were trained using XGBoost[22] (boosting ensembles) and the Scikit-learn ridge regression[23] (ridge regression).

The trained models were used to predict variant fitness across the entire landscape, and the top 96 predicted variant fitness values ($n_{test} = 96$) were used for the evaluation metric calculations. Boosting models were trained with the `reg:tweedie` objective and an `early_stopping_rounds` set to 10. Ridge regression used an alpha value of 1. Hyperparameters were not tuned. Models trained with boosting ensembles using one-hot encoded sequences were reported in the main text. Alternative models (ridge regression), sequence representations (ESM-2 with three pooling options), and strategies (ESM-2 fine-tuning) were detailed in the Discussion and Supplemental information. All sampling and model splits were performed with fixed seeds for reproducibility.

B.2.8.3 ALDE simulations

We tested two surrogate models (boosting ensembles and deep neural network ensembles (DNN)) and three acquisition functions (greedy, upper confidence bound (UCB), and Thompson sampling (TS)) following the methods described by Yang *et al.*[24] We presented ALDE boosting ensembles with greedy acquisition function in the main results and supplemented the alternatives in the Discussion section.

An ensemble of models (predicting fitness values from sequences) was used to estimate the posterior distribution of the objective function (denoted as f). The acquisition function quantified the potential benefit of evaluating any given batch of inputs based on these predictions. In each iteration of the optimization loop (termed "rounds" and denoted n_{round} in the main text), a new batch of inputs was selected by maximizing the acquisition function. After evaluating the objective function at these new inputs, the surrogate model was updated, and the process repeats. BoTorch[25] and GPyTorch[26] were used. Hyperparameters were based on those reported by Yang *et al.*

Surrogate models

Let $\boldsymbol{X}$ denote all feasible protein sequences over an N-site (n_{site}) landscape and f denote the objective function (fitness) to be optimized. To approximate f, we trained two types of surrogate model ensembles:

Boosting ensembles

Each boosting ensemble was trained using bootstrapping, where five independent boosting models were trained on 90% randomly sampled subsets of the total training data and implemented the `reg:tweedie` objective with `early_stopping_rounds = 10`.

Deep neural network ensembles (DNN)

Each DNN ensemble was trained using bootstrapping, where five independently initialized deep neural networks were trained on 90% randomly sampled subsets of the total training data. Models were optimized using `torch.optim.Adam` optimizer with the `torch.nn.MSELoss` loss from PyTorch.[27]

Acquisition functions

The acquisition function quantified the potential benefit of evaluating any given batch of inputs based on the predictions from the surrogate models.

Upper confidence bound (UCB)

The UCB acquisition function selects sequences by balancing exploitation and exploration:

$$\alpha_n(x) = \mu_n(x) + \beta_n^{1/2}\sigma_n(x)$$

where $\mu_n(x)$ is the predicted (posterior) mean, $\sigma_n(x)$ is the standard deviation, and $\beta_n = 4$ controls the exploration-exploitation trade-off. To form a batch of sequences, we selected the top q sequences that yield the highest values of $\alpha_n(x)$, evaluated across all discrete sequence x in the design space.

Greedy acquisition

The greedy acquisition function selects sequences solely based on the predicted mean $\mu_n(x)$. This makes it a special case of UCB with $\beta_n = 0$, focusing purely on exploitation.

Thompson sampling (TS)

TS selects sequences by drawing random samples from the posterior distribution of f, choosing:

$$x_{n+1} = \arg\max_{x \in X} f_{sampled}(x)$$

where $f_{sampled}(x)$ is randomly selected from one of the models in the ensemble. To from a batch of sequences, each sequence was sampled independently.

ALDE simulation details

For each fitness landscape, models were trained and evaluated on a total of 120, 144, 192, 288, 384, 480, 576, 672, 1056, and 2016 samples (n_{total}). These samples were distributed across two, three, or four acquisition rounds (n_{round}), with each round sampling n_{batch} = n_{total} / n_{round} variants. Samples for the initial round were drawn either randomly from the full N-site library or randomly from a focused library containing variants with ZS scores in the top 12.5%, 25%, or 50% of the full N-site library (12.5% was used for the main results, with alternative cutoffs discussed in the Discussion and Supplemental information). Samples for the subsequential rounds were selected based on the different acquisition functions.

For example, given n_{total} = 120, n_{round} = 2 (ALDE x 2), the initial round samples n_{batch} = 60 variants randomly from the full N-site library or from a focused library, followed by a second round of n_{batch} = 60 variants sampled based on the initial round; For n_{round} = 3

(ALDE x 3), the initial round samples n_{batch} = 40 variants, either randomly from the full N-site library or from a focused library, followed by two subsequent rounds, each sampling n_{batch} = 40 variants based on its previous round. The evaluation metrics were calculated based on the fitness of the variants sampled in the final batch.

Results using boosting ensembles with greedy acquisition and one-hot sequence encoding were reported in the main text. Alternative models (deep neural network ensembles (DNN)) and acquisition functions (upper confidence bound (UCB), Thompson sampling (TS)) were detailed in the Discussion and Supplemental information.

B.2.8.4 Fine-tuning simulations

LoRA fine-tuning was performed based on the study by Schmirler *et al.*[28] The script in SSMuLA codebase was adapted from the original Jupyter notebook (https://github.com/RSchmirler/data-repo_plm-finetune-eval/blob/main/notebooks/finetune/Finetuning_per_protein.ipynb).

Specifically, ESM-2 (esm2_t33_650M_UR50D) was fine-tuned using the default `LoraConfig` parameters, with a low-rank adaptation factor of `r=4`, `lora_alpha=1`, and `bias="all"`, applied to the query, key, value, and dense transformer modules. The dataset consisted of random or focused training samples of varying sizes (n_{train} = 24, 48, 96, 192, 288, 384, 480, 576, 960, and 1920), which were split into 90% training and 10% validation sets. Spearman's rank correlation was used as the validation metric to assess model performance. The training setup followed the recommended configuration, with an effective batch size of eight (`batch=4` for training batch size and `accum=2` for gradient accumulation), a validation batch size of `val_batch=16`, and a total of 10 training epochs. The learning rate was set to `lr=3e-4`, and mixed precision training was enabled (`mixed=True`). DeepSpeed acceleration was disabled (`deepspeed=False`), and only LoRA parameters were trained (`full=False`). After training, the fine-tuned model was used to predict variant fitness across the entire landscape, and the top 96 predicted variants were analyzed.

B.2.8.5 Focused training simulations

Three focused training set cutoffs were tested, corresponding to 50%, 25%, and 12.5% of the total mutant library. A 12.5% cutoff was used for all simulations except those investigating the optimal focused training library size (Figure S41). To construct the focused training sets for each fitness landscape, all ZS scores were first computed for all variants, and the variants were then ranked accordingly. The top 1,000 variants in three-site landscapes and the top 20,000 variants in four-site landscapes (12.5% of the total mutants) were selected as the focused training library. These focused training sets were subsequently used to randomly sample training data for MLDE, the initial round of ALDE, or fine-tuning. Due to computational resource constraints, only EVmutation-guided focused training was tested in the fine-tuning simulations.

B.2.9 Decision tree simulations

Following the decision tree (Figure 5), we simulated a prospective decision-making process to choose a ML-strategy and associated ZS predictor. We defined a landscape as "hard-to-navigate" if the average pairwise C-alpha distance of substituted residues was ≤10 Å[29] or if the sites were located in an enzyme active site. A "good ZS prior" was defined as using EVmutation if the MSA contained more than 1,000 sequences, using ESM-IF for binding interactions, and having no good ZS prior otherwise. We defined "low n_{total}" as $n_{total} \leq 480$ and "large search space" as a landscape targeting four or more sites. The ML strategies were the same as those in the main text. The MLDE strategies used boosting ensemble with one-hot encoding. 384 variants for training-validation round with 96 for evaluation round were tested in the low n_{total} setting and 1920 with 96 variants otherwise. The ALDE strategies used boosting ensembles with greed acquisition using one-hot encoding. 480 variants were equally split over four rounds in the low n_{total} setting and 2016 variants over four rounds otherwise. The evaluation metrics were calculated based on 50 replicates.

B.2.10 Elo rating

We performed Elo rating calculations to compare (1) different DE, MLDE, ALDE, and focused training strategies and (2) various ZS predictors applied to different ftMLDE and ftALDE methods. The calculations were performed across a range of total unique variants screened (n_{total} = 120, 144, 192, 288, 384, 480, 576, 672, 1056, and 2016) for each ML-based strategy. We modified the code from the Colab notebook by Large Model Systems (LMSYS Corp; https://colab.research.google.com/drive/1RAWb22-PFNI-X1gPVzc927SGUdfr6nsR?usp=sharing).[30] "Methods" refer to either different DE or ML-assisted strategies in (1), where the average of the six ZS predictors was taken for each of the focused training strategy, or the different ZS predictors and their ensembles in (2). Elo ratings were computed for both evaluation metrics: average max fitness achieved and the fraction reaching the global optimum. For each analysis, bootstrap Elo ratings were computed separately for different evaluation metrics. The results of the Elo rankings were used to assess the relative effectiveness of different strategies and predictors in optimizing variant screening outcomes.

B.2.10.1 Elo rating computation

Elo ratings were computed based on pairwise comparisons between methods within each landscape. Pairwise comparisons were generated by grouping data by the landscape, and then computing the mean performance of each method within the library. All possible pairwise comparisons between methods within a given library were then performed. If one method had a higher mean score than another, it was assigned as the winner; otherwise, the result was recorded as a tie.

The Elo rating system was used to quantify the relative ranking of methods based on the pairwise comparison results.[31] All Elo parameters were set according to standard values to ensure a balance between stability and responsiveness in the Elo rating system. Each method was initially assigned a default Elo rating of 1000 to provide a fair starting point for all methods, preventing negative values and ensuring comparability across different strategies.

The ratings were then updated iteratively based on match outcomes. The rating updates followed the standard Elo update formula:

$$E_A = \frac{1}{1 + base^{(R_B - R_A)/scale}}$$

$$E_B = \frac{1}{1 + base^{(R_A - R_B)/scale}}$$

$$R'_A = R_A + K(S_A - E_A)$$

$$R'_B = R_B + K(S_B - E_B)$$

where E_A and E_B are the expected probabilities of winning for methods A and B, R_A and R_B are the current Elo ratings, and S_A and S_B are the actual score of the match (1 for a win, 0 for a loss, and 0.5 for a tie). The $base$ for probability computation was set to 10, following standard Elo implementations in ranking systems for interpretability and consistency. The K-factor (scaling factor that controls the magnitude of rating updates) was set to 4 ($K = 4$) to ensure that adjustments occurred gradually over multiple comparisons and reducing the influence of individual matchups. The scale factor ($scale$) for score differences was set to 400 (the "algorithm of 400") to allow meaningful differentiation between methods.

B.2.10.2 Bootstrapped Elo computation

To account for variability in the comparisons and ensure robust ranking estimates, we applied bootstrapping with 1000 resampling iterations. Each iteration involved: randomly sampling the dataset of pairwise comparisons with replacement; recomputing Elo ratings using the resampled data; and storing the results for statistical aggregation. Deterministic random seeds were used to ensure reproducibility in bootstrapping. The median Elo score across bootstrap iterations was used as the final ranking score for each method.

Additionally, stratified bootstrapping was applied to balance the number of comparisons between methods with different frequencies of occurrence. This was applied to ZS predictor comparisons, as non-Hamming distance ensembles were not tested for all ftALDE variations. In this case, each method's comparisons were resampled independently to ensure fair representation across bootstrap iterations.

B.2.11 Feature correlation and importance analysis

To analyze how each landscape attribute correlated with the simulation targets, a Spearman's correlation was calculated between the values of the attribute and the performance values of the model. Both the Spearman's ρ and p-value were reported. To test the differences between binding and enzyme activities, t-tests were performed, where the t-statistic and p-values were reported. A p-value less than 0.05 was considered statistically significant.

B.2.11 Computational information

B.2.11.1 Hardware

The majority of computational analyses were conducted on a computing server running Ubuntu 22.04.4 LTS, equipped with two AMD EPYC 9654 96-core processors (384 logical CPUs) and two NVIDIA H100 PCIe GPUs (80GB each), running CUDA 12.4. Triad calculations were performed on a server running Ubuntu 20.04.6 LTS, equipped with two Intel Xeon Gold 6248R processors (96 logical CPUs). Fine-tuning simulations were conducted on a local workstation equipped with a 12th Gen Intel Core i9-12900KS processor (24 logical CPUs) and two NVIDIA RTX A6000 GPUs (48GB each), running CUDA 12.7.

B.2.11.2 ZS calculations

Computing Hamming distance for a full landscape is the least costly computationally. Obtaining each EVcouplings model from the online server took a few hours to a day, though this process could be expedited locally with parallelization and optimization. Computing EVmutation scores from the model took only seconds per landscape. Computing ESM-2, ESM-IF, and CoVES scores took a few minutes to under an hour per landscape, accelerated by an H100 GPU. Triad calculations took a few seconds per variant with 48 CPU cores, requiring to several hours or up to a day for an entire landscape. The computational cost for ensembling precomputed ZS scores was minimal.

B.2.11.3 ML strategies

Each MLDE and ALDE simulation was conducted with 50 replicates, running for several minutes to under an hour with GPU acceleration, though execution on CPUs was also feasible. Fine-tuning simulations required approximately 30GB of GPU memory and took several minutes to two hours per replicate on a single A6000 GPU.

B.2.11.4 Scaling

The computational cost scales linearly with number of rounds and number of samples for training and exponentially with number of sites for evaluation.

B.3 Supplemental information

Table B.3.1. *Combinatorial landscapes with additional details including landscapes with fewer than 1% active variants (italic rows), related to* ***Table 3.1****.*[1–4,6,32]

Landscape	PDB ID	Sites	Percent complete	Percent active	Fraction of local optima	Fraction of non-magnitude epistasis	Cauchy peak location	Kurtosis	Number of KDE peaks
ParD2	6X0A	I61, L64, K80	98.52	82.89	0.001	0.34	0.0807	0.07	3
ParD3	5CEG	D61, K64, E80	98.52	91.96	0.001	0.31	0.2521	-0.29	3
GB1	2GI9	V39, D40, G41, V54	93.35	23.13	0.005	0.40	0.0003	76.92	33
DHFR	6XG5	A26, D27, L28	100.00	10.68	0.004	0.42	0.1271	19.21	7
T7	1CEZ	N748, R756, Q758	84.06	3.48	0.368	0.52	0.0000	46.51	11
TEV	1LVM	T146, D148, H167, S170	99.46	11.5	0.060	0.56	-0.0114	37.74	27
TrpB3A	8VHH	*A104, E105, T106*	*99.64*	*0.74*	*0.390*	*0.60*	*-0.0399*	*53.44*	*9*
TrpB3B		*E105, T106, G107*	*99.95*	*0.23*	*0.667*	*0.54*	*-0.0554*	*84.31*	*8*
TrpB3C		*T106, G107, A108*	*99.93*	*0.44*	*0.514*	*0.59*	*-0.0736*	*7.15*	*8*
TrpB3D		T117, A118, A119	97.04	9.26	0.043	0.50	0.0036	32.52	13
TrpB3E		F184, G185, S186	99.55	2.02	0.348	0.63	0.0008	355.09	15
TrpB3F		L162, I166, Y301	96.71	1.06	0.232	0.54	-0.0230	47.49	15
TrpB3G		V227, S228, Y301	98.64	1.37	0.213	0.52	-0.0037	131.81	23
TrpB3H		*S228, G230, S231*	*96.45*	*0.69*	*0.547*	*0.62*	*-0.0152*	*464.45*	*13*
TrpB3I		Y182, V183, F184	97.30	32.04	0.006	0.43	0.0228	9.38	6
TrpB4		V183, F184, V227, S228	99.46	6.15	0.057	0.46	0.0118	48.56	27

Table B.3.2. *MLDE percent improvement from three types of DE, related to* ***Figure 3.2a****. Calculations were based on landscapes with at least 1% active variants.*

	Recomb		Single-step		Top96 recomb	
Number of MLDE training samples	Average max fitness achieved	Fraction reaching the global optimum	Average max fitness achieved	Fraction reaching the global optimum	Average max fitness achieved	Fraction reaching the global optimum
24	-7.35	-31.60	-15.25	-67.40	-23.89	-79.13
48	3.13	30.59	-5.66	-37.76	-15.28	-60.16
96	18.29	108.32	8.20	-0.71	-2.84	-36.44
192	28.66	162.73	17.69	25.22	5.68	-19.84
288	33.27	201.60	21.91	43.75	9.47	-7.98
384	34.95	238.91	23.44	61.53	10.85	3.40
480	37.94	282.44	26.17	82.28	13.30	16.69
576	40.24	313.53	28.28	97.09	15.20	26.17
960	47.09	427.02	34.54	151.18	20.82	60.80
1920	50.85	487.65	37.99	180.08	23.92	79.30

Table B.3.3. *ftMLDE and ftALDE percent improvement from MLDE and ALDE, related to* ***Figure 3.2a****. Calculations were based on landscapes with at least 1% active variants.*

	ftMLDE from MLDE		ftALDE from ALDE		ftALDE x 3 from ALDE x 3		ftALDE x 4 from ALDE x 4	
Number of training samples	Average max fitness achieved	Fraction reaching the global optimum	Average max fitness achieved	Fraction reaching the global optimum	Average max fitness achieved	Fraction reaching the global optimum	Average max fitness achieved	Fraction reaching the global optimum
24	11.17	76.89	19.32	154.76	14.34	72.90	17.11	61.31
48	11.86	53.57	19.50	151.47	15.61	62.93	13.24	48.66
96	9.57	60.32	15.22	88.94	12.99	67.55	12.09	48.48
192	8.64	76.73	10.50	58.57	8.39	40.86	6.78	33.39
288	7.08	70.02	8.95	49.07	5.41	20.24	5.22	20.86
384	6.94	58.94	6.42	39.11	4.04	15.78	4.05	17.71
480	5.18	44.44	4.97	30.14	3.35	14.07	1.70	7.32
576	4.04	38.85	3.32	27.76	2.72	9.10	1.77	3.93
960	0.13	17.60	1.25	12.57	0.79	5.42	0.74	3.81
1920	-1.60	9.48	0.07	7.90	-0.14	6.10	-0.05	5.67

Table B.3.4. *Spearman's rank correlation between landscape attributes and single-step DE average maximum fitness achieved, as well as the average maximum fitness improvement from DE using MLDE, ALDE and focused training with total sample size of 480, related to* ***Figures 3.2c*** *and* ***B.3.6****. Calculations were based on landscapes with at least 1% active variants.*

Attribute	Single-step DE	MLDE over DE	ftMLDE over DE	ALDE over DE	ftALDE over DE	ALDE x 3 over DE	ftALDE x 3 over DE	ALDE x 4 over DE	ftALDE x 4 over DE
Percent active	0.50	-0.85	-0.80	-0.59	-0.80	-0.88	-0.80	-0.81	-0.81
Fraction of local optima	-0.76	0.76	0.54	0.48	0.60	0.63	0.53	0.50	0.52
Fraction of non-magnitude epistasis	-0.73	0.81	0.66	0.56	0.72	0.76	0.65	0.54	0.64
Cauchy peak location	0.70	-0.64	-0.55	-0.64	-0.58	-0.69	-0.58	-0.62	-0.57
Kurtosis (tailedness)	-0.82	0.71	0.78	0.38	0.71	0.70	0.83	0.62	0.86
Number of KDE peaks	-0.80	0.34	0.50	0.23	0.36	0.36	0.57	0.47	0.60

Table B.3.5. *MLDE, ALDE and focused training with 480 total sample size fold improvement from single-step DE, related to* ***Figures 3.2c*** *and* ***B.3.6****. Bold row indicates the landscape with the max improvement and italic rows indicate landscapes with fewer than 1% active variants.*

Landscape	MLDE	ftMLDE	ALDE	ftALDE	ALDE x 3	ftALDE x 3	ALDE x 4	ftALDE x 4
DHFR	1.08	1.07	1.04	1.07	1.06	1.08	1.05	1.07
GB1	1.17	1.34	1.16	1.27	1.25	1.37	1.34	1.38
ParD2	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04
ParD3	1.01	1.01	1.02	1.02	1.02	1.02	1.02	1.02
T7	1.40	1.32	1.31	1.31	1.38	1.34	1.38	1.33
TEV	1.42	1.28	1.28	1.26	1.38	1.32	1.42	1.31
TrpB3A	*1.52*	*2.20*	*1.31*	*1.84*	*1.60*	*2.02*	*1.61*	*2.15*
TrpB3B	*1.25*	*2.05*	*0.85*	*1.47*	*0.96*	*1.44*	*0.87*	*1.48*
TrpB3C	*1.09*	*1.71*	*0.89*	*1.43*	*0.88*	*1.35*	*0.98*	*1.37*
TrpB3D	1.24	1.32	1.26	1.28	1.30	1.30	1.30	1.32
TrpB3E	**2.30**	**3.48**	**1.43**	**2.84**	**1.97**	**2.96**	**1.71**	**2.96**
TrpB3F	1.30	1.43	1.38	1.39	1.40	1.43	1.37	1.43
TrpB3G	1.38	1.55	1.30	1.49	1.37	1.52	1.49	1.55
TrpB3H	*1.30*	*2.46*	*0.89*	*2.10*	*1.07*	*2.13*	*1.52*	*2.28*
TrpB3I	1.21	1.26	1.24	1.25	1.25	1.26	1.25	1.26
TrpB4	1.24	1.34	1.10	1.25	1.31	1.35	1.36	1.41

Table B.3.6. *Protein function and MSA impact ZS predictor performances test significance, related to* ***Figure 3.3c****. Correlation between Spearman's correlation of ZS predictor fitness ranking prediction with MSA depth, where the depth for the EVmutation calculation covering the full sequence is used. Bold font indicates statistically significant (p-value < 0.05).*

Metric	MSA depth (Spearman's correlation)	
ZS predictor	Spearman ρ	p-value
Hamming distance	0.54	0.07
EVmutation	0.49	0.11
ESM-2	0.71	**0.01**
ESM-IF	0.55	0.07
CoVES	0.03	0.93
Triad	-0.05	0.87

Table B.3.7. *T-test for ZS predictor between binding and enzyme activities for landscapes with at least 1% active variants, related to* ***Figure 3.4a****. Bold font indicates statistically significant (p-value < 0.05).*

Metric	Binding vs. Enzyme activities (Spearman's correlation)		Binding vs. Enzyme activities (ROC-AUC)	
ZS predictor	t-statistics	p-value	t-statistics	p-value
Hamming distance	1.740	0.210	-2.379	**0.042**
EVmutation	1.738	0.167	-1.669	0.126
ESM-2	1.297	0.308	-0.747	0.493
ESM-IF	3.316	0.052	0.749	0.494
CoVES	3.641	0.057	1.289	0.279
Triad	4.332	**0.001**	1.101	0.334

Table B.3.8. *T-test for focused training MLDE (480 total sample size) between binding and enzyme activities for landscapes with at least 1% active variants, related to* ***Figure 3.4b****.*

Metric	Binding vs. Enzyme activities (Spearman's correlation)		Binding vs. Enzyme activities (ROC-AUC)	
ZS predictor	t-statistics	p-value	t-statistics	p-value
Hamming distance	0.041	0.969	-0.720	0.525
EVmutation	0.351	0.738	-0.111	0.918
ESM-2	0.341	0.784	-0.338	0.758
ESM-IF	0.582	0.577	-0.267	0.802
CoVES	0.342	0.745	0.217	0.843
Triad	0.904	0.397	-0.210	0.845
Hamming distance + EVmutation	1.445	0.184	1.313	0.219

Table B.3.9. *T-test for focused training ALDE (480 total sample size split into four rounds) between binding and enzyme activities for landscapes with at least 1% active variants, related to* ***Figure 3.4b****.*

Metric	Binding vs. Enzyme activities (Spearman's correlation)		Binding vs. Enzyme activities (ROC-AUC)	
ZS predictor	t-statistics	p-value	t-statistics	p-value
Hamming distance	0.070	0.948	-0.441	0.688
EVmutation	0.787	0.454	0.414	0.704
ESM-2	0.267	0.801	-0.235	0.829
ESM-IF	0.768	0.465	0.279	0.795
CoVES	0.369	0.726	0.335	0.761
Triad	1.612	0.139	0.128	0.905
Hamming distance + ESM-IF	0.582	0.557	-0.024	0.982
Hamming distance + EVmutation	0.323	0.759	0.157	0.884
Hamming distance + CoVES	0.128	0.905	0.289	0.790

Table B.3.10. *Relationship between the percentage of pairwise non-magnitude epistasis (where higher values indicate harder-to-navigate landscapes) and the average pairwise C-alpha distance of substituted residues (the smaller the distance, the closer the central carbon atoms of the two amino acids at the targeted sites,* ***Appendix B.2 Methods****).*

Landscape	Average pairwise C-alpha distance	Fraction of non-magnitude epistasis
DHFR	4.43 ± 0.99	0.42
GB1	5.94 ± 1.90	0.40
ParD2	17.18 ± 10.53	0.34
ParD3	17.57 ± 10.90	0.31
T7	5.55 ± 0.90	0.52
TEV	9.04 ± 1.35	0.56
TrpB3A	4.77 ± 1.71	0.60
TrpB3B	4.88 ± 1.90	0.54
TrpB3C	4.61 ± 1.41	0.59
TrpB3D	4.33 ± 0.91	0.50
TrpB3E	4.46 ± 1.14	0.63
TrpB3F	7.58 ± 2.20	0.54
TrpB3G	6.84 ± 3.24	0.52
TrpB3H	5.74 ± 2.30	0.62
TrpB3I	4.57 ± 1.33	0.43
TrpB4	12.36 ± 6.87	0.46

Table B.3.11. *T-test for ZS predictor between binding and enzyme activities for landscapes with at least 1% active variants but with single substitution only, related to the* ***Chapter 3 Discussion section*** *and* ***Figure B.3.29****. Bold font indicates statistically significant (p-value < 0.05).*

Metric	Binding vs. Enzyme activities (Spearman's correlation)		Binding vs. Enzyme activities (ROC-AUC)	
ZS predictor	t-statistics	p-value	t-statistics	p-value
Hamming distance	0.336	0.746	-3.352	**0.010**
EVmutation	-0.803	0.478	-0.116	0.924
ESM-2	0.467	0.660	0.521	0.668
ESM-IF	1.074	0.367	4.078	**0.008**
CoVES	1.032	0.371	3.116	**0.044**
Triad	1.534	0.205	5.036	**0.001**

Table B.3.12. *Simulated campaign outcome following the recommended ML strategies flowchart. Hard-to-navigate was defined as the average pairwise C-alpha distance of substituted residues less than or equal to 10 Å or if the sites were located in an enzyme active site. Good ZS prior was defined as using Hamming distance with EVmutation if more than 1,000 sequences were covered in the MSA, using Hamming distance with ESM-IF for binding interactions, and no good ZS prior otherwise. Low n_{total} / low evolvability was defined as n_{total} ≤ 480. Large search space / split into rounds was defined as if the landscape targeted four or more sites. Average maximum fitness achieved and fraction reaching global optimum with n_{total} = 480.*

Landscape	Hard-to-navigate?	Deep MSA?	Low n_{total} / low evolvability?	Large search space / split into rounds?	Strategy	Average maximum fitness achieved	Fraction reaching the global optimum
DHFR	Yes	Yes	Yes	No	Hamming distance + EVmutation ftMLDE	1.00	1.00
GB1	Yes	No	Yes	Yes	Hamming distance + ESM-IF ftALDE x 4	0.94	0.76
ParD2	No	-	Yes	No	MLDE	1.00	0.98
ParD3	No	-	Yes	No	MLDE	0.99	0.12
T7	Yes	No	Yes	No	Hamming distance ftMLDE	0.78	0.00
TEV	Yes	No	Yes	Yes	Hamming distance ftALDE x 4	0.42	0.02
TrpB3A	Yes	Yes	Yes	No	Hamming distance + EVmutation ftMLDE	1.00	1.00
TrpB3B	Yes	Yes	Yes	No		1.00	1.00
TrpB3C	Yes	Yes	Yes	No		1.00	1.00
TrpB3D	Yes	Yes	Yes	No		1.00	1.00
TrpB3E	Yes	Yes	Yes	No		1.00	1.00
TrpB3F	Yes	Yes	Yes	No		1.00	1.00
TrpB3G	Yes	Yes	Yes	No		1.00	1.00
TrpB3H	Yes	Yes	Yes	No		1.00	1.00
TrpB3I	Yes	Yes	Yes	No		1.00	1.00
TrpB4	Yes	Yes	Yes	Yes	Hamming distance + EVmutation ftALDE x 4	0.81	0.06

Table B.3.13. *Simulated campaign outcome following the recommended ML strategies flowchart. Hard-to-navigate was defined as the average pairwise C-alpha distance of substituted residues less than or equal to 10 Å or if the sites were located in an enzyme active site. Good ZS prior was defined as using Hamming distance with EVmutation if more than 1000 sequences were covered in the MSA, using Hamming distance with ESM-IF for binding interactions, and no good ZS prior otherwise. Low n_{total} / low evolvability was defined as n_{total} $\leq$480. Large search space / split into rounds was defined as if the landscape targeted four or more sites. Average maximum fitness achieved and fraction reaching global optimum with n_{total} = 2016.*

Landscape	Hard-to-navigate?	Deep MSA?	Low n_{total} / low evolvability?	Large search space / split into rounds?	Strategy	Average maximum fitness achieved	Fraction reaching the global optimum
DHFR	Yes	Yes	No	No	EVmutation ftMLDE	1.00	1.00
GB1	Yes	No	No	Yes	ESM-IF ftALDE x 4	0.99	0.96
ParD2	No	-	No	No	MLDE	1.00	1.00
ParD3	No	-	No	No	MLDE	0.99	0.28
T7	Yes	No	No	No	MLDE	0.96	0.34
TEV	Yes	No	No	Yes	ALDE x 4	0.52	0.00
TrpB3A	Yes	Yes	No	No	EVmutation ftMLDE	1.00	1.00
TrpB3B	Yes	Yes	No	No		1.00	1.00
TrpB3C	Yes	Yes	No	No		1.00	1.00
TrpB3D	Yes	Yes	No	No		1.00	1.00
TrpB3E	Yes	Yes	No	No		1.00	1.00
TrpB3F	Yes	Yes	No	No		1.00	1.00
TrpB3G	Yes	Yes	No	No		1.00	1.00
TrpB3H	Yes	Yes	No	No		1.00	1.00
TrpB3I	Yes	Yes	No	No		1.00	1.00
TrpB4	Yes	Yes	No	Yes	EVmutation ftALDE x 4	1.00	0.98

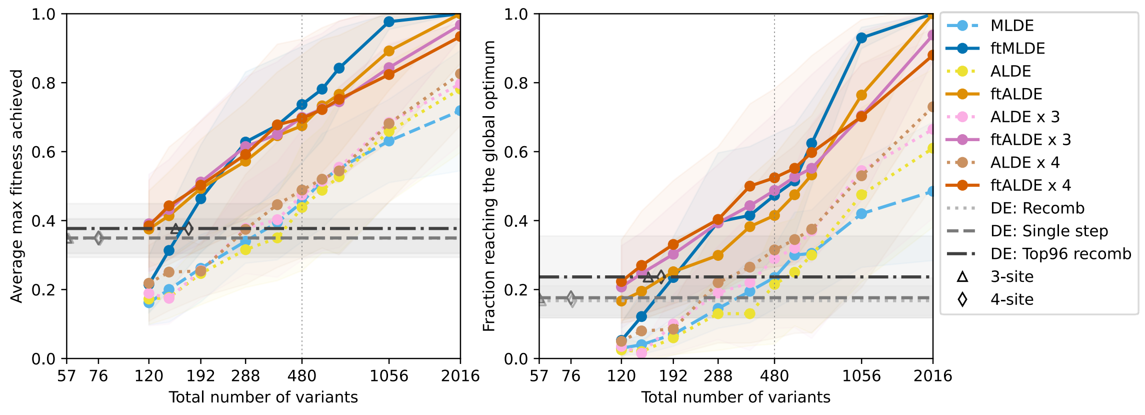
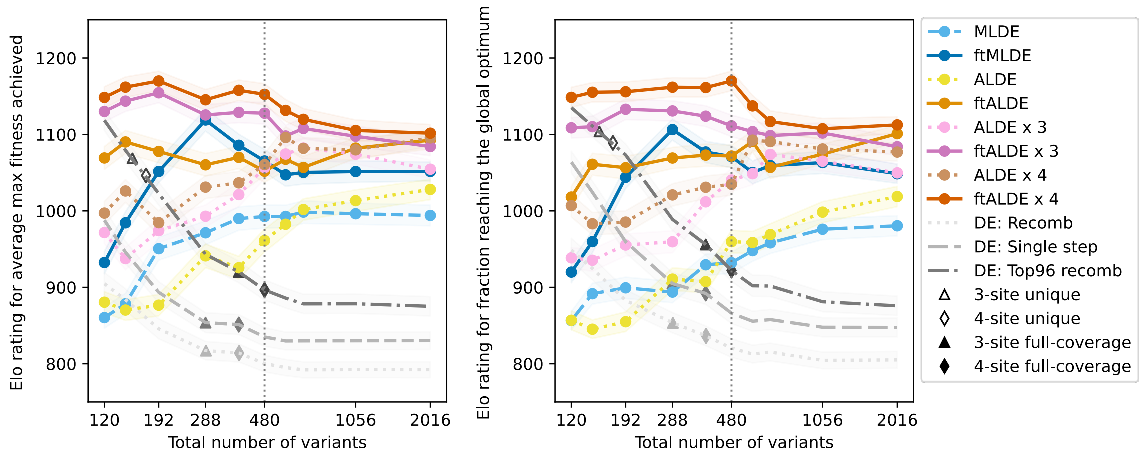


Figure B.3.1. *DE, MLDE, ALDE, and focused training performance averaged across four landscapes with fewer than 1% active variants, related to* ***Figure 3.2a****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*



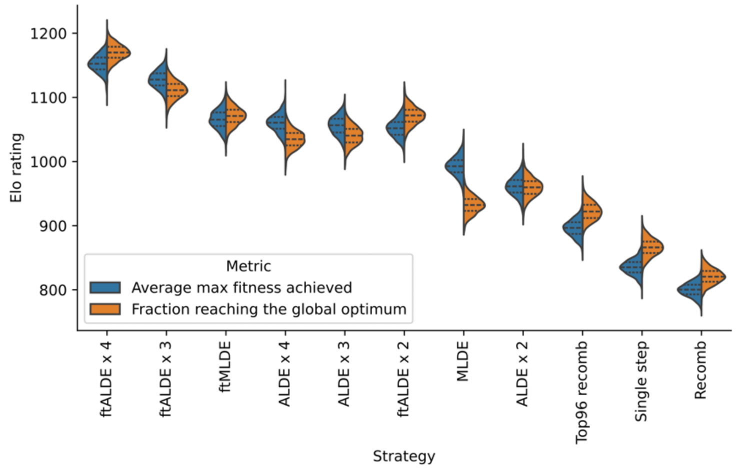


Figure B.3.2. *Elo ratings for DE, MLDE, ALDE, and focused training strategies across all 16 landscapes, related to* ***Figures 3.2a*** *and* ***B.3.1****. The top panel shows Elo rating across all total sample sizes for each ML-based method. The bottom panel shows a violin plot for total sample size of 480 ($n_{total} = 480$) across both metrics, with inner black lines marking quartiles (25th, 50th/median, and 75th percentiles). Both panels reflect results from 1,000 bootstrap resampling iterations; shading in the top panel indicates standard deviation from these bootstraps.*

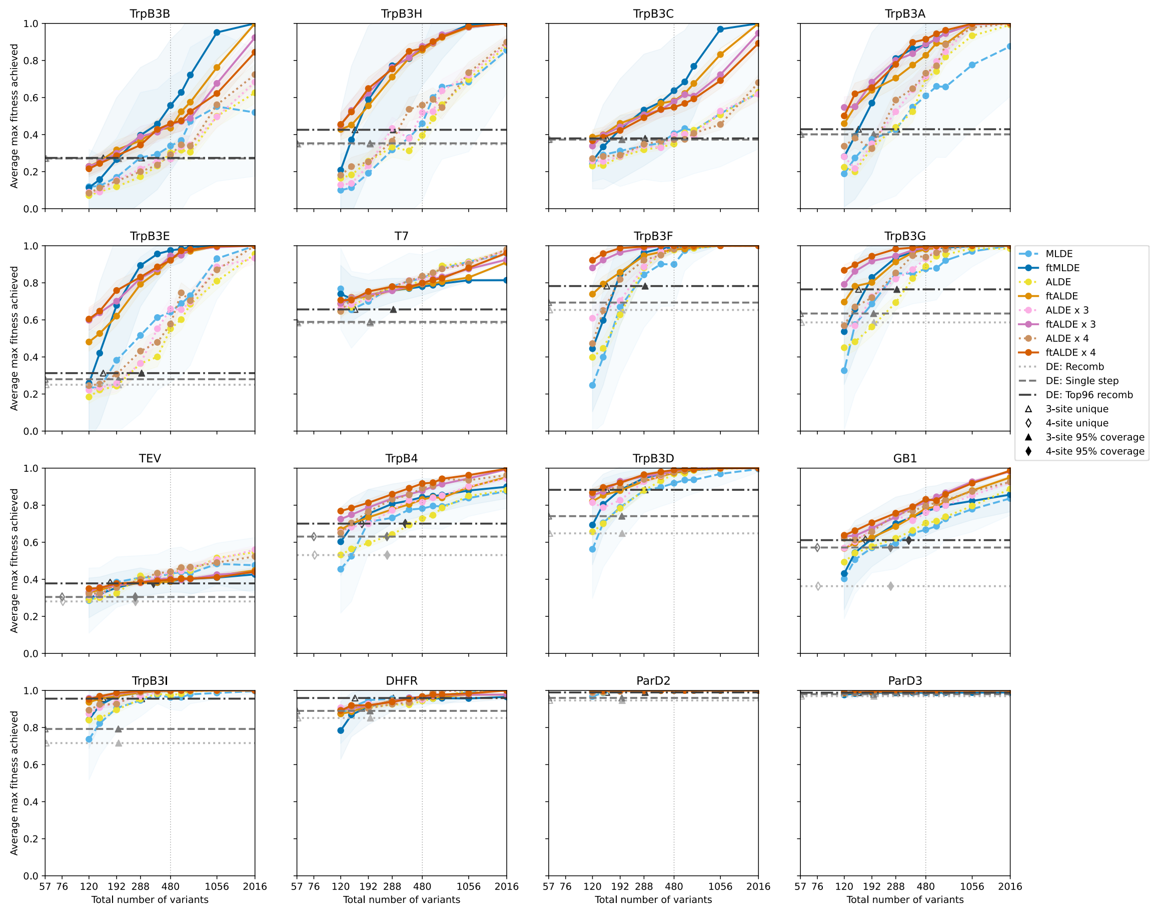


Figure B.3.3. *Performance of DE, MLDE, ALDE, and focused training, measured by the average maximum fitness achieved for each of the 16 landscapes individually, related to* ***Figures 3.2a*** *and* ***B.3.1****. The hollow triangle and diamond indicate the total number of unique variants sampled for DE, where* $n_{total} = n_{sample} + n_{test}$ *and* $n_{sample} = 19 \times n_{site} + 1$ *(****Box 3.1****;* ***Appendix B.2 Methods****). The solid triangle and diamond indicate the total number of variants screened to achieve 95% variant space coverage, given by* $n_{total}{}^{\dagger} = n_{screen} + n_{test}$*, where* $n_{screen} = -n_{codon} \times \ln(1 - \mathrm{p}) \times n_{site}$ *and* p *is the desired library coverage.*[19,20] *The number of codons,* n_{codon}*, was set to 22 based on the 22-coden trick. In practice,* n_{codon} *varies depending on the SSM library generation methods.*[19] *Shading indicates the standard deviation, averaged across 50 replicates.*

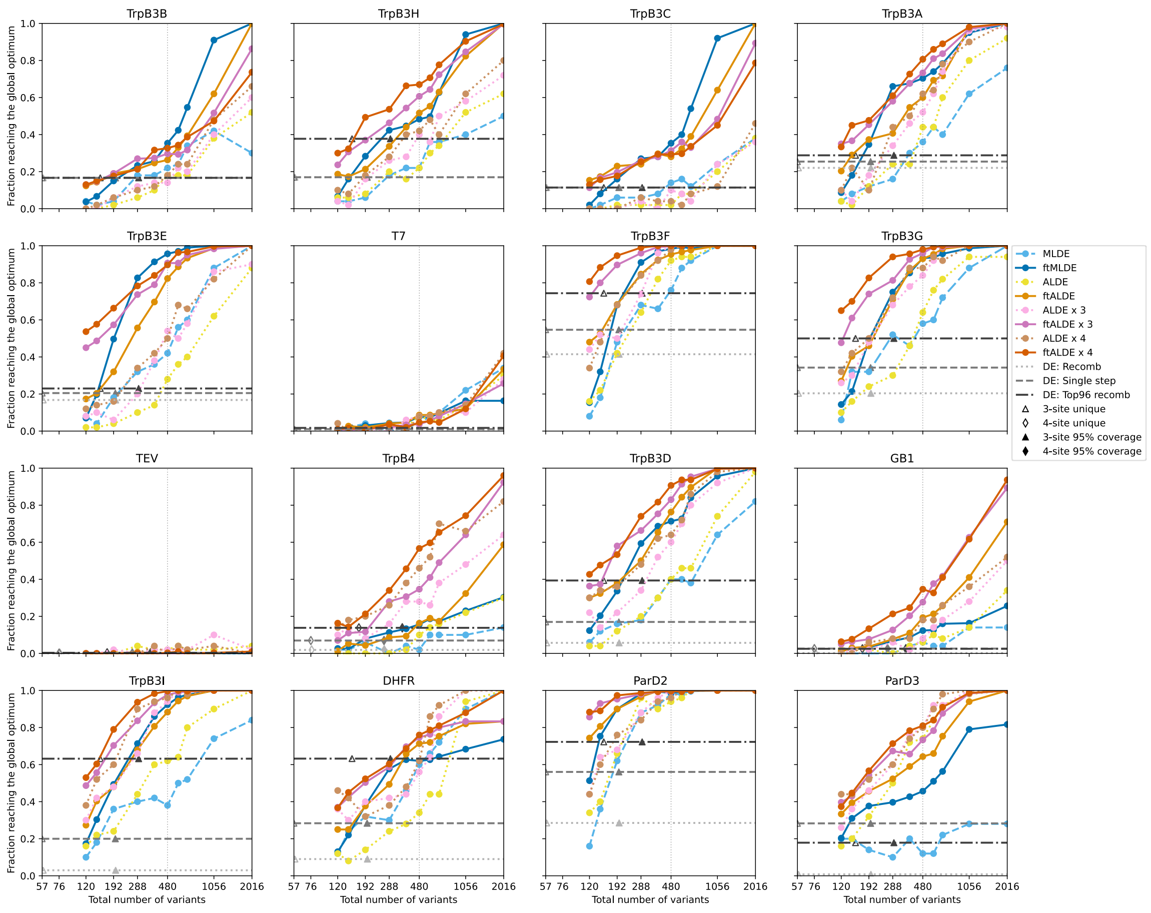


Figure B.3.4. *Performance of DE, MLDE, ALDE, and focused training, measured by the fraction reaching the global optimum for each of the 16 landscapes individually over 50 replicates, related to* ***Figures 3.2a*** *and* ***B.3.1****. The hollow triangle and diamond indicate the total number of unique variants sampled for DE, where* $n_{total} = n_{sample} + n_{test}$ *and* $n_{sample} = 19 \times n_{site} + 1$ *(**Box 3.1**; **Appendix B.2 Methods**). The solid triangle and diamond indicate the total number of variants screened to achieve 95% variant space coverage, given by* $n_{total}{}^{\dagger} = n_{screen} + n_{test}$*, where* $n_{screen} = -n_{codon} \times \ln(1 - \mathrm{p}) \times n_{site}$ *and* p *is the desired library coverage.*[19,20] *The number of codons,* n_{codon}*, was set to 22 based on the 22-coden trick. In practice,* n_{codon} *varies depending on the SSM library generation methods.*[19]

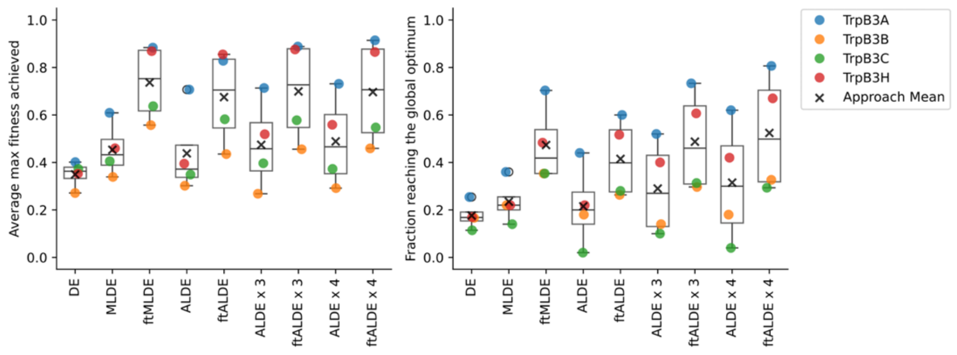


Figure B.3.5. *Single-step DE, MLDE, ALDE, and focused training results broken down by four landscapes with fewer than 1% active variants. A total sample size of 480 was used for all ML strategies across both metrics, related to* ***Figure 3.2b****.*

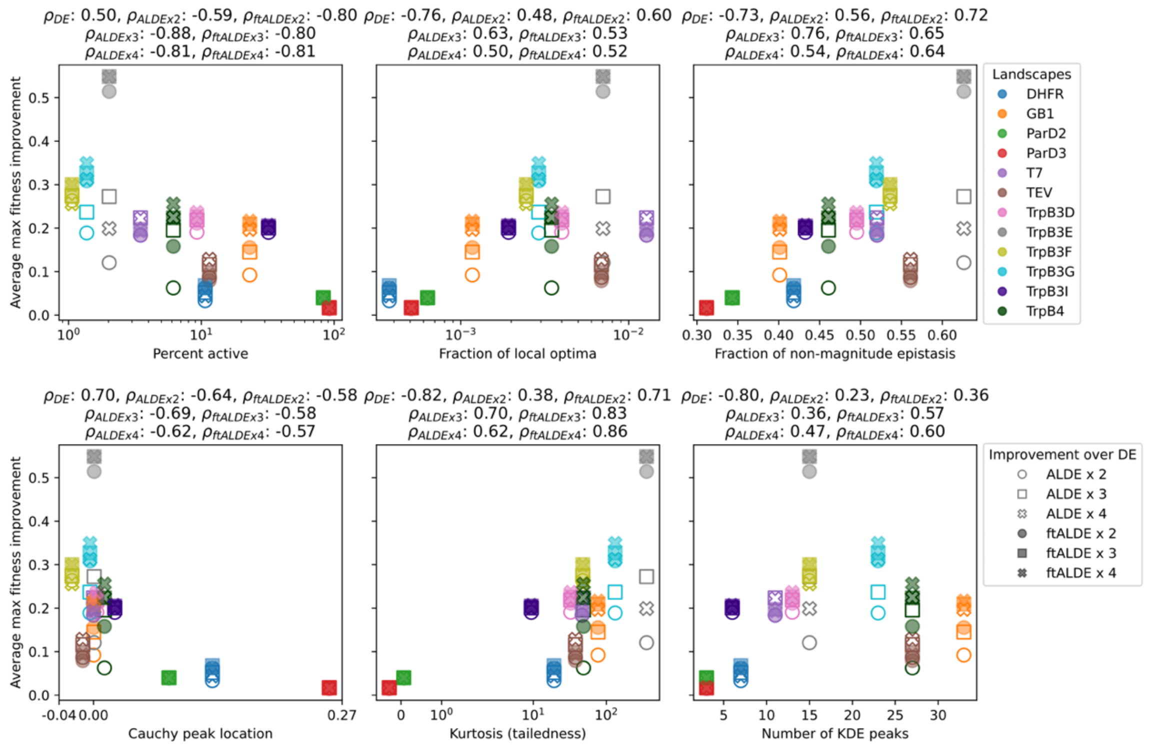


Figure B.3.6. *Correlation of ALDE and ftALDE performance improvement (the average maximum fitness of the top 96 predicted variants by ALDE and ftALDE over single-step DE, y-axis) with six landscape attributes (x-axis), related to* ***Figure 3.2c****.*

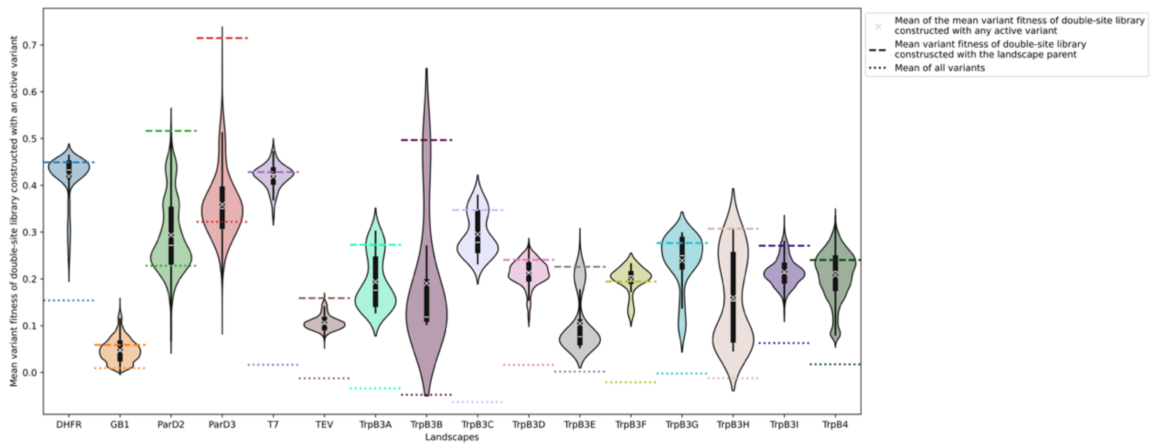


Figure B.3.7. *Mean variant fitness of double-site library (Hamming distance of two) from active variant as the parent, related to Hamming distance in* ***Figure 3.3***.

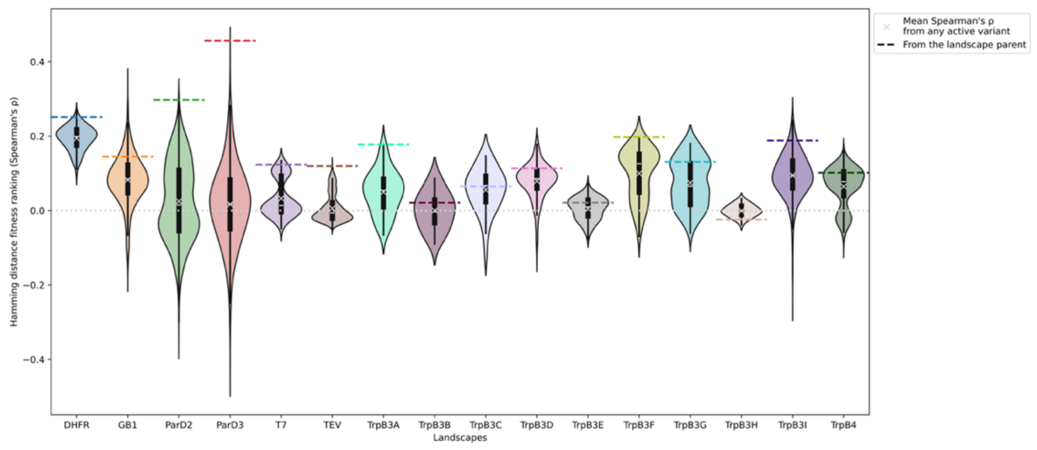


Figure B.3.8. *Hamming distance fitness ranking using any active variant as the parent, related to Hamming distance in* ***Figure 3.3***. *The dotted line indicates random predictions.*

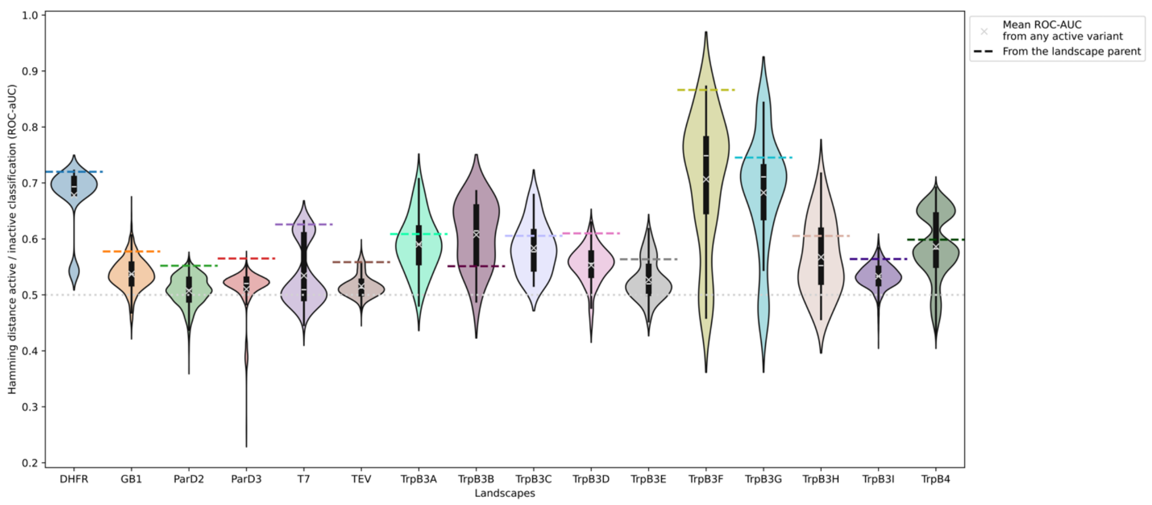


Figure B.3.9. *Hamming distance active/inactive variant classification using any active variant as the parent, related to Hamming distance in* ***Figure 3.3****. The dotted line indicates random predictions.*

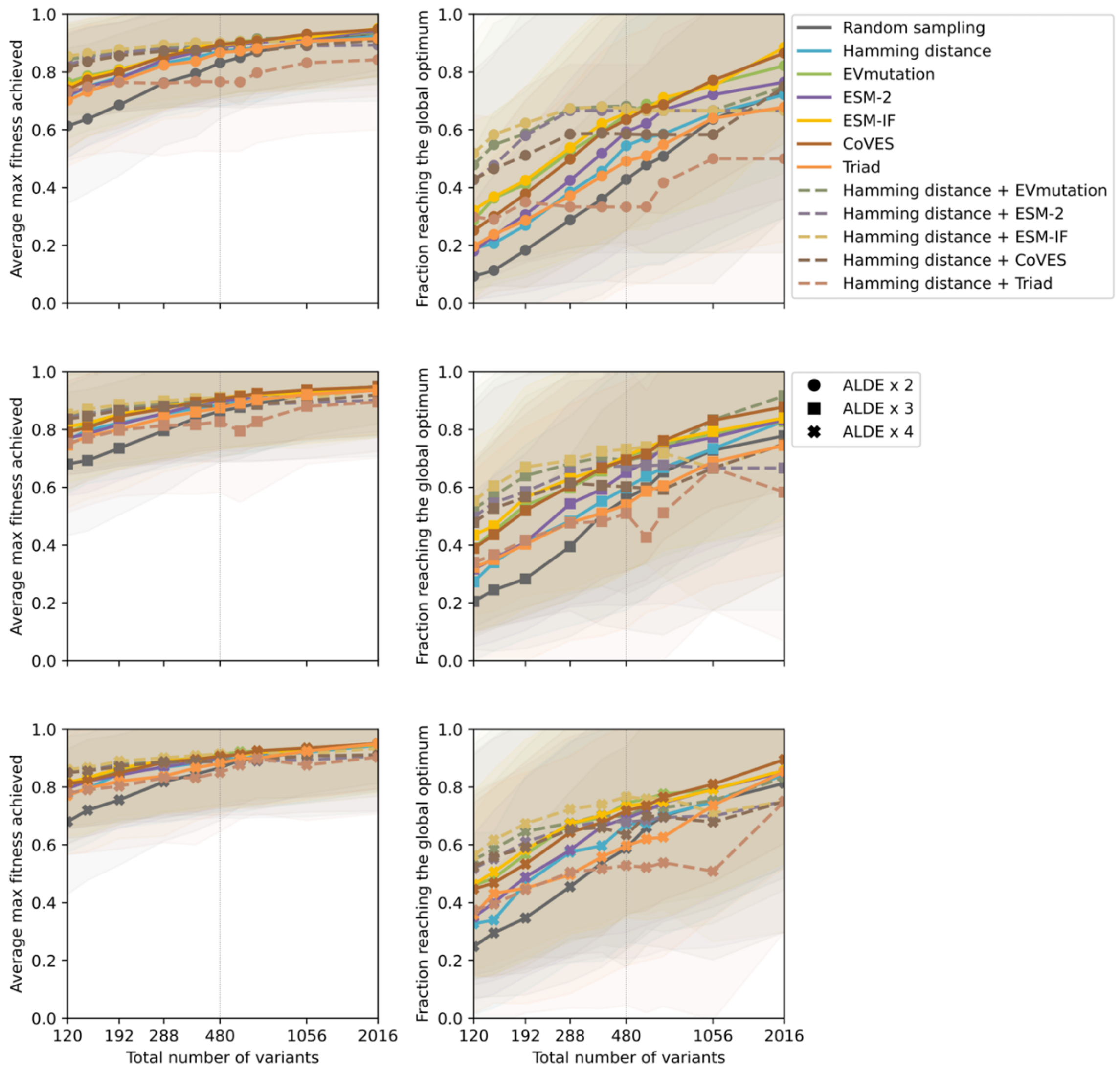


Figure B.3.10. *Multiple rounds of ftALDE averaged across 12 landscapes with more than 1% active variants, related to* ***Figure 3.3e****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

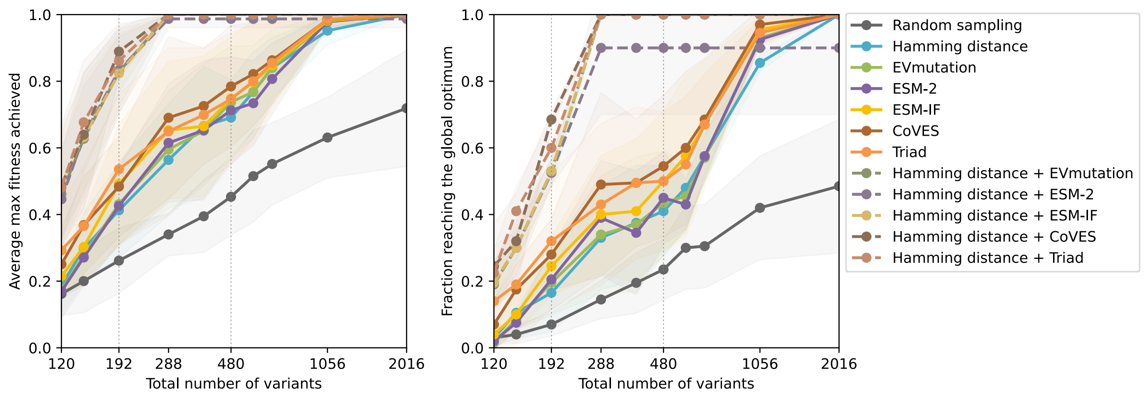


Figure B.3.11. *ftMLDE with Hamming distance-ensembled ZS predictors averaged across four landscapes with fewer than 1% active variants, related to* ***Figure 3.3e****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

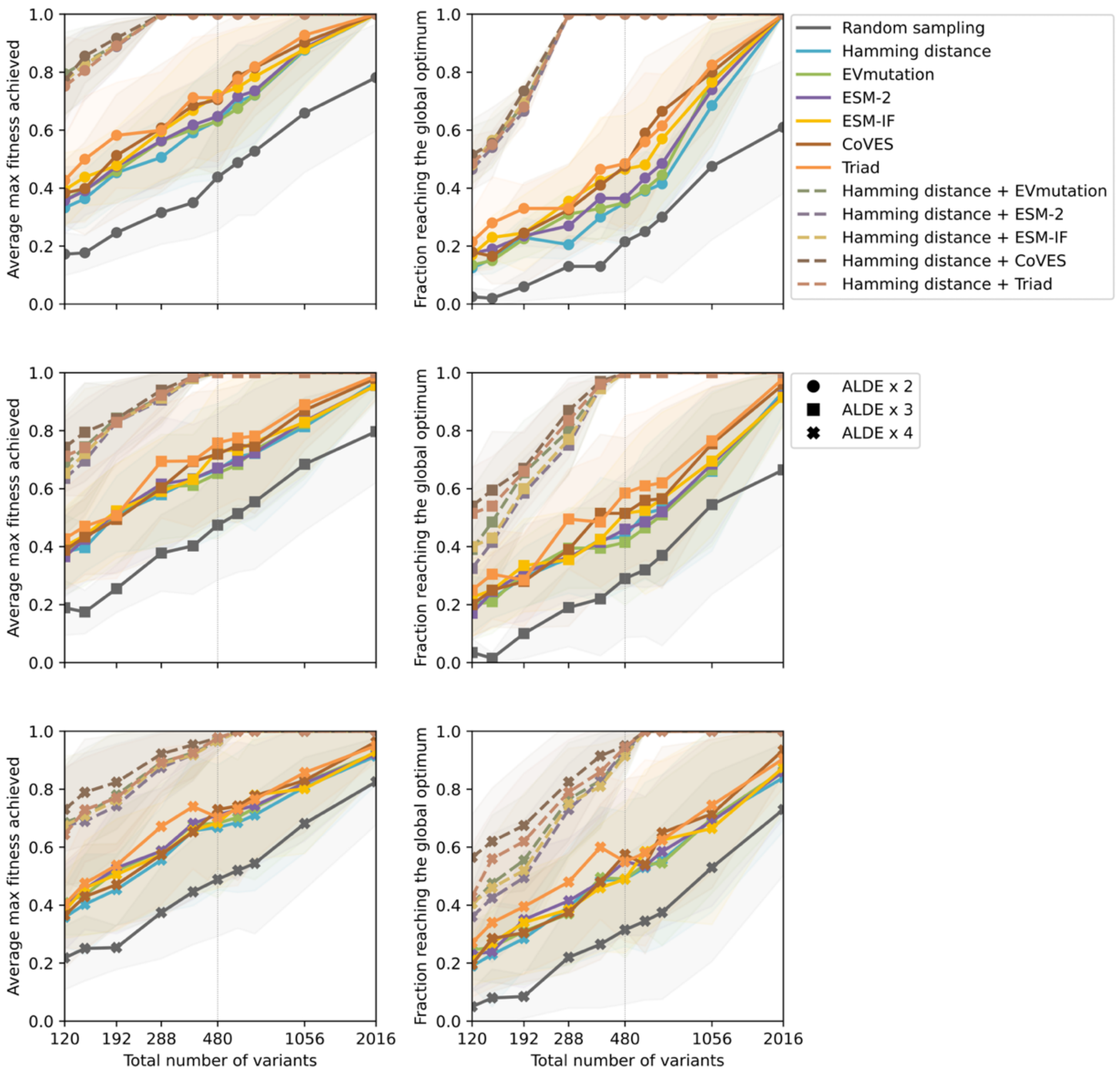


Figure B.3.12. *Multiple rounds of ftALDE averaged across four landscapes with fewer than 1% active variants, related to* ***Figure 3.3e****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

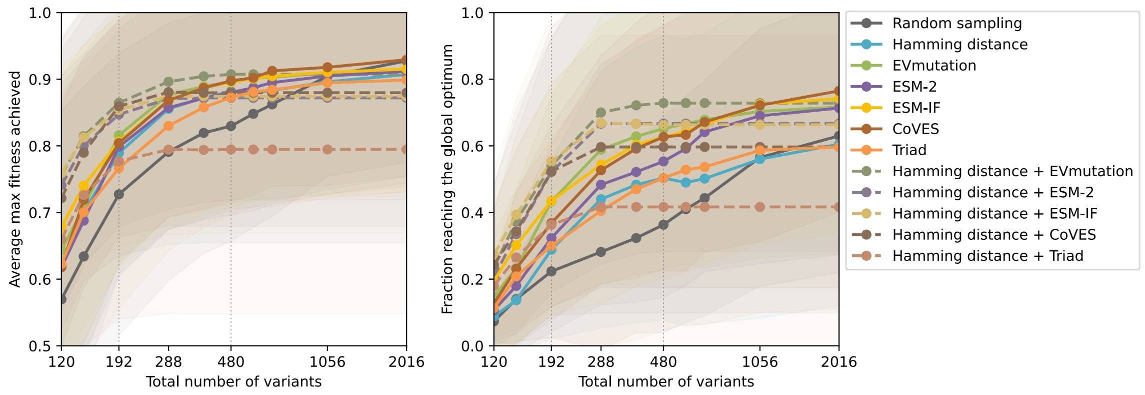


Figure B.3.13. *ftMLDE with Hamming distance-ensembled ZS predictors, averaged across 12 landscapes with more than 1% active variants, related to* ***Figure 3.3e****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

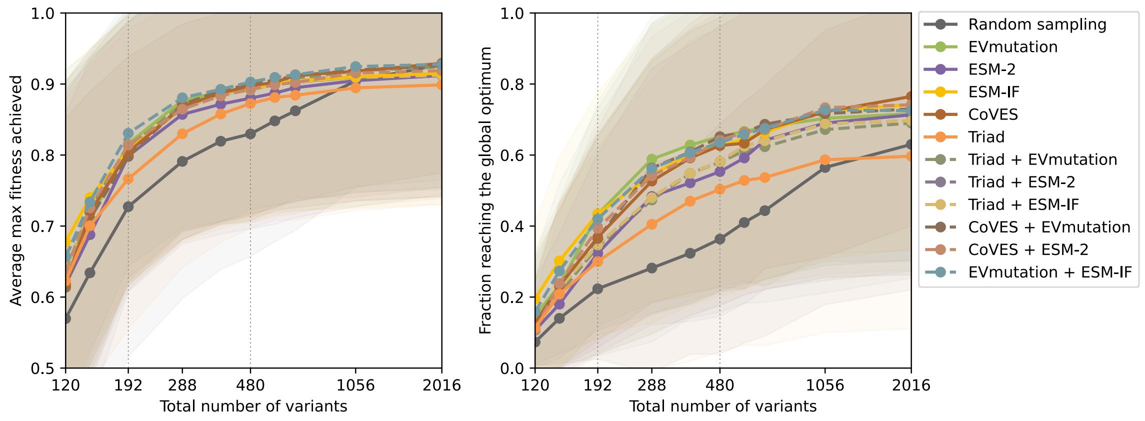


Figure B.3.14. *ftMLDE with Triad-ensembled ZS predictors, CoVES-ensembled ZS predictors, or ESM-IF and EVmutation ensemble, averaged across 12 landscapes with more than 1% active variants, related to* ***Figure 3.3e****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

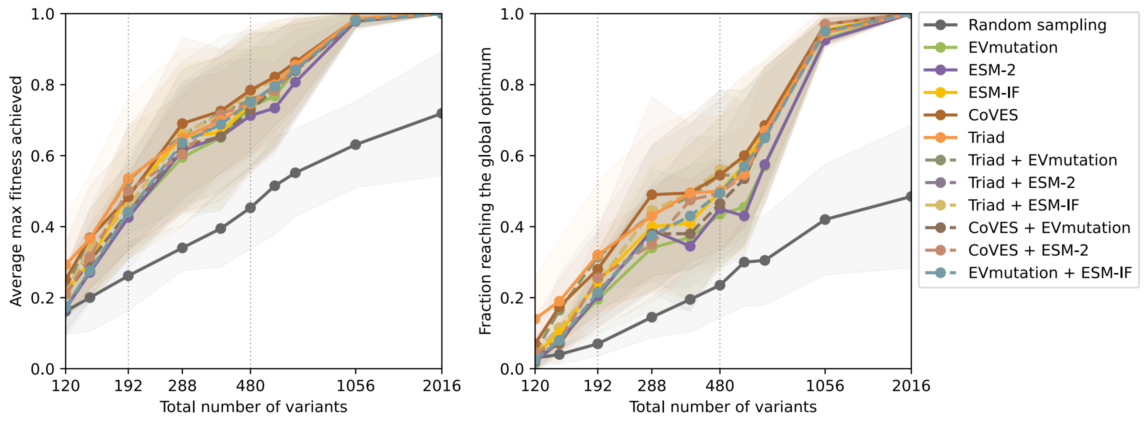


Figure B.3.15. *ftMLDE with Triad-ensembled ZS predictors, CoVES-ensembled ZS predictors, or ESM-IF and EVmutation ensemble, averaged across four landscapes with fewer than 1% active variants, related to* ***Figure 3.3e****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

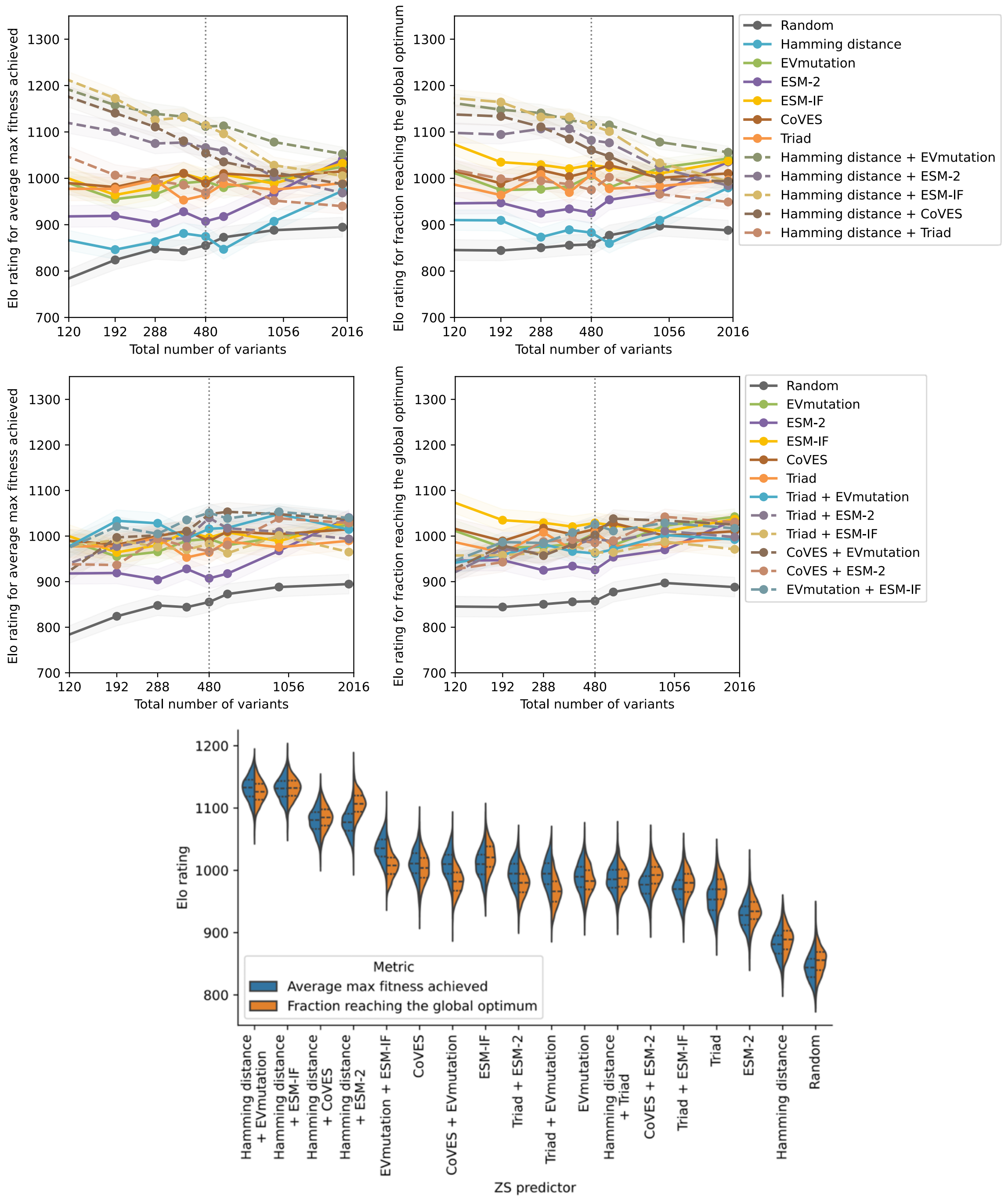


Figure B.3.16. *Elo ratings for different ZS predictors used to guide focused training strategies for all 16 landscapes, related to* ***Figures 3.3e*****,** ***B.3.10–B.3.15****. The top and middle panels show Elo rating across all total sample sizes for focused-training with Hamming distance-based ensembles or other high-performing but orthogonal ZS predictor ensembles. The bottom panel shows a violin plot for 480 ($n_{total} = 480$) across both metrics, with inner black lines marking quartiles (25th, 50th/median, and 75th percentiles). All panels reflect results from 1,000 bootstrap resampling iterations; shading in the top and middle panels indicate standard deviation from these bootstraps.*

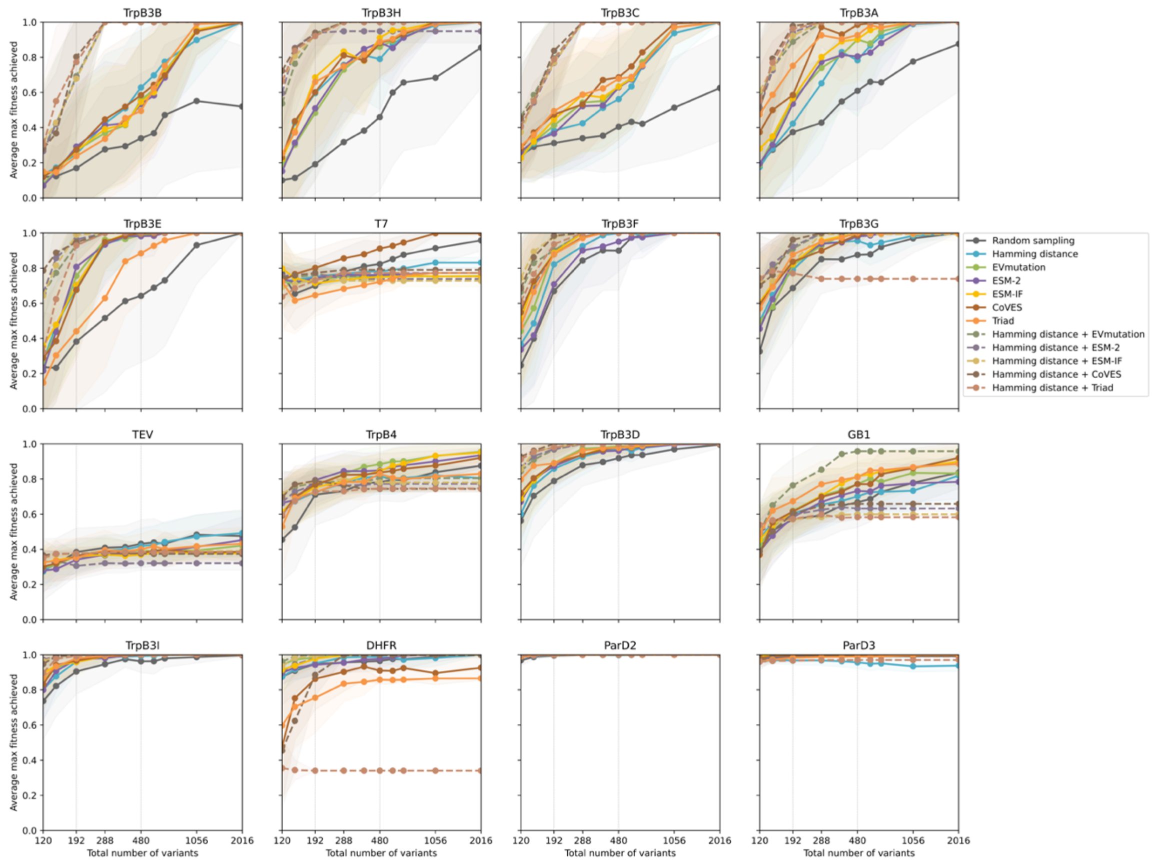


Figure B.3.17. *ftMLDE with Hamming distance-ensembled ZS predictors, measured by the average maximum fitness achieved for each of the 16 landscapes individually, related to* ***Figures 3.3e****,* ***B.3.11****, and* ***B.3.13****. Shading indicates the standard deviation, averaged across 50 replicates.*

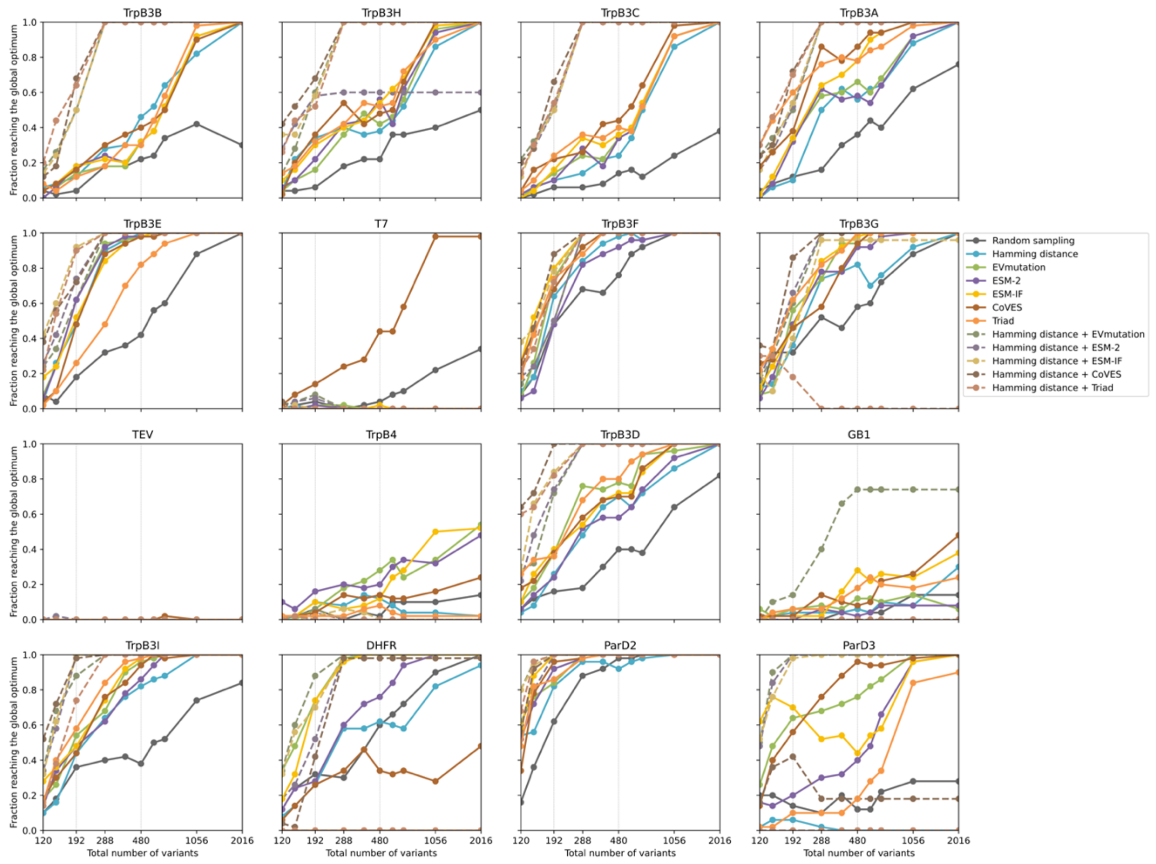


Figure B.3.18. *ftMLDE with Hamming distance-ensembled ZS predictors, measured by the average maximum fitness achieved for each of the 16 landscapes individually, related to* ***Figures 3.3e***, ***B.3.11***, *and* ***B.3.13***. *Results were calculated from 50 replicates.*

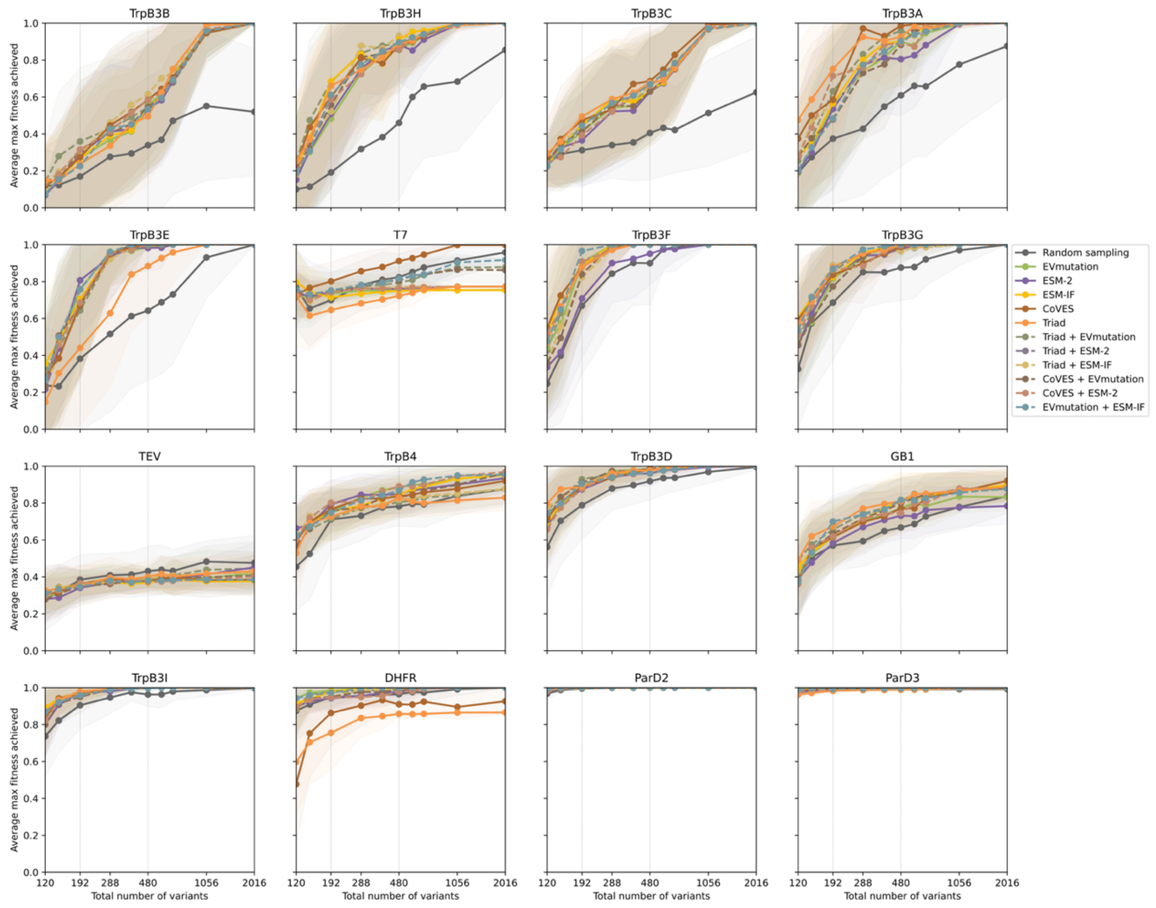


Figure B.3.19. *ftMLDE with Triad-ensembled ZS predictors, CoVES-ensembled ZS predictors, or ESM-IF and EVmutation ensemble, measured by the average maximum fitness achieved for each of the 16 landscapes individually, related to* ***Figures 3.3e****,* ***B.3.14****, and* ***B.3.15****. Shading indicates the standard deviation, averaged across 50 replicates.*

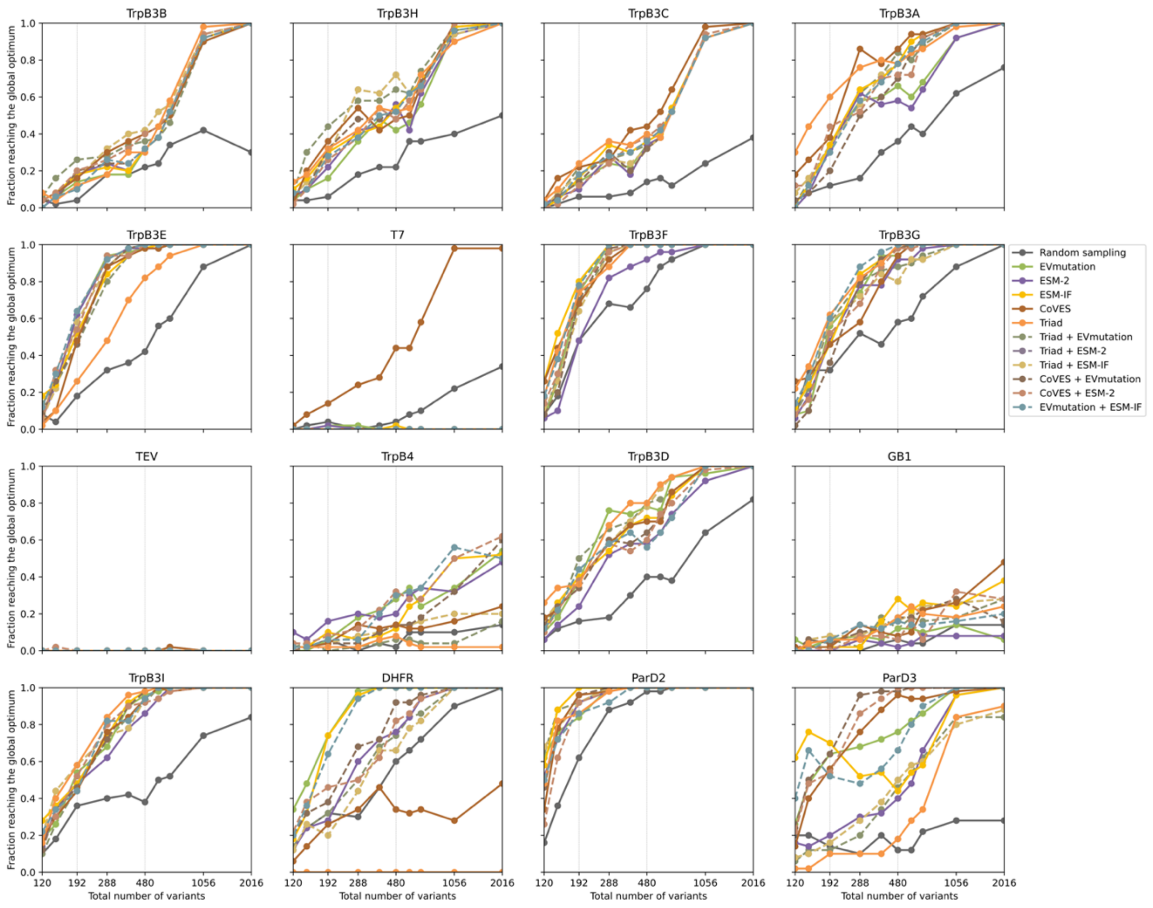


Figure B.3.20. *ftMLDE with Triad-ensembled ZS predictors, CoVES-ensembled ZS predictors, or ESM-IF and EVmutation ensemble, measured by the fraction reaching the global optimum for each of the 16 landscapes individually, related to* ***Figures 3.3e****,* ***B.3.14****, and* ***B.3.15****. Results were calculated from 50 replicates.*

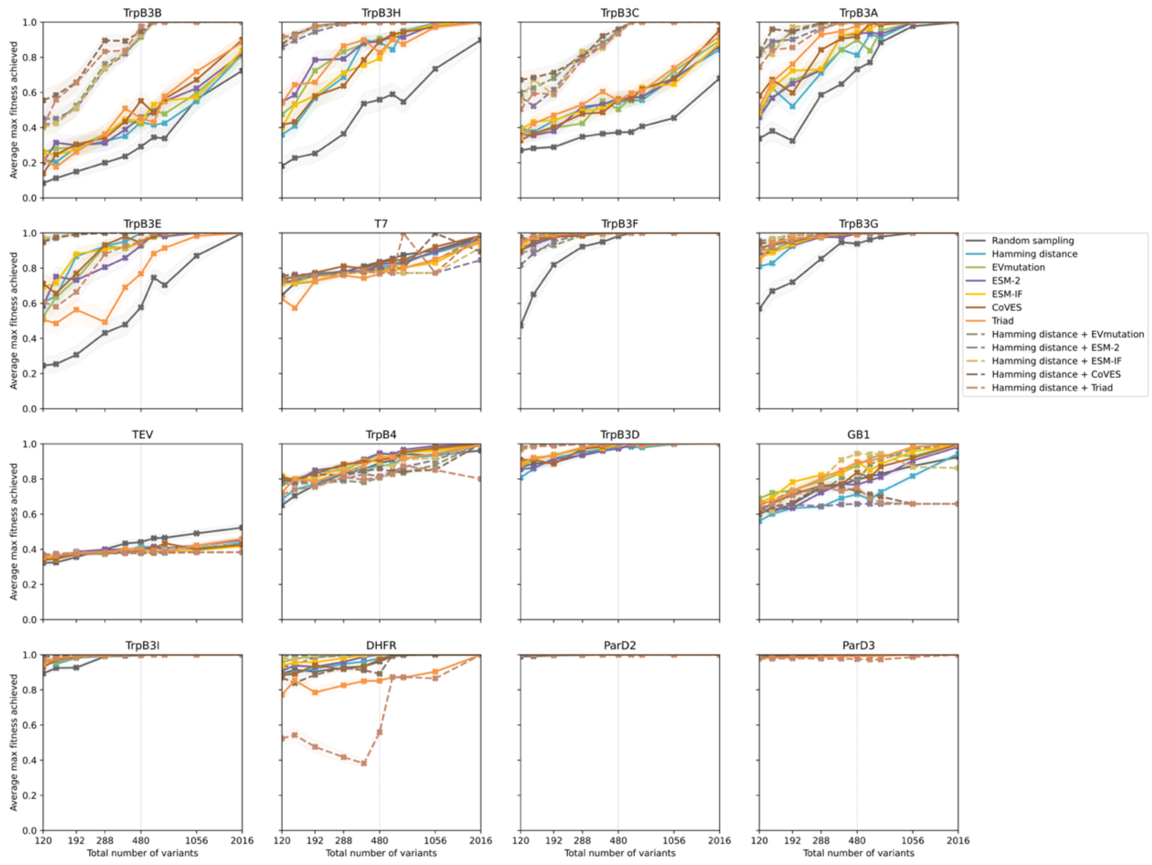


Figure B.3.21. *ftALDE x 4 with Hamming distance-ensembled ZS predictors, measured by the average maximum fitness achieved for each of the 16 landscapes individually, related to* ***Figures 3.3e****,* ***B.3.10****, and* ***B.3.12****. Shading indicates the standard deviation, averaged across 50 replicates.*

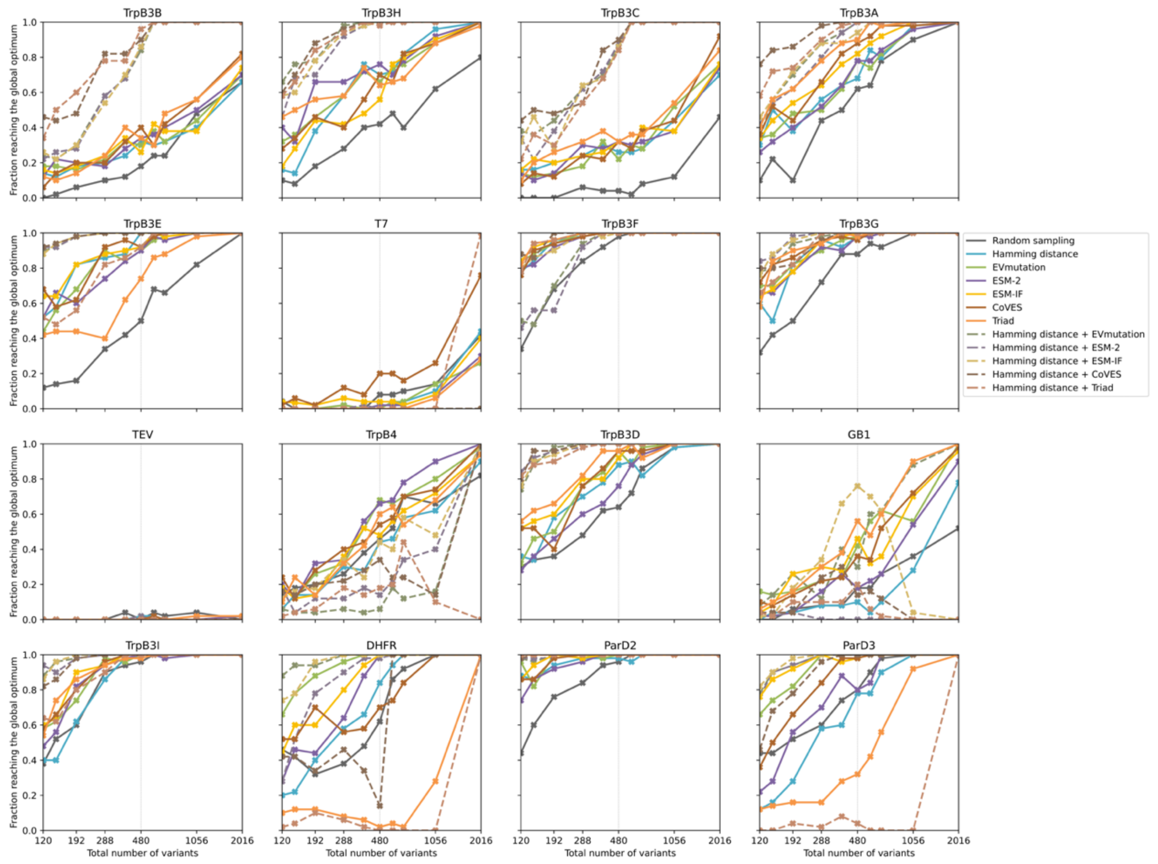


Figure B.3.22. *ftALDE x 4 with Hamming distance-ensembled ZS predictors, measured by the fraction reaching the global optimum for each of the 16 landscapes individually, related to* ***Figures 3.3e****,* ***B.3.10****, and* ***B.3.12****. Results were calculated from 50 replicates.*

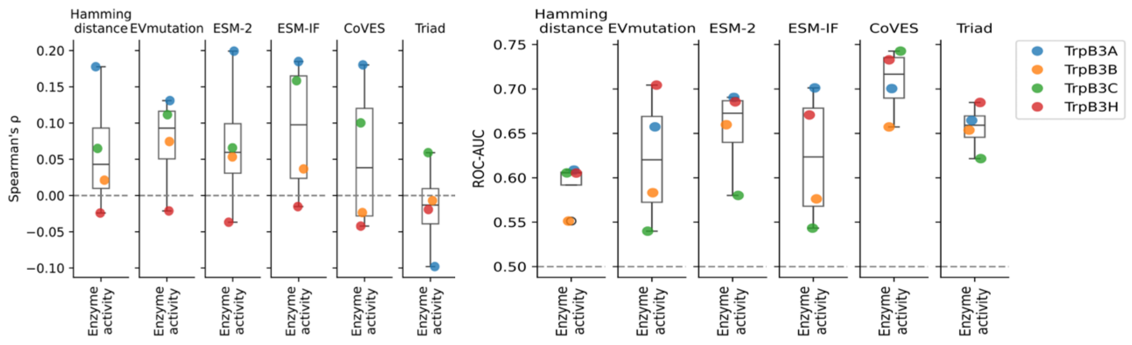


Figure B.3.23. *ZS predictor fitness value ranking (left) and active/inactive variant classification (right) for four landscapes with fewer than 1% active variants, related to* ***Figure 3.4a****.*

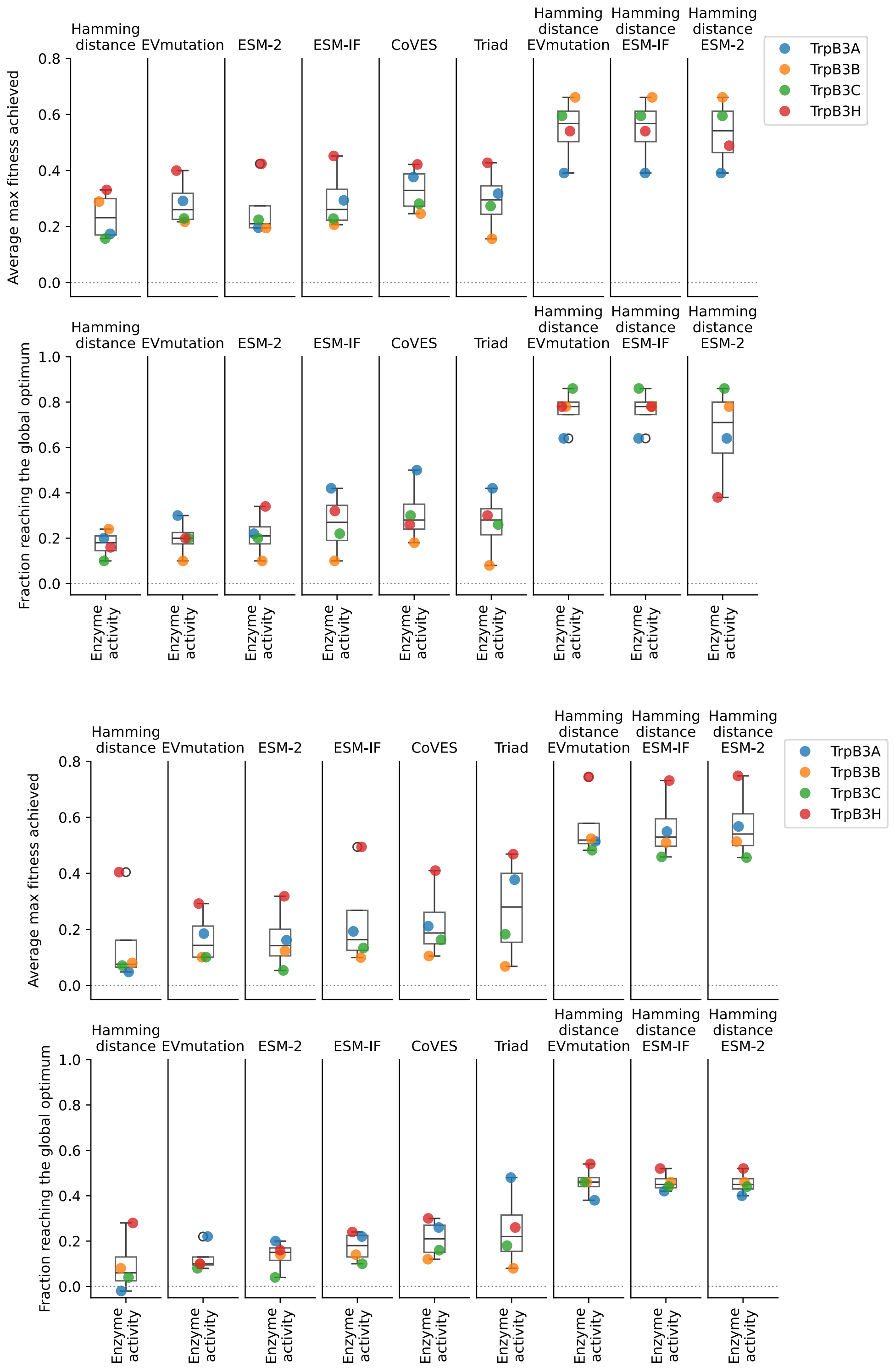
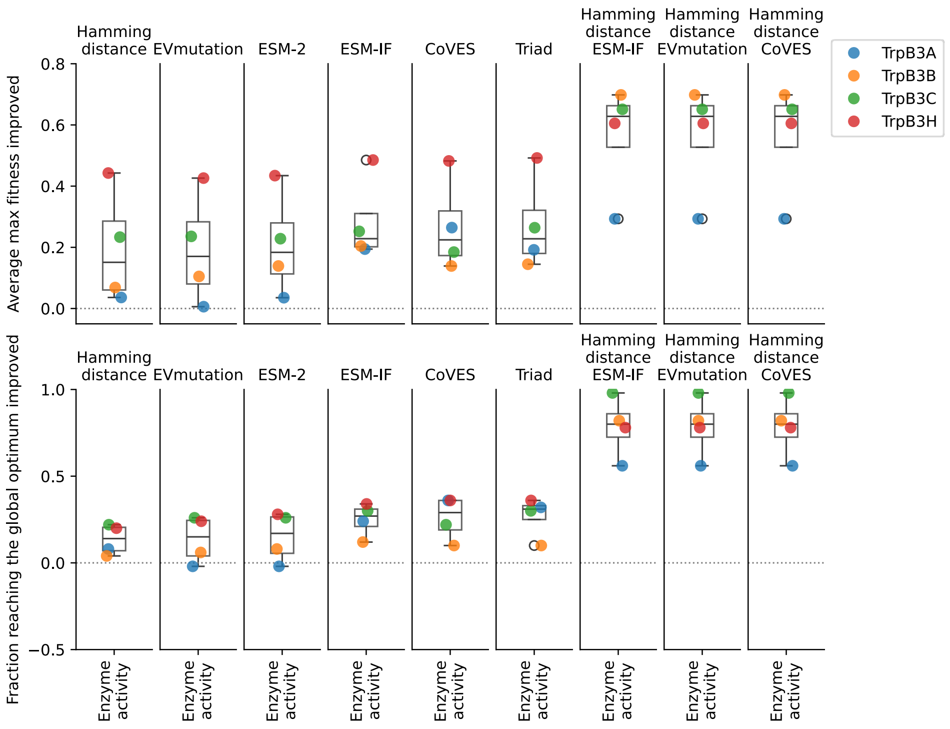


Figure B.3.24. *Effects of focused training for ftMLDE with a total sample size of 480 (384 training and 96 testing, top panel) and 192 (96 training and 96 testing, bottom panel) for four landscapes with fewer than 1% active variants, related to **Figure 3.4b**.*



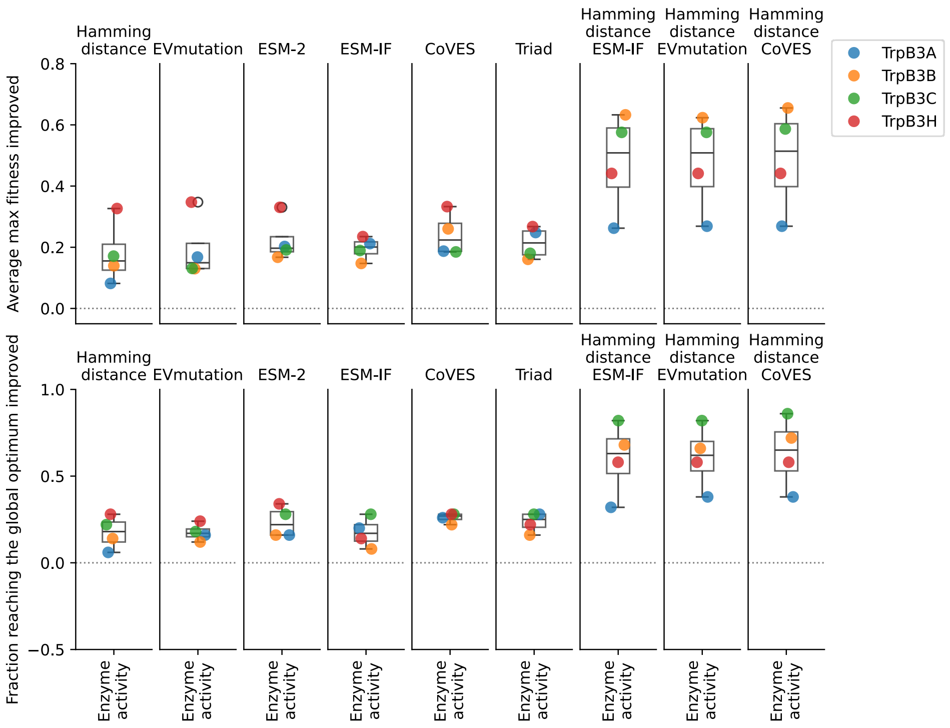
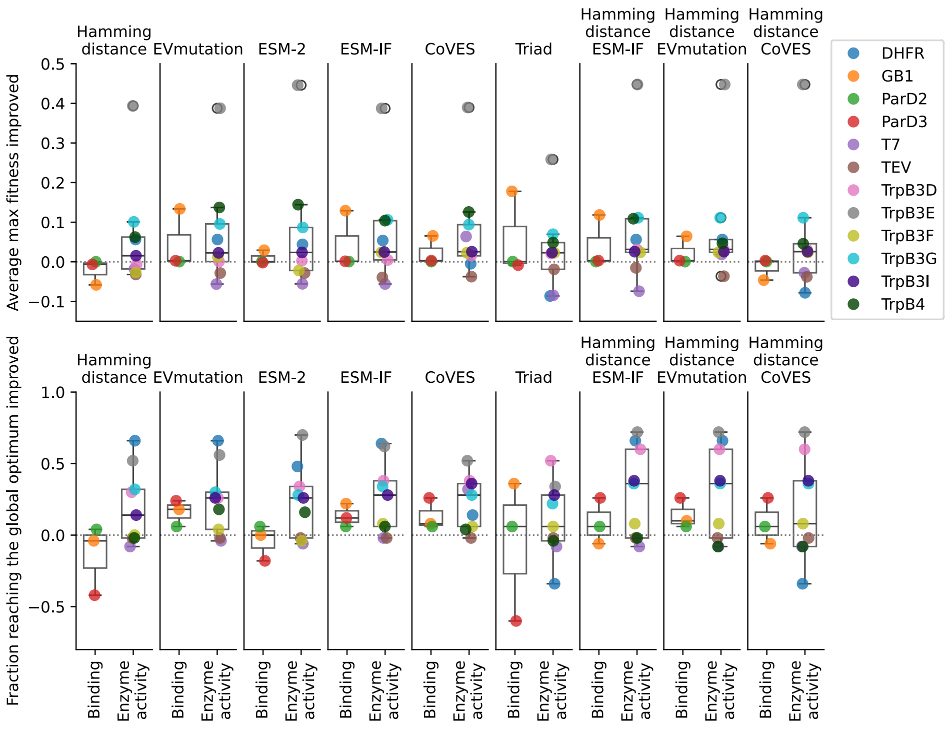


Figure B.3.25. *Effects of focused training for two (top panel) and four (bottom panel) rounds of ftALDE with a total sample size of 480 for four landscapes with fewer than 1% active variants, related to* ***Figure 3.4b****.*



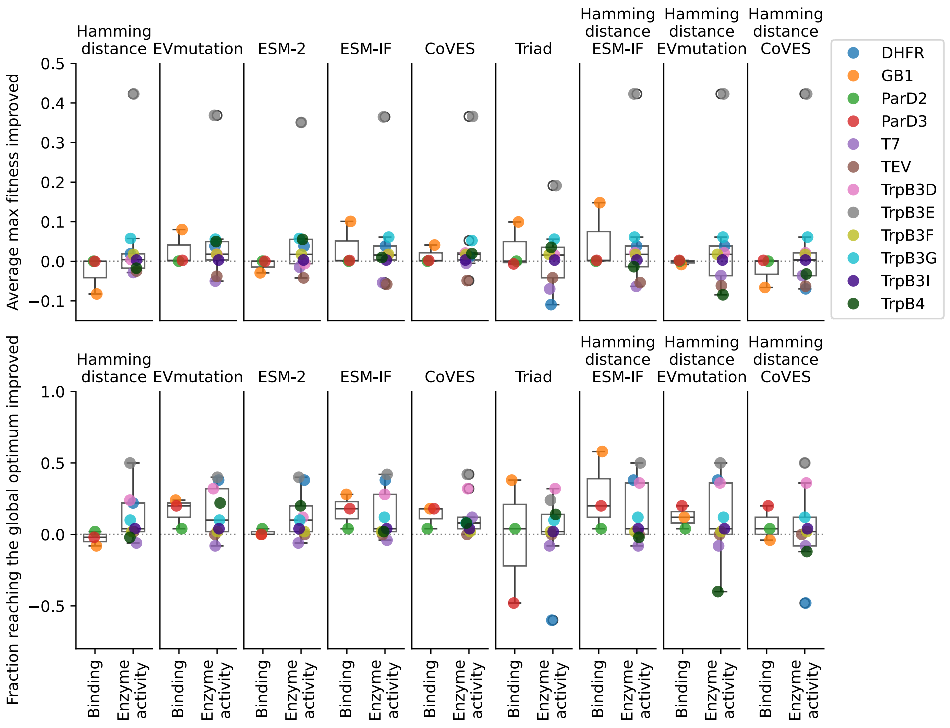
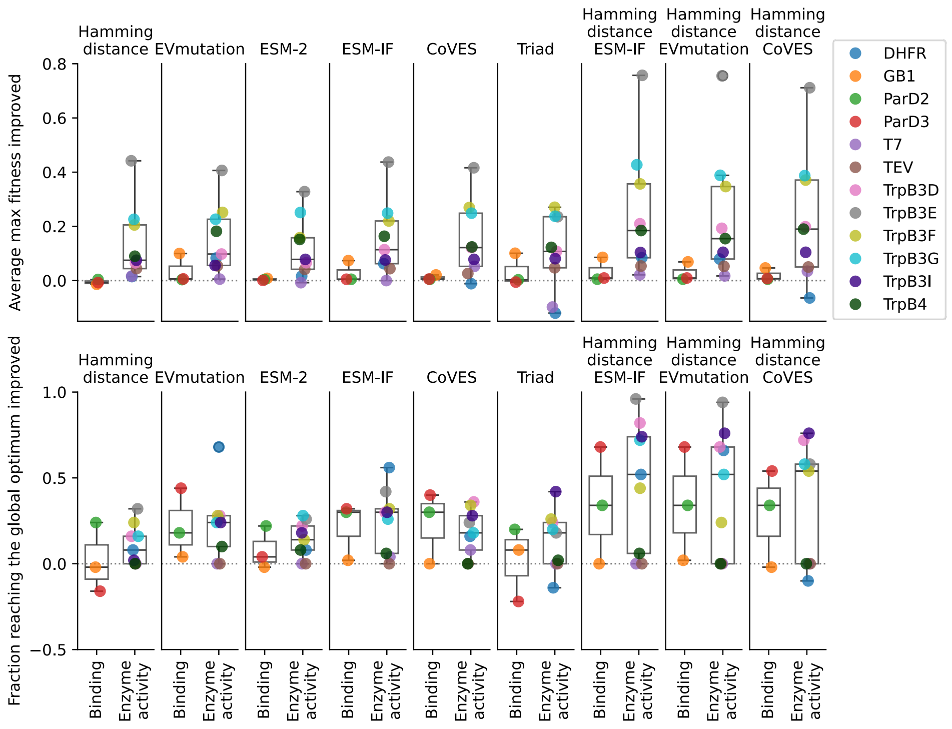


Figure B.3.26. *Effects of focused training for two (top panel) and four (bottom panel) rounds of ftALDE with a total sample size of 480 for 12 landscapes with at least 1% active variants, related to* ***Figure 3.4b****.*



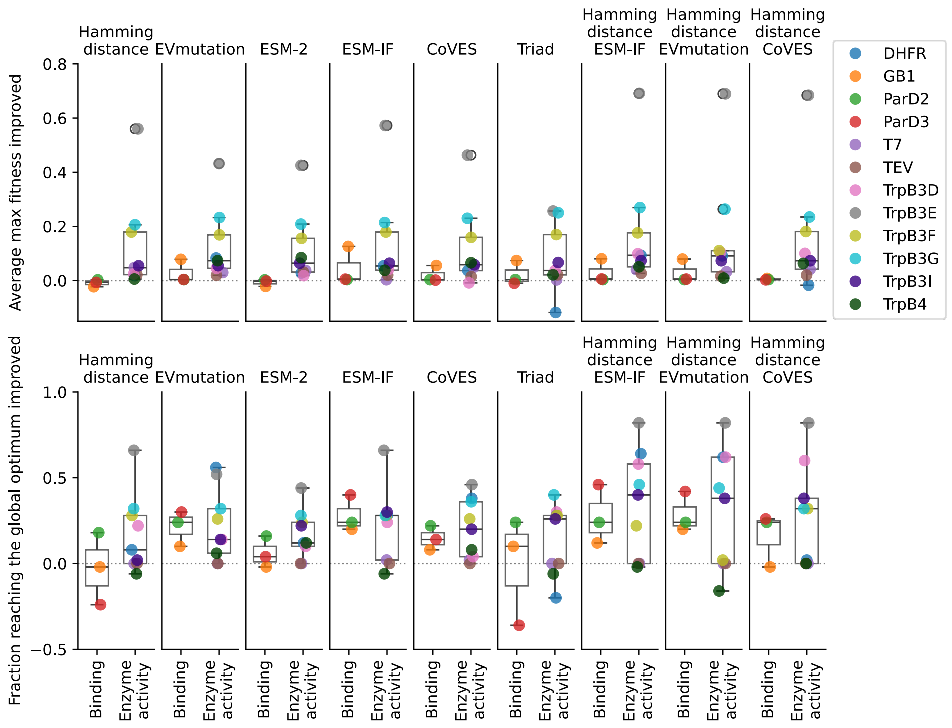


Figure B.3.27. *Effects of focused training for two (top panel) and four (bottom panel) rounds of ftALDE with a total sample size of 192 for 12 landscapes with at least 1% active variants, related to* ***Figure 3.4b****.*

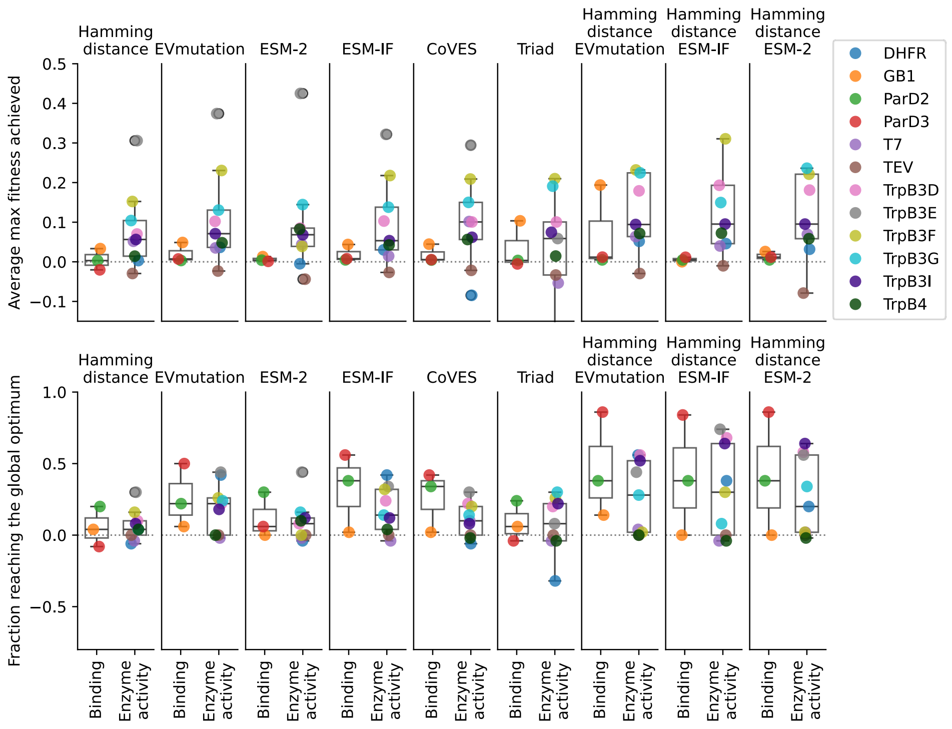


Figure B.3.28. *Effects of focused training for ftMLDE with a total sample size of 192 (96 training and 96 testing) for 12 landscapes with at least 1% active variants, related to* ***Figure 3.4b****.*

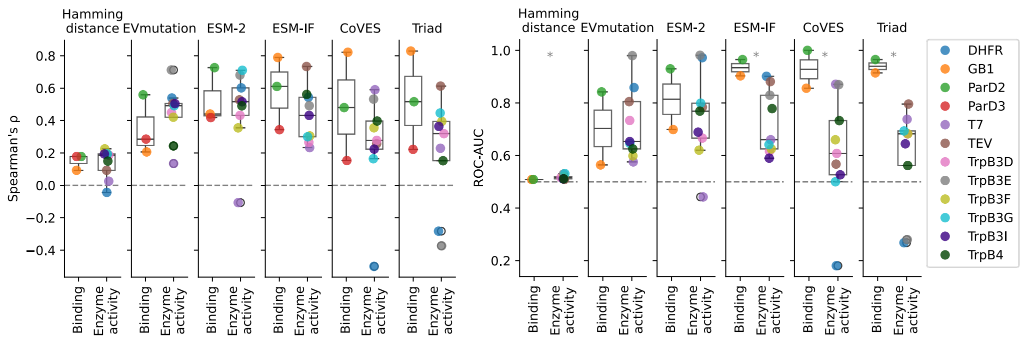


Figure B.3.29. *ZS predictor for single substitution fitness value ranking (left) and active/inactive variant classification (right) for 12 landscapes with at least 1% active variants, related to the* ***Chapter 3 Discussion section*** *and* ***Figure 3.4a****. Statistical significance (p-value <0.05) is indicated as* *******.

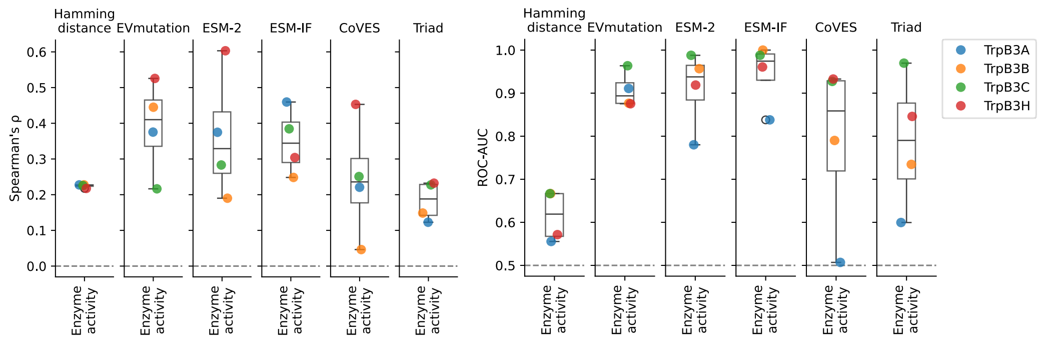


Figure B.3.30. *ZS predictor for single substitution fitness value ranking (left) and active/inactive variant classification (right) for landscapes with fewer than 1% active variants, related to the* ***Chapter 3 Discussion section*** *and* ***Figure 3.4a****.*

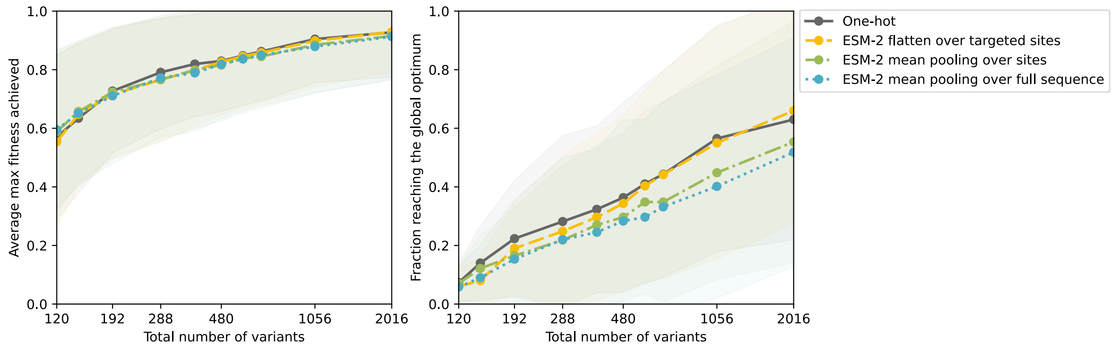


Figure B.3.31. *Encoding strategies for MLDE performance, averaged across 12 landscapes with at least 1% active variants. Comparison of learned embeddings from the protein language model ESM-2 using different pooling methods vs. one-hot encoding flattened over the substitution sites, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

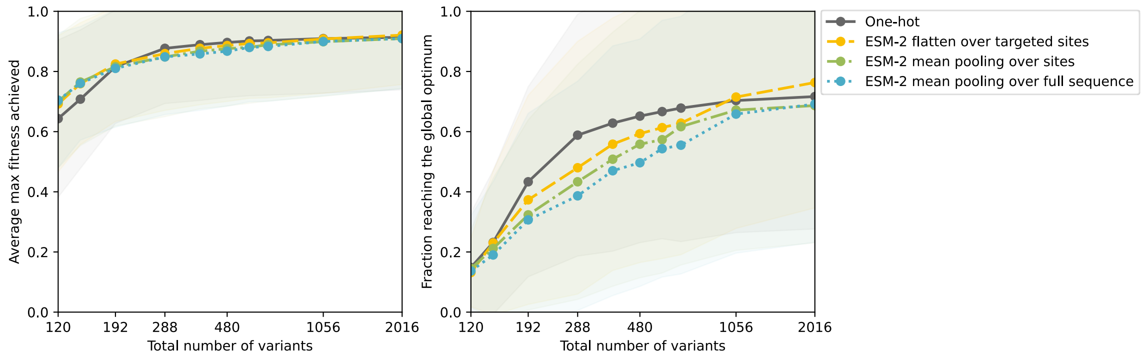


Figure B.3.32. *Encoding strategies for EVmutation-guided ftMLDE performance, averaged across landscapes with at least 1% active variants. Comparison of learned embeddings from the protein language model ESM-2 using different pooling methods vs. one-hot encoding flattened over the substitution sites, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

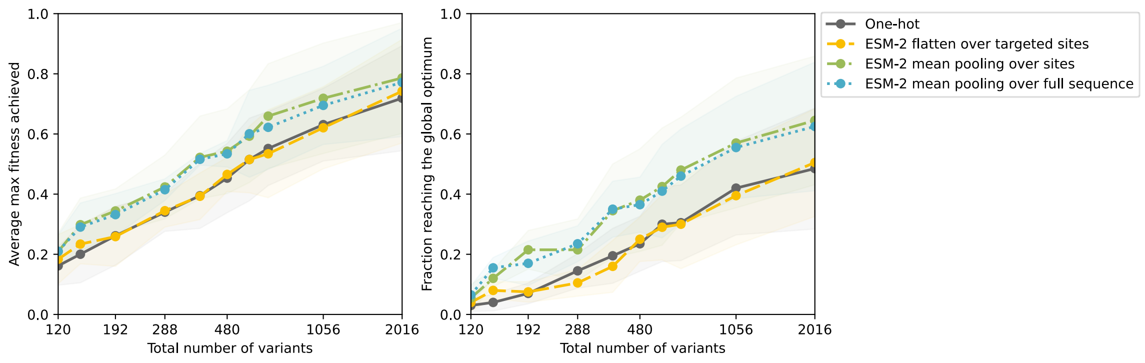


Figure B.3.33. *Encoding strategies for MLDE performance, averaged across four landscapes with fewer than 1% active variants. Comparison of learned embeddings from the protein language model ESM-2 using different pooling methods vs. one-hot encoding flattened over the substitution sites, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

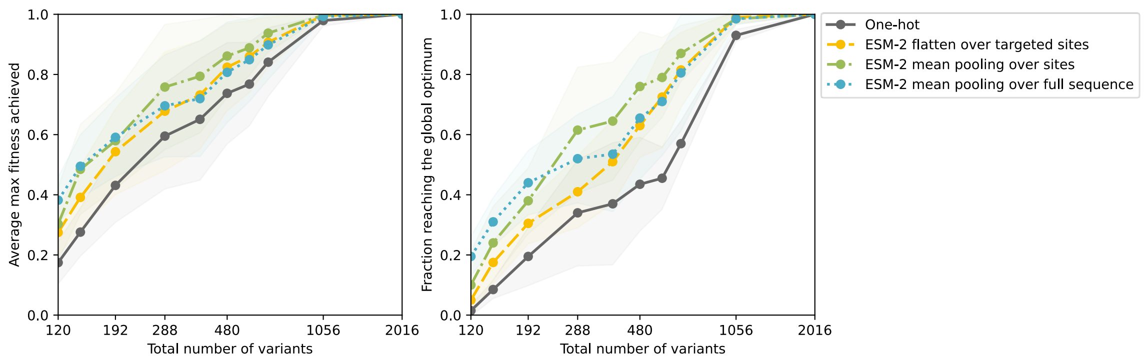


Figure B.3.34. *Encoding strategies for EVmutation-guided ftMLDE performance, averaged across four landscapes with fewer than 1% active variants. Comparison of learned embeddings from the protein language model ESM-2 using different pooling methods vs. one-hot encoding flattened over the substitution sites, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

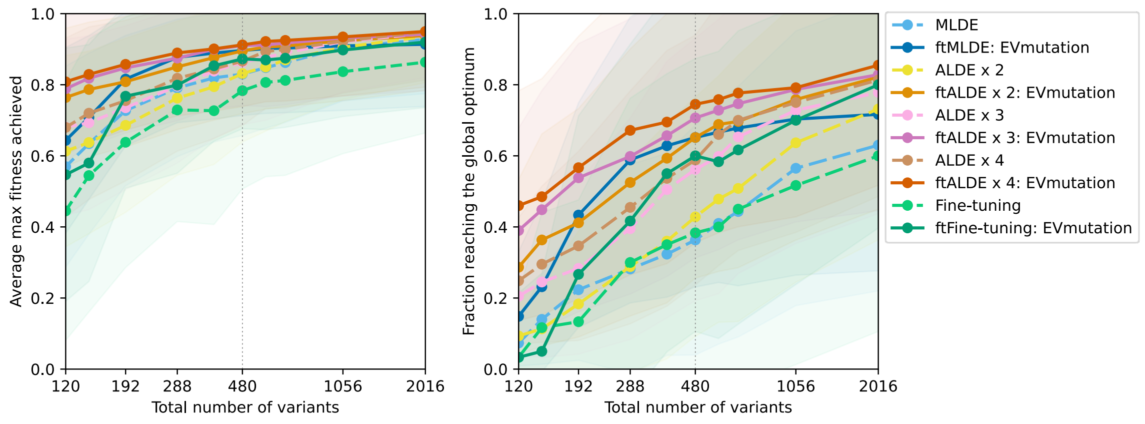


Figure B.3.35. *Performance of fine-tuning and EVmutation-guided fine-tuning (ftFine-tuning) compared with MLDE, ftMLDE, ALDE, and ftALDE, averaged across 12 landscapes with at least 1% active variants. Fine-tuning was performed using ESM-2 with Low Rank Adaptation (LoRA), with five replicates per landscape, related to the **Chapter 3 Discussion section**. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

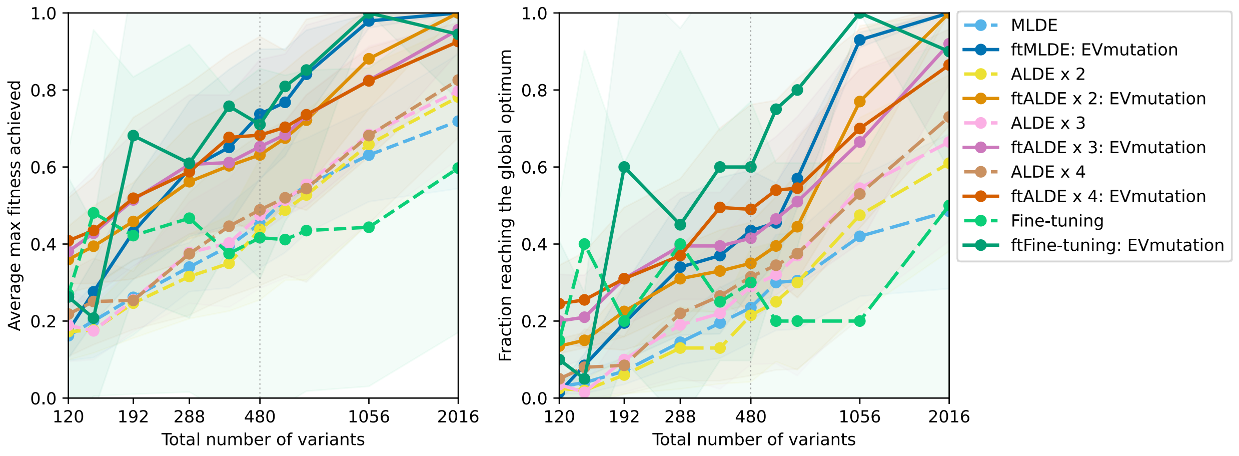


Figure B.3.36. *Performance of fine-tuning and EVmutation-guided fine-tuning (ftFine-tuning) compared with MLDE, ftMLDE, ALDE, and ftALDE, averaged across four landscapes with fewer than 1% active variants. Fine-tuning was performed using ESM-2 with Low Rank Adaptation (LoRA), with five replicates per landscape, related to the **Chapter 3 Discussion section**. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

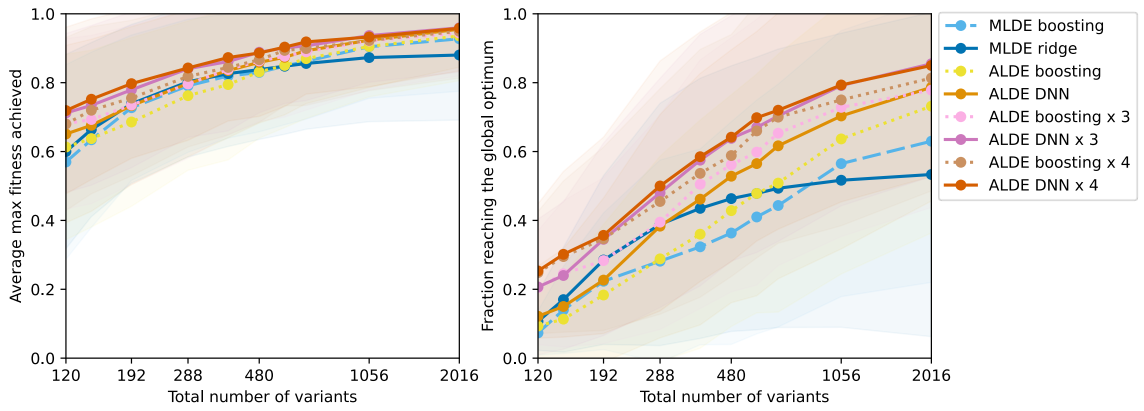


Figure B.3.37. *MLDE and ALDE with different model types, averaged across 12 landscapes with at least 1% active variants. MLDE with boosting or ridge regression. Different rounds of ALDE with boosting or deep neural network ensembles (DNN). No focused training included, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

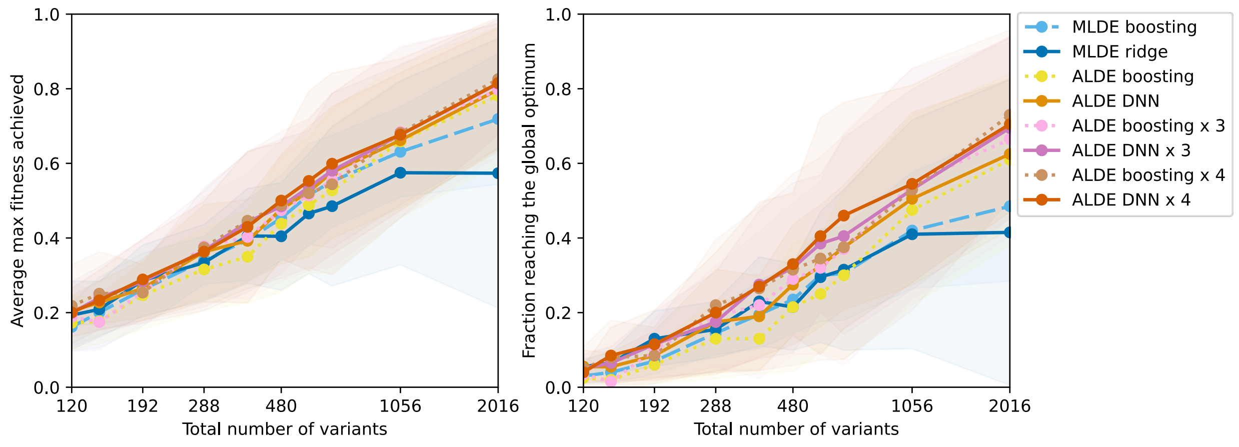


Figure B.3.38. *MLDE and ALDE with different model types, averaged across four landscapes with fewer than 1% active variants. MLDE with boosting or ridge regression. Different rounds of ALDE with boosting or deep neural network ensembles (DNN). No focused training included, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

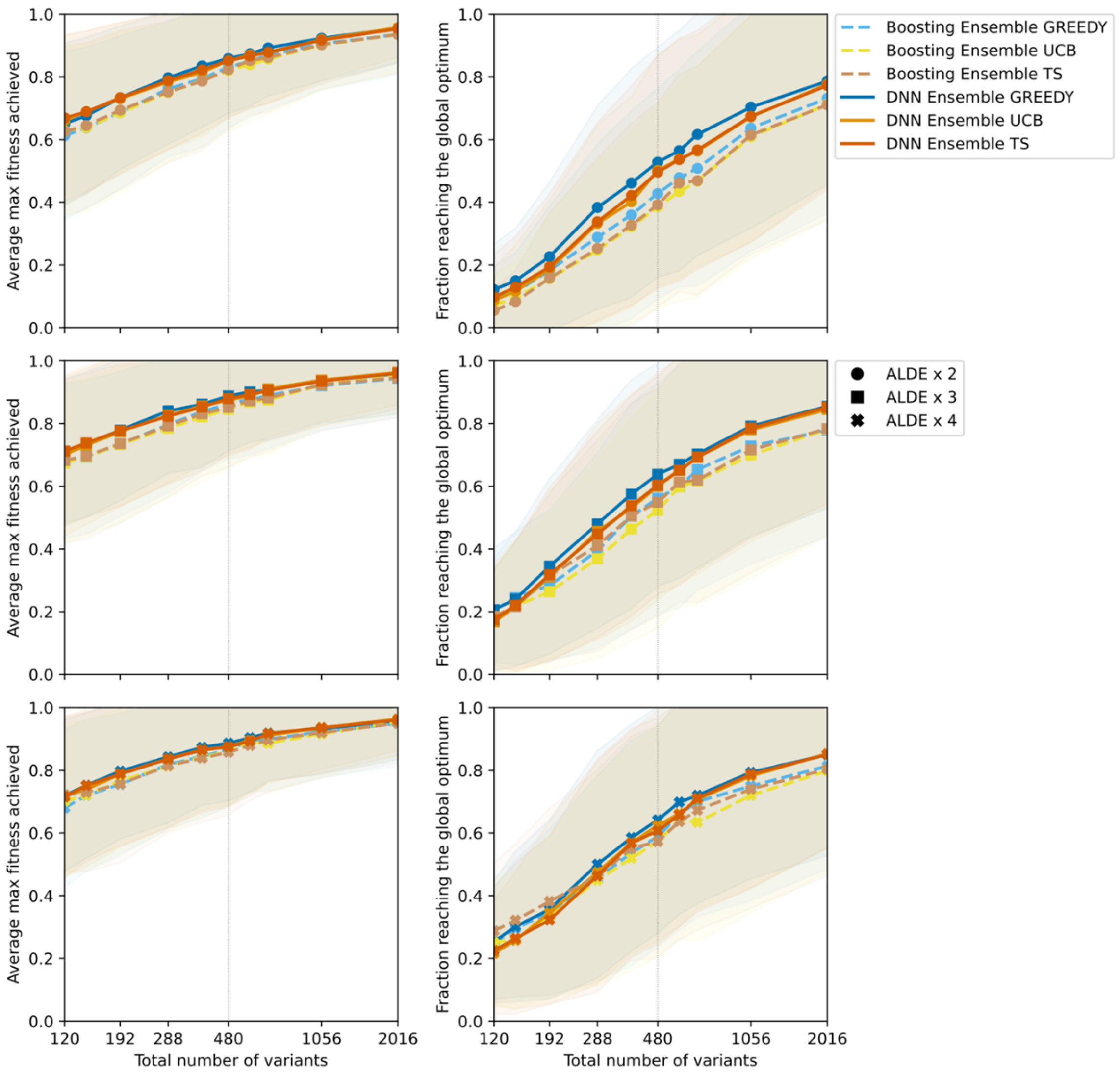


Figure B.3.39. *ALDE with different model type and acquisition function options, averaged across 12 landscapes with at least 1% active variants. Different rounds of ALDE with boosting or deep neural network ensembles (DNN) in combination with greedy, upper confidence bound (UCB), and Thompson sampling (TS). No focused training included, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

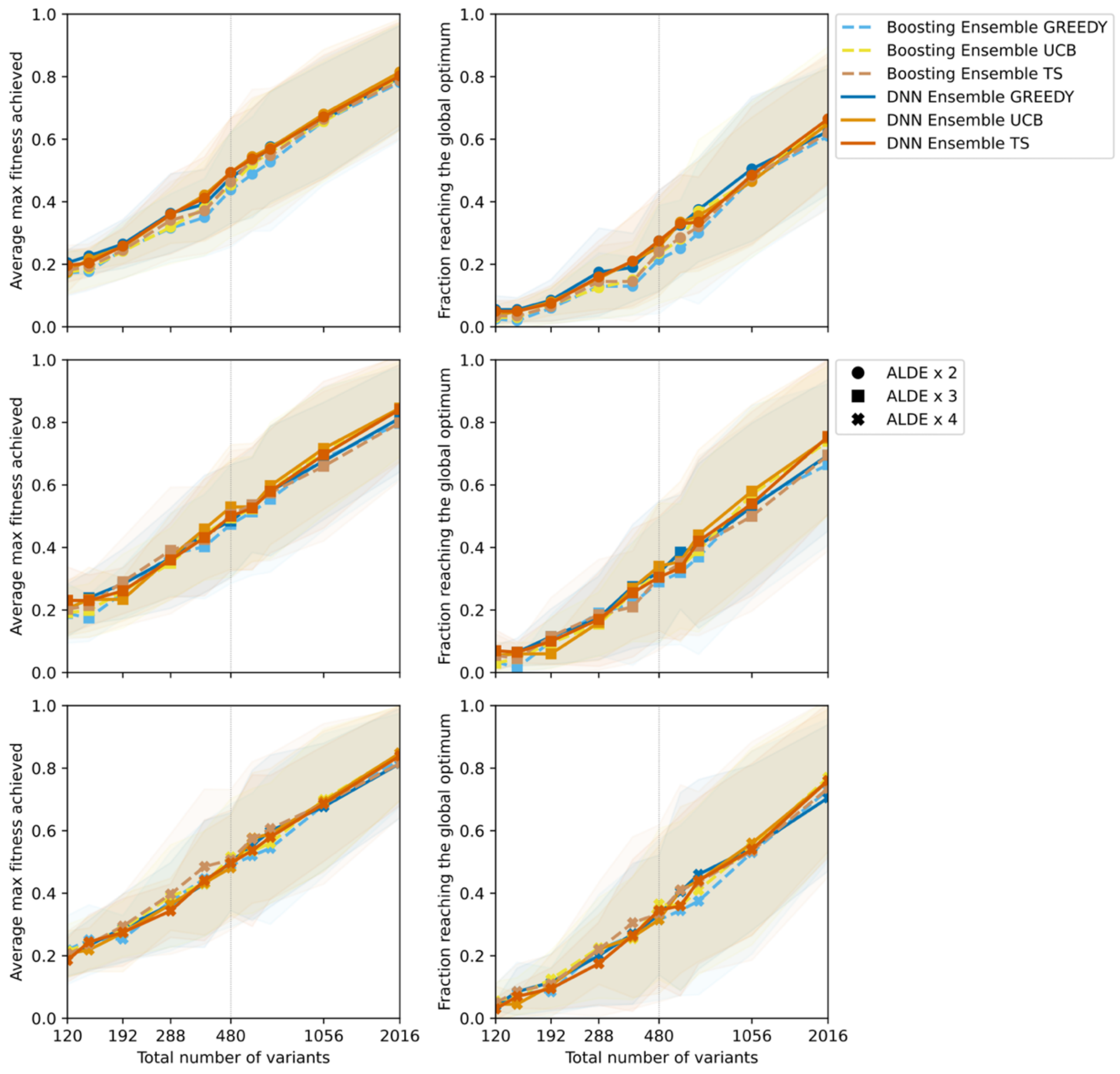


Figure B.3.40. *ALDE with different model type and acquisition function options, averaged across four landscapes fewer than 1% active variants. Different rounds of ALDE with boosting or deep neural network ensembles in combination with greedy, upper confidence bound (UCB), and Thompson sampling (TS). No focused training included, related to the* ***Chapter 3 Discussion section****. Shading indicates the standard deviation across landscape means, each averaged over 50 replicates.*

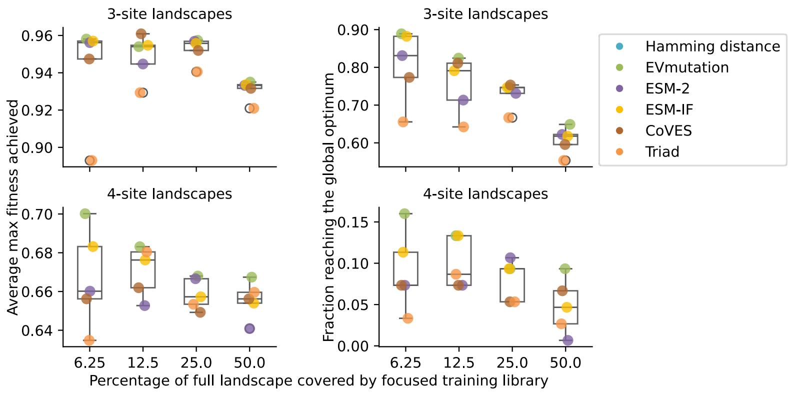


Figure B.3.41. *The impact of reducing the size of the focused training library relative to the full library on ftMLDE performance averaged across 12 landscapes with at least 1% active variants split into three-site landscapes (top row) and four-site landscapes (bottom row). Related to the* ***Chapter 3 Discussion section****.*

B.4 Bibliography for Appendix B

1. Johnston, K. E. *et al.* A combinatorially complete epistatic fitness landscape in an enzyme active site. *Proc. Natl. Acad. Sci.* **121**, e2400439121 (2024).
2. Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife* **5**, e16965 (2016).
3. Tu, B., Sundar, V. & Esvelt, K. M. An ultra-high-throughput method for measuring biomolecular activities. Preprint at https://doi.org/10.1101/2022.03.09.483646 (2024).
4. Sundar, V., Tu, B., Guan, L. & Esvelt, K. FLIGHTED: Inferring fitness landscapes from noisy high-throughput experimental data. Preprint at https://doi.org/10.1101/2024.03.26.586797 (2024).
5. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272 (2020).
6. Papkou, A., Garcia-Pastor, L., Escudero, J. A. & Wagner, A. A rugged yet easily navigable fitness landscape. *Science* **382**, eadh3860 (2023).
7. de Visser, J. A. G. M. & Krug, J. Empirical fitness landscapes and the predictability of evolution. *Nat. Rev. Genet.* **15**, 480–490 (2014).
8. Szendro, I. G., Schenk, M. F., Franke, J., Krug, J. & Visser, J. A. G. M. de. Quantitative analyses of empirical fitness landscapes. *J. Stat. Mech. Theory Exp.* **2013**, P01005 (2013).
9. Kauffman, S. & Levin, S. Towards a general theory of adaptive walks on rugged landscapes. *J. Theor. Biol.* **128**, 11–45 (1987).
10. Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).
11. Hopf, T. A. *et al.* The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35**, 1582–1584 (2019).
12. Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. in *Advances in Neural Information Processing Systems* vol. 34 29287–29303 (Curran Associates, Inc., 2021).

13. Ding, K. *et al.* Machine learning-guided co-optimization of fitness and diversity facilitates combinatorial library design in enzyme engineering. *Nat. Commun.* **15**, 6392 (2024).

14. Hsu, C. *et al.* Learning inverse folding from millions of predicted structures. Preprint at https://doi.org/10.1101/2022.04.10.487779 (2022).

15. Berman, H. M. *et al.* The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242 (2000).

16. Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L. & Dror, R. Learning from protein structure with geometric vector perceptrons. Preprint at https://doi.org/10.48550/arXiv.2009.01411 (2021).

17. Ding, D. *et al.* Protein design using structure-based residue preferences. *Nat. Commun.* **15**, 1639 (2024).

18. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* (2021) doi:10.1016/j.cels.2021.07.008.

19. Kille, S. *et al.* Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. *ACS Synth. Biol.* **2**, 83–92 (2013).

20. Pines, G., Pines, A. & Eckert, C. A. Highly efficient libraries design for saturation mutagenesis. *Synth. Biol.* **7**, ysac006 (2022).

21. Wu, Z., Kan, S. B. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci.* **116**, 8852–8858 (2019).

22. Chen, T. & Guestrin, C. XGBoost: A scalable tree boosting system. in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 785–794 (Association for Computing Machinery, New York, NY, USA, 2016). doi:10.1145/2939672.2939785.

23. Buitinck, L. *et al.* API design for machine learning software: experiences from the scikit-learn project. Preprint at https://doi.org/10.48550/arXiv.1309.0238 (2013).

24. Yang, J. *et al.* Active learning-assisted directed evolution. *Nat. Commun.* **16**, 714 (2025).

25. Balandat, M. *et al.* BoTorch: A framework for efficient Monte-Carlo bayesian optimization. Preprint at https://doi.org/10.48550/arXiv.1910.06403 (2020).

26. Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q. & Wilson, A. G. GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. Preprint at https://doi.org/10.48550/arXiv.1809.11165 (2021).

27. Paszke, A. *et al.* PyTorch: An imperative style, high-performance deep learning library. Preprint at https://doi.org/10.48550/arXiv.1912.01703 (2019).

28. Schmirler, R., Heinzinger, M. & Rost, B. Fine-tuning protein language models boosts predictions across diverse tasks. *Nat. Commun.* **15**, 7407 (2024).

29. Duarte, J. M., Sathyapriya, R., Stehr, H., Filippis, I. & Lappe, M. Optimal contact definition for reconstruction of Contact Maps. *BMC Bioinformatics* **11**, 283 (2010).

30. Chiang, W.-L. *et al.* Chatbot Arena: An open platform for evaluating LLMs by human preference. Preprint at https://doi.org/10.48550/arXiv.2403.04132 (2024).

31. Boubdir, M., Kim, E., Ermis, B., Hooker, S. & Fadaee, M. Elo uncovered: Robustness and best practices in language model evaluation. *Adv. Neural Inf. Process. Syst.* **37**, 106135–106161 (2024).

32. Lite, T.-L. V. *et al.* Uncovering the basis of protein-protein interaction specificity with a combinatorially complete library. *eLife* **9**, e60924 (2020).

Appendix C

SUPPLEMENTARY INFORMATION FOR CHAPTER 4

C.1 Datasets

C.1.1 Summary

All datasets are available on Zenodo: https://zenodo.org/records/15226690.

Table C1. *Dataset summary.*

Enzyme	Substrates	Cofactor	# Pairs	# Sites	Sites	Activity	Selectivity
*Pf*TrpB	4bromo indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	4cyano indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	5bromo indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	5chloro indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	5cyano indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	5iodo indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	6chloro indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	7bromo indole + L-serine	PLP	68	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	7iodo indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	7methyl indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
*Pf*TrpB	5,6chloro indole + L-serine	PLP	241	3	I165, I183, Y301	Absorbance	N.A.
Rma cyt *c*	NHC-borane + Me-EDA	heme	150	6	V75, M99, M100, T101T, D102, M103	% yield	Enantio-
Rma cyt *c*	phenyldimethyl-silane + Me-EDA	heme	150	6	V75, M99, M100, T101T, D102, M103	% yield	Enantio-
ParLQ	a: 4-vinylanisole + EDA	heme	490	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	b: styrene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	c: 1-methyl-4-vinylbenzene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	d: 1-methyl-3-vinylbenzene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	e: 1-methyl-2-vinylbenzene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	f: 1-chloro-4-vinylbenzene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	g: 1-bromo-4-vinylbenzene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	h: 1-(trifluoromethyl)-4-vinylbenzene + EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-
ParLQ	i: 2-vinylnaphthylene +EDA	heme	91	5	W56, Y57, L59, Q60, F89	% yield	Diastereo-

C.1.2 Dataset backgrounds

The tryptophan synthase β-subunit (TrpB) catalyzes a native reaction between L-serine and indole to form tryptophan. Engineered TrpBs extend this function to non-native substrates such as serine analogues and substituted indoles, enabling the synthesis of tryptophan analogs and other noncanonical amino acids that are important precursors to pharmaceuticals and natural products (**Figure 4.1c**).[1–4]

Heme-containing enzymes have been engineered to carry out a plethora of valuable reactions that have not been found in biological systems.[5] These new-to-nature reactivities include carbene transfers for stereoselective olefin cyclopropanation, traditionally requiring unsustainable transition metals,[6,7] and the formation of carbon–silicon (C–Si)[8] and carbon–boron (C–B)[9] bonds (**Figure 4.1d**).

C.1.2.1 Multi-substrate Pf*TrpB dataset*

Library Generation Beginning with a TrpB variant discovered in a directed evolution campaign for 4-nitroTrp formation, *Pf*5G8,[2] a triple-site saturation mutagenesis library was generated. Primers from Table A2 were used to amplify out the vector in three pieces and install variation at positions 165, 183, and 301 via the 22-codon trick.[10] Amplification was performed with Phusion® High-Fidelity DNA Polymerase according to manufacturer recommendations (New England Biolabs, Catalog M0530L). A Gibson assembly was used to generate full-length vectors which were transformed via electroporation into electrocompetent BL21-DE3 *E. coli* cells. These cells were plated onto LB agar containing 100 µg/mL ampicillin.

Single colonies were picked into 96-well deep well plates containing 300 µL TB containing 100 µg/mL ampicillin (TB_{Amp}) and grown overnight at 37 °C, 250 rpm, and 80% humidity. The following day, expression 96-well deep well plates were filled with 630 µL TB_{Amp} and 20 µL culture and grown for 3 h at 37 °C, 250 rpm, and 80% humidity. These plates were then cooled on ice for 20 min prior to adding 50 µL 14 mM IPTG (1 mM final) and incubating overnight at 25 °C and 250 rpm overnight. Cells were pelleted at 3500–4000 rpm for 10 min, the supernatant was decanted, and the plates were then frozen at -20 °C overnight.

Screening To prepare cell lysate, pellets were first resuspended in lysis buffer composed of 1mg/mL HEWL, 2 mM $MgCl_2$, 10X bug buster, 200 µM PLP, and a small amount of DNAse in 50 mM potassium phosphate buffer, pH 8.0 (KPi). They were then incubated at 37 °C for 30 min and heat treated at 75 °C for at least 30 min. Plates were then spun down at 5000 rpm for 10 min and the supernatant was used as cell lysate.

Table C2. *Primer sequences, where XXX = 22 codon trick.*[10]

Name	Sequence
I165_f	GTTCTCGCACCCTGAAAGACGCAXXXGACGAGGCTCTGCGTGATTGG
I165_r	TGCGTCTTTCAGGGTGCGAGAAC
I183_f	GTGGCTACTTTTGAATACACCCACTACCTAXXXGGTTCCGTGGTCGGTCCAC
I183_r	TAGGTAGTGGGTGTATTCAAAAGTAGCCAC
Y301_f	CTCCATCGCACCAGGTCTGGATXXXCCAGGTGTTGGTCCAGAACACG
Y301_r	ATCCAGACCTGGTGCGATGGAG

Table C3. *Wavelengths.*

Compound	Wavelength (nm)
4bromo	304
5bromo	306
7bromo	300
5chloro	306
6chloro	304
5,6chloro	310
5iodo	306
7iodo	306
4cyano	294
5cyano	310
7methyl	296

Nucleophile stocks were made for all indole analogs at 200 mM in either EtOH (4bromo, 5bromo) or DMSO (7bromo, 5chloro, 6chloro, 5,6chloro, 5iodo, 7iodo, 7methyl, 4cyano, 5cyano). Reactions were set up in 96-well deep well plates. Reactions were prepared with 10 µL nucleophile stock (10 mM final), 20 µL lysate, 10 µL L-serine (25 mM final), and 160 µL KPi and incubated in a tightly sealed plate at 75 °C overnight. The next day the reactions were acidified with 200 µL 1M HCl and the unreacted indole extracted with 500 µL EtOAc. The plates were sealed tightly and shaken vigorously, then spun down at 1000 rpm for 3 min to separate the layers before drawing the bottom (aqueous) layer of the mixture into a 96-

well UV-transparent flat-bottom plate. Absorbance was collected every 2 nm from 290–310 nm for every substrate using a Tecan InfiniTe. The absorbance wavelength used for quantification for each substrate was selected according to its absorbance properties (**Table C3**).

C.1.2.2 Multi-substrate ParLQ dataset

The dataset was sourced from the study by Yang and Lal *et al.*[11] The model substrate was presented in their main text, while the substrate scope data was provided in the supplementary information. Additional experimental details were confirmed through direct communication with the authors.

Activity was calculated based on GC-FID measurements, where the product area was normalized to the internal standard area and converted using the calibration curve from Figures S11–S28. The yield calculation followed the author's notebook (GitHub repository: https://github.com/jsunn-y/ALDE/blob/master/analysis/visualization.ipynb), normalizing the area to the maximum possible product concentration and accounting for the 1.5X dilution from the reaction. The *cis* isomer was the major product. Selectivity was determined by calculating the ratio of the *cis* to *trans* isomer.

C.1.2.3 Rma *cytochrome* c *C–B and C–Si dataset*

The dataset was sourced from the study by Ding *et al.* (Supplementary Tables 3 and 4).[12] Upon communication with the corresponding author, we confirmed that no sequence information was collected from the random mutagenesis libraries presented in Supplementary Tables 5 and 6.

C.2 Mechanism

C.2.1 PLP-dependent TrpB reactions

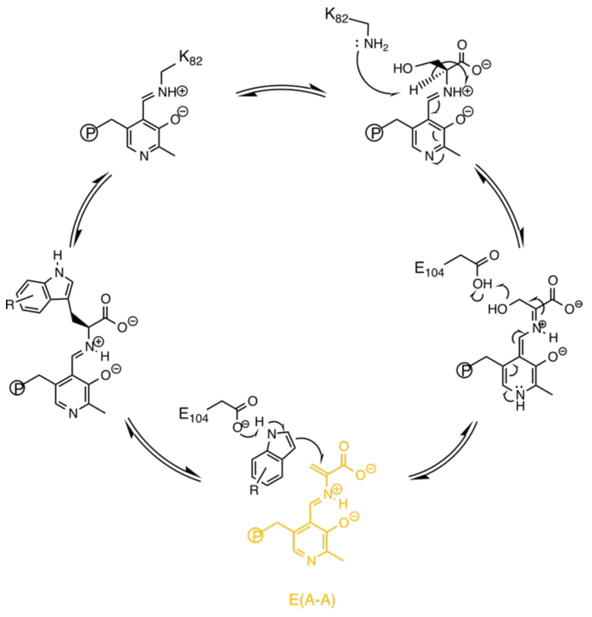


Figure C1. *TrpB mechanism based on published studies.*[1,2,13] *The E(A-A) intermediate together with the substrates were used for ZS predictors.*

C.2.2 Heme-based carbene transfer reactions

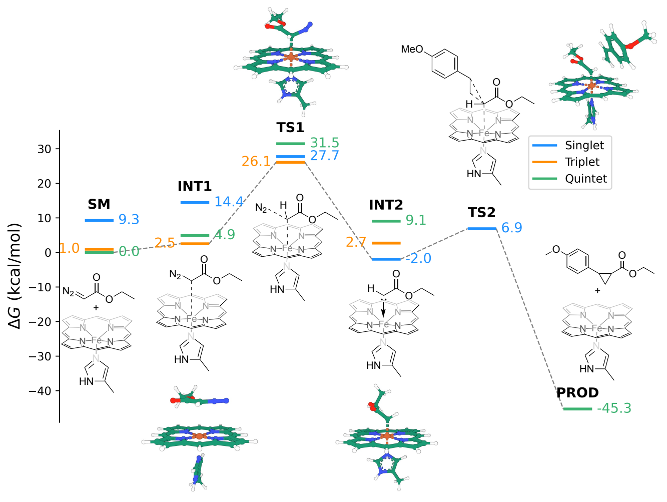


Figure C2. *Mechanism for heme-based carbene transfer reaction as computed by DFT (**Appendix C.3.15**).*[14,15] *TS2 was used for ZS predictors.*

C.3 Methods

C.3.1 General ZS predictors

Hamming distance, EVmutation, ESM, ESM-IF, CoVES, and ($\Delta\Delta G_f$) ZS scores were calculated based on the study by Li *et al.*[16]

C.3.2 Vina

AutoDock Vina v1.2.5 was used. PDBQT files for substrates were prepared from corresponding SMILES strings using RDKit at pH 7.4 and Open Babel.[17–19] The cofactor was extracted from the parent PDB and converted to PDBQT using Open Babel, while metal ions were prepared separately. Receptor structures were derived from parent PDB structures (PDB ID: 5DW0 for *Pf*TrpB and 3CP5 for *Rma* cyt *c*), while the structure for ParLQ was modeled by the authors using Alphafold 3 with a bound-heme. Variant structures were generated using MDAnalysis.

Docking coordinates were defined by the centroid of the substrate-cofactor complex with a box size of 20 Å. Each docking experiment was performed in five replicates, with nine docking modes and an exhaustiveness setting of 32. The lowest energy from each replicate was recorded, and the final energy was averaged across replicates. The negative values of the energies were used as the ZS predictor.

C.3.3 Rosetta GALigandDock

The Pyrosetta GALigandDock-based ZS scores were obtained from a local copy of the pyrosetta distribution *pyrosetta-2025.3+release.1f5080a079-py3.12-linux-x86_64.egg/pyrosetta/distributed*. Two conda environments (*anaconda.org*) were created. One for the input preprocessing and one for inference of Pyrosetta GALigandDock. To set them up, download the corresponding .yaml files (*ambertools.yml* and *pyrosetta env.yml*) and execute the following console commands:

```
$ conda env create -f <path/to/ambertools.yml>
```

and

```
$ conda env create -f <path/to/pyrosetta_env.yml>
```

respectively.

To preprocess the inputs, the first conda environment was activated and the script *pyrosetta pipeline.py* was executed with the following parameters:

```
python -m substrate_aware.zs.pyrosetta_pipeline --meta_list
<path/to/campaign_1_meta.csv>
                    <path/to/campaign_2_meta.csv>
                    <...>
                    <path/to/campaign_n_meta.csv>
--struc_dir <path/to/structures_dir>
--tmp_dir <path/to/dir/for/tempfiles>
--out_dir <path/to/dir/for/output_files>
--rosettascript_path <rosetta/source/.../mol2genparams.py>
--net_charge_unit_1 <net charge>
--net_charge_unit_2 <net charge>
```

The script takes docked structures for a given campaign as input and returns them adequately reformatted for Pyrosetta, alongside a Pyrosetta-specific parameter file for each of the campaign's substrates. The script runs into a tracepoint and prompts the user to manually correct a newly created mol2 file of the substrates and then save it under a printed location. For this purpose, the file was then downloaded, observed in a 3D molecular viewer, such as Avogadro[20] and edited to meet antechambers[21] requirements for am1bcc charge generation. This includes adding hydrogen, correcting unnatural bond orders, and ensuring that the molecule only contains atoms of the element set {H, C, N, O, F, P, S, Cl, Br, I} on which antechamber is parametrized. In the case of iron coordination centers, the metal atom was replaced by a phosphorous. Boron and silicon were substituted with carbon. Lastly, each connected unit must consist of 4 or more atoms. Units with less than that (e.g., ions) were omitted. To finish the preprocessing, the console prompts were followed.

The second script, *pyrosetta inference.py*, runs the actual GALigandDock docking by executing it with the following parameters:

```
python      -m      substrate_aware.zs.pyrosetta_inference      --meta_list
<path/to/campaign_1_meta.csv>
                    <path/to/campaign_2_meta.csv>
                    <...>
                    <path/to/campaign_n_meta.csv>
--preprocessed_dir <path/to/directory_containing_pdbs_and_params>
--results_dir <path/to/results_dir>
```

The docking mover is parametrized within this script. This will create two output files for each variant of all campaigns. The *variantname aligned enzyme final.pkl* contains the best scoring docked poses and *variantname aligned enzyme final.csv* contains a table with Rosetta metrics of these poses. Finally, for each campaign *campaignname.csv* summarizes the Rosetta-metrics of the best docked pose of each variant together with variant ground-truth data.

The negative values of all energy terms were extracted. The dH value, representing enthalpy, was used as the score to indicate the thermodynamic stability of the binding event, where more exothermic values correspond to stronger binding.

C.3.4 Alphafold 3 (AF3)

For *Pf*TrpB, the substrate SMILES was joint with the E(A-A) intermediate (**Figure C1**) and the crystallographic sodium ion to prevent the substrate from docking onto the enzyme surface. For heme-based reactions, the substrate SMILES was assigned to chain B, while the carbene-heme intermediate complex (**Figure C2**) TS2 was assigned to chain C. All scores were extracted from five replicates, and the final structure for each variant was aggregated. The scores from the replicates were averaged. For chain-predicted aligned errors (PAE), the negative values were used as the predictor. The confidence scores of each residue at the targeted site were also extracted and averaged as a predictor.

C.3.5 Chai-1

Chai-1 version 0.1.0 was used, following the same process as AF3, except without MSAs and using PAE as scores.

C.3.6 LigandMPNN

Code from LigandMPNN GitHub (https://github.com/dauparas/LigandMPNN) was adopted to extract the ZS scores.[22] The model with 20 Å Gaussian noise was chosen. Only the mutated residues of the campaign were redesigned with autoregressive scoring. To mitigate biases introduced through decoding order, the number of batches was set to 100. Variant likelihoods were thus obtained through:

$$P_{\text{variant}} = \frac{1}{100}\sum_{i=1}^{100}\prod_{n=1}^{N_{\text{mut}}} P\left(AA_n \middle| \text{Backbone}, \{AA_j | j < n\}_i\right)$$

C.3.7 FlowSite

Code from FlowSite GitHub (https://github.com/HannesStark/FlowSite) was adopted to extract ZS scores.[23] The parameters were chosen according to the author’s suggestion. To evaluate the docking and sequence co-generation as appropriate to directed evolution campaigns, both the residues to design and the pocket were defined via the mutated sites. For each variant, 100 inference trajectories were generated. Predicted likelihoods were averaged among inferences and position to yield the final variant ZS score.

C.3.8 Bond distance

Bond distances were derived from AF3 docked structures based on the mechanisms for bond-forming atoms (**Appendix C.2**). For *Pf*TrpB, distances were measured between the catalytic Glu104 and N1-hydrogen. For heme-based carbene transfer reactions, the distances were measured between the carbene carbon and either boron, silicon, or the styrene double bond.

Distances were calculated for each replicate and averaged. The negative value of the bond distance was used as the predictor, based on the hypothesis that closer reactive atoms lead to stronger reactivity and, consequently, higher activity.

C.3.9 Protein-ligand-interaction-profiler (PLIP)

A local copy of the PLIP software (release 2.4.0) was obtained from the GitHub (https://github.com/pharmai/plip).[24] The AF3 docked strucutres were used as inputs. An output XML report file was generated to characterize each variant's ligand-active-site-interactions.

C.3.10 Active-site identification

Two different active site extraction heuristics were explored. The first heuristic defines all residues to belong to the active site, that bear the centroid of their side-chain atoms within a 10 Å distance threshold of the ligand's centroid.

The second extraction heuristic used PLIP to define the active site. The residues tagged with "bindingsite" were considered (**Appendix C.3.9**).

C.3.11 Hydrogen bonds

The AF3 docked structures were used to run PLIP (**Appendix C.3.9**). The number of hydrogen bonds identified in the active site was extracted from the output files and used as a ZS predictor.

C.3.12 Hydrophobicity

For the enzyme, active-site hydrophobicity was calculated based on different active-site identification methods (**Appendix C.3.10**) using various scales, including the Kyte-Doolittle scale,[25] the Hopp-Woods scale,[26] the Eisenberg scale,[27] and theoretical and empirical residue solvent accessibility.[28]

For the ligand, logP was calculated. While previous literature used the Kyte-Doolittle scale to identify hydrophobic regions likely to be in transmembrane segments,[29] we instead chose the Hopp-Woods scale, which highlights antigenic (hydrophilic) regions on protein surfaces.

C.3.13 Active-site volume

The substrate volume was estimated based on the Convex-Hull of the substrate. The active-site volume of the parent was estimated with CASTp based on PDB ID 5DW0 for *Pf*TrpB, 3CP5 for *Rma* cyt *c* and 3ZJI for ParLQ.[30] The variant active-site volume was estimated by the different in different amino acid side chain at the targeted sites.

C.3.14 Similarity calculations

To quantify the similarity between the indole analogs to the native indole, Tanimoto similarity of atom-pair fingerprints[31] was calculated with RDKit.[17,18]

C.3.15 DFT calculations

DFT calculations were conducted using Orca 6.0.[32] We constructed a model containing the porphyrin core, Fe center and an imidazole to mimic the histidine ligand. Geometry optimizations and frequency calculations were performed using the unrestricted B3LYP hybrid functional with def2-TZVP basis set and with D3(BJ) dispersion correction. All geometries were verified as minima or first-order saddle points by frequency analysis. Enthalpies and entropies were calculated for 1 atm and 298.15 K. The SMD continuum solvation model was used in all optimizations and single point calculations with water as the implicit solvent to approximate the energy otherwise required when the reaction is performed without the enzyme. In **Figure C2**, we show the complete energetics of heme-catalyzed cyclopropanation with different spin-states. In other carbenoid reactions, we report energetic barriers derived from open-shell singlet calculations of the C–Si insertion/borylation transition state, in comparison against the carbene-porphryin intermediate. Readers should note that while DFT can derive reasonable geometries for transition states, the absolute energy values can have significant margin of error and should only serve as qualitative estimates. Surprisingly, DFT calculations yielded similar activation energies of ~ 9 – 13 kcal/mol for the three new-to-nature reactions, as shown by other studies.[33–35] We also obtained the ΔG of the reaction considering all substrates and products (**Table C12**).

C.3.16 Ensemble models

To ensemble ZS predictors into a unified score, unweighted ensemble and different types of learned linear models were explored. Results from the shallow neural network were excluded due to overfitting.

Unweighted ensemble. Each ZS predictor was ranked, and the ranks of different chosen ZS predictors were summed up for the final score.

Learned ensemble. Each model was fitted on one specific enzyme optimization campaign and successively tested on all other campaigns. The models included linear regression, piecewise linear regression with a threshold. By doing so, we tested whether a model's learned relationship between feature scores and measured activities. During prediction, these models thereby weighted individual ZS scores and introduced nonlinearities. Given the data:

$$\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}, \qquad \mathbf{x}^{(i)} = (x_1{}^{(i)}, x_2{}^{(i)}, \cdots, x_n{}^{(i)}), \qquad y^{(i)} \in \mathbb{R} \qquad .$$

The goal is defined as:

$$\min \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

where $\hat{y}^{(i)}$depends on the chosen transformation. And that fitting on 1 set of $y^{(i)}$ generalizes to other sets of ys.

Inputs where normalized according to:

$$x_i = \frac{x_i - \mu_x}{\sigma_x}$$

Linear regression. In the case of linear regression (*w*), the prediction is obtained by the transformation:

$$\hat{y}^{(i)} = w_0 + \sum_{j=1}^{n} w_j x_j^{(i)}$$

where w_0 and w_j are obtained through the optimization problem:

$$\min_{w_0, \cdots, w_n} \sum_{i=1}^{m} \left(y^{(i)} - \left[w_0 + \sum_{j=1}^{n} w_j x_j^{(i)} \right] \right)^2$$

Piecewise linear regression. Although linear regression is straightforward to fit and interpret, it may fail to capture threshold-dependent behaviors (e.g., scores only become useful after a certain threshold and optimization is capped after a certain cutoff). To address this, we additionally considered a piecewise linear regression model, which introduces simple nonlinearities via a learned threshold for each feature. The prediction is obtained by the transformation:

$$\hat{y}^{(i)} = w_0 + \sum_{j=1}^{n} w_j \phi_j \left(x_j^{(i)}; \alpha_{j1}, \alpha_{j2}\right)$$

where the mapping function ϕ_j introduces the nonlinearity:

$$\phi_j(x_j; \alpha_{j1}, \alpha_{j2}) = \begin{cases} 0, & x_j < \alpha_{j1}, \\ \dfrac{x_j - \alpha_{j1}}{\alpha_{j2} - \alpha_{j1}}, & \alpha_{j1} \le x_j < \alpha_{j2}, \\ 1, & x_j \ge \alpha_{j2}. \end{cases}$$

The piecewise model parameters $\{w_0, w_j\}$ and thresholds $\{\alpha_{j1}, \alpha_{j2}\}$ are fit by minimizing the sum of squared errors:

$$\min_{\substack{w_0,\cdots,w_n \\ \alpha_{j1},\alpha_{j2}}} \sum_{i=1}^{m} \left(y^{(i)} - \left[w_0 + \sum_{j=1}^{n} w_j \phi_j \left(x_j^{(i)}; \alpha_{j1}, \alpha_{j2}\right) \right] \right)^2$$

subject to $\alpha_{j1} < \alpha_{j2}$ for each feature j.

C.4 Additional results

C.4.1 Dataset visualization

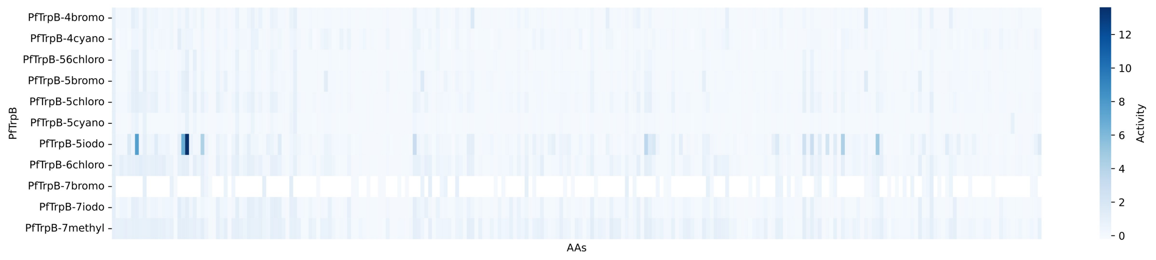


Figure C3. Pf*TrpB activity.*

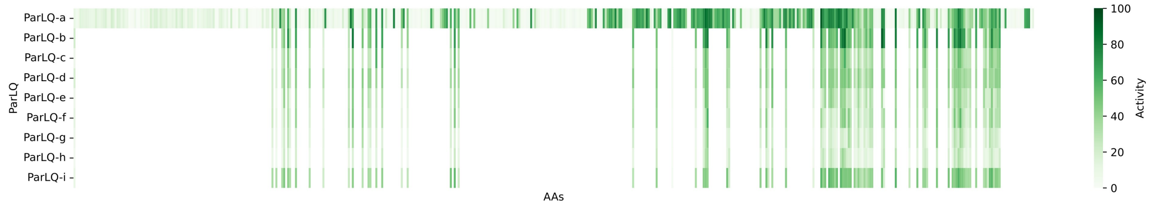


Figure C4. *ParLQ activity.*

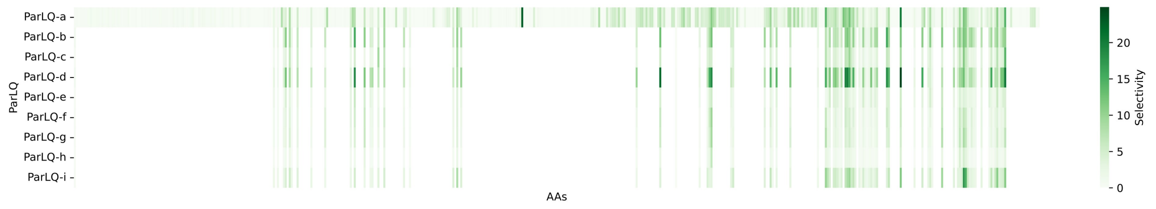


Figure C5. *ParLQ diastereoselectivity.*

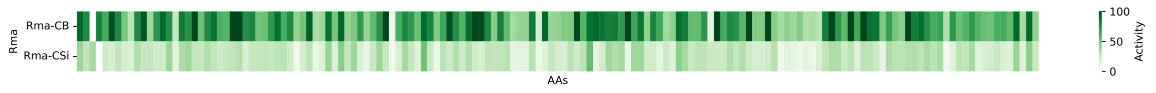


Figure C6. Rma *cyt* c *activity.*

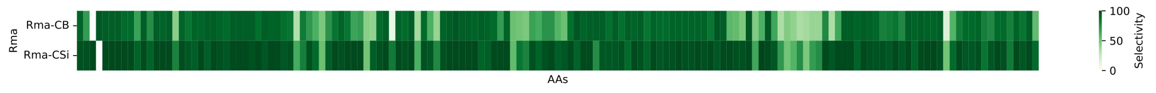


Figure C7. Rma *cyt* c *enantioselectivity.*

C.4.2 Individual ZS predictor performance

Table C4. *ZS predictors are averaged across all, non-native, and new-to-nature datasets.* **Bold** *indicates the best predictor,* ***bold italics*** *indicates the second-best predictor, and* *italics* *highlight the third-best predictor within each category.*

ZS predictor	All	Non-native substrate	New-to-nature chemistry
Hamming distance	0.1056	0.3378	-0.1266
EVmutation	**0.2768**	0.4652	0.0885
ESM-2	0.1904	***0.5125***	-0.1316
ESM-IF	*0.2534*	*0.4810*	0.0257
CoVES	0.1473	0.4075	-0.1129
$\Delta\Delta G_f$	0.2379	**0.5253**	-0.0495
Vina	0.0361	0.0257	0.0465
GALigandDock	0.1393	0.1228	***0.1559***
AF3	***0.2751***	0.2416	**0.3086**
Chai-1	0.1420	0.2094	0.0746
LigandMPNN	0.2105	0.4780	-0.0570
FlowSite	0.2176	0.4007	0.0345
Bond distance	0.0607	0.1853	-0.0639
Hydrogen bonds	0.1633	0.2802	0.0464
Hydrophobicity	0.2028	0.4128	-0.0072
Active-site volume	0.0828	0.0469	*0.1187*

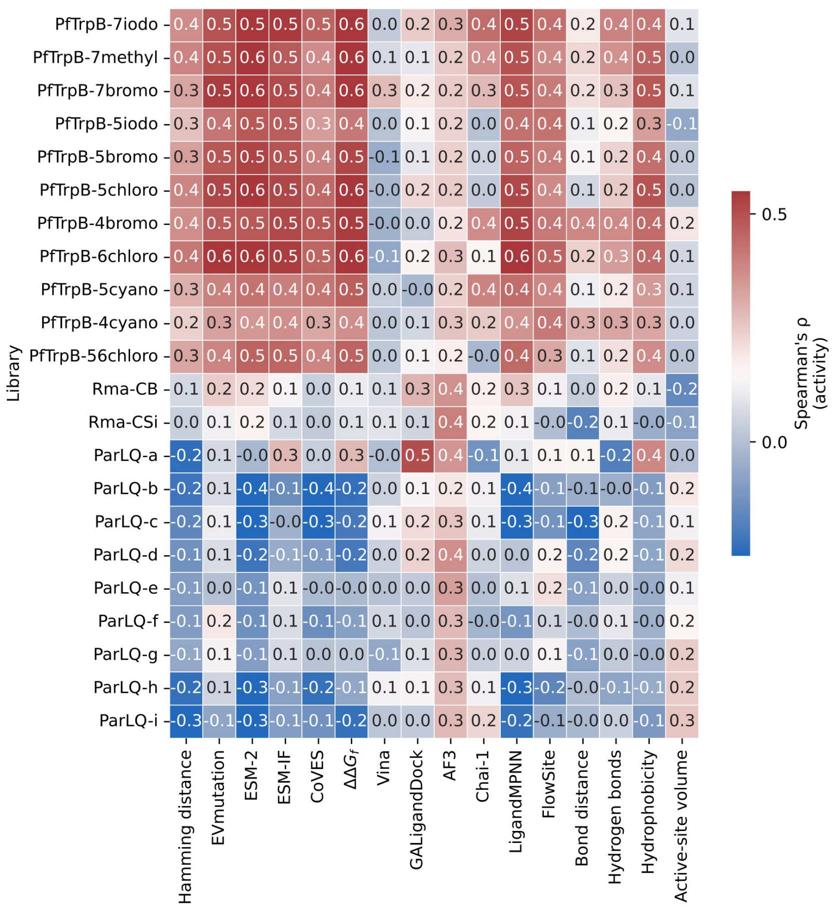


Figure C8. *Spearmen's correlation for activity.*

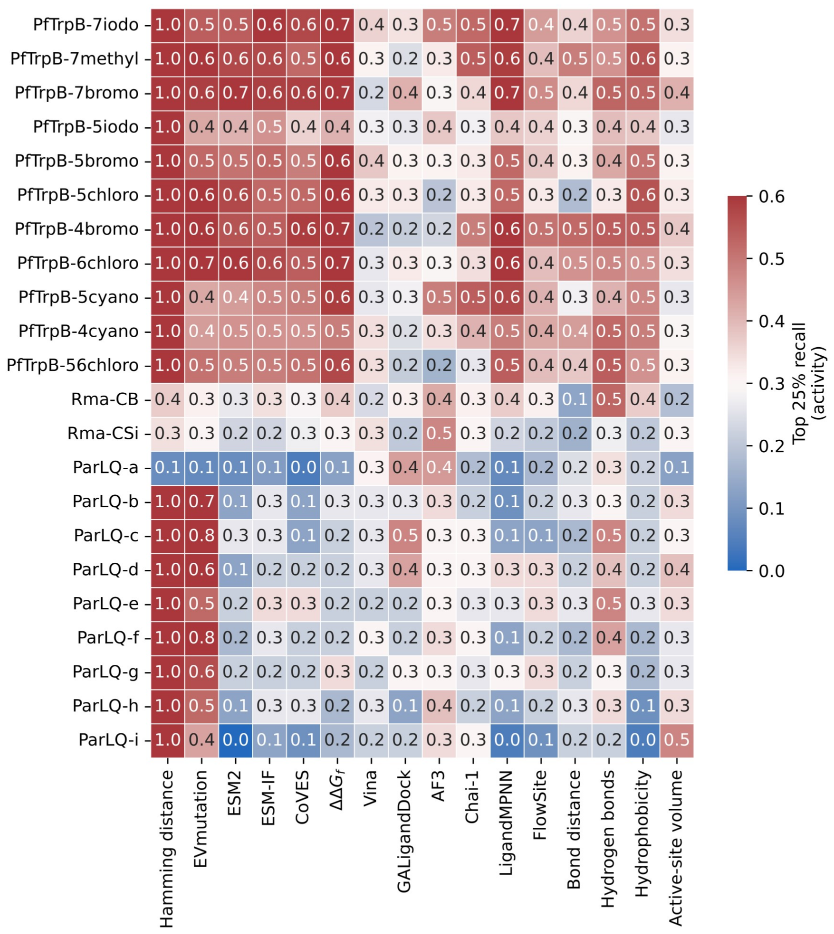


Figure C9. *Top 25% recall for activity.*

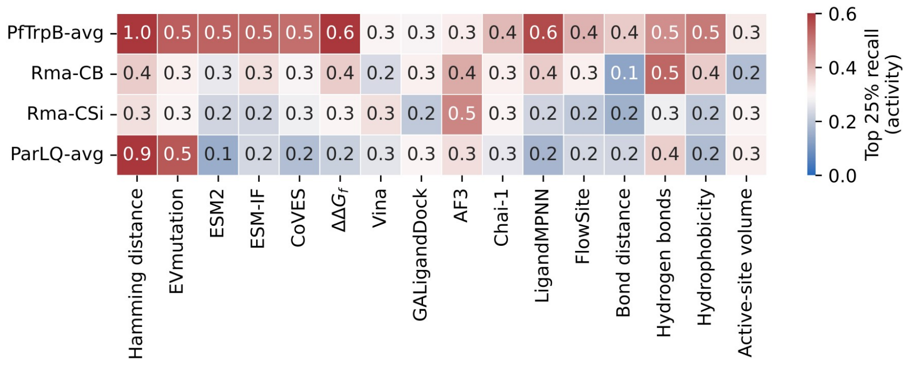


Figure C10. *Top 25% recall for activity, averaged by chemistry.*

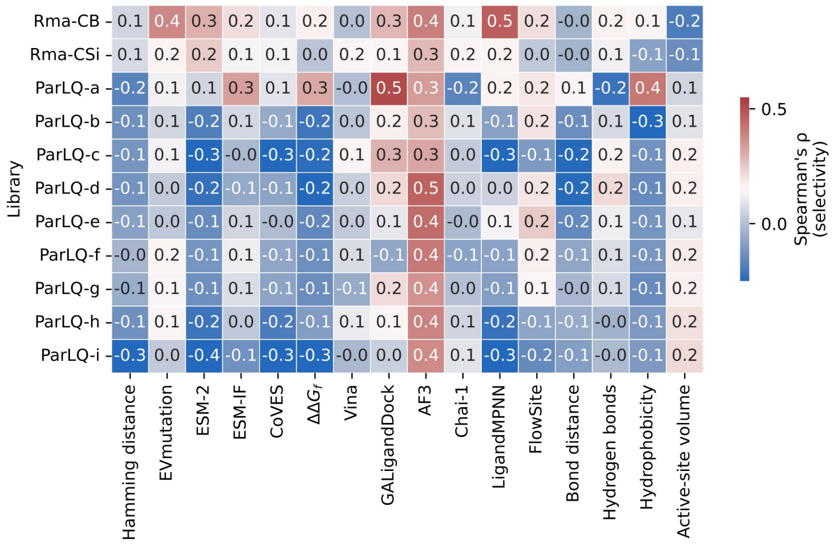


Figure C11. *Spearmen's correlation for selectivity.*

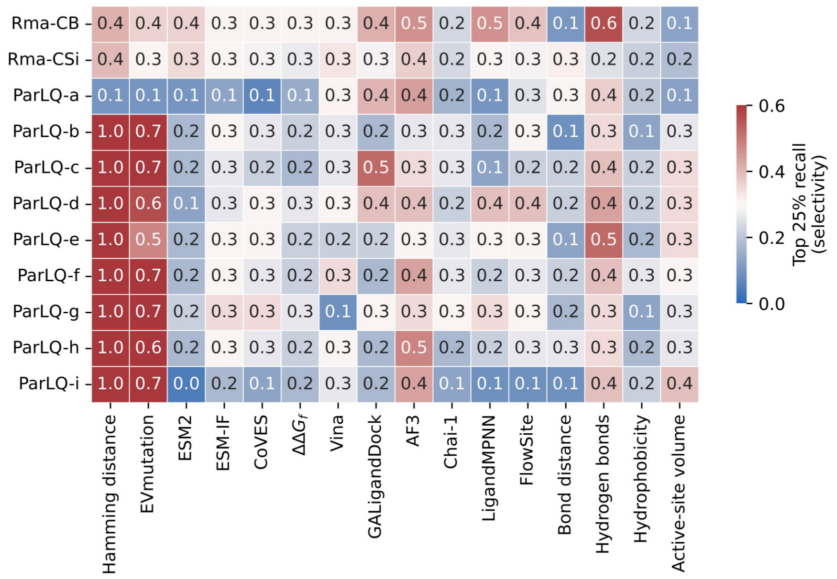
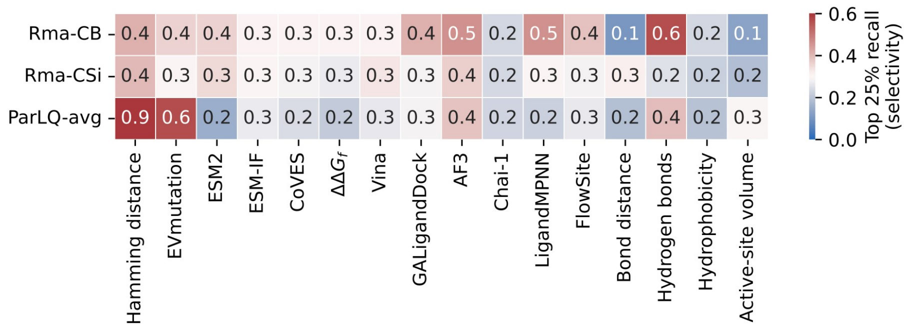


Figure C12. *Top 25% recall for selectivity.*



	Hamming distance	EVmutation	ESM2	ESM-IF	CoVES	$\Delta\Delta G_f$	Vina	GALigandDock	AF3	Chai-1	LigandMPNN	FlowSite	Bond distance	Hydrogen bonds	Hydrophobicity	Active-site volume
Rma-CB	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.4	0.5	0.2	0.5	0.4	0.1	0.6	0.2	0.1
Rma-CSi	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.2	0.3	0.3	0.3	0.2	0.2	0.2
ParLQ-avg	0.9	0.6	0.2	0.3	0.2	0.2	0.3	0.3	0.4	0.2	0.2	0.3	0.2	0.4	0.2	0.3

Figure C13. *Top 25% recall for selectivity, averaged by chemistry.*

C.4.3 Combination of ZS predictors

Table C5. *Spearman's ρ of 34 unweighted ensembles of ZS predictors generalized better than the top individual ZS across all chemistries. See* ***Table C4*** *for topN predictors.*

Predictor Combination	Average Spearman's ρ across all chemistries
All top2	0.3859
New-to-nature top8	0.3802
New-to-nature top9	0.3769
New-to-nature top5	0.3757
All top3	0.3738
New-to-nature top7	0.3708
New-to-nature top11	0.3598
New-to-nature top10	0.3588
New-to-nature top6	0.3579
All top4	0.3531
New-to-nature top12	0.3460
All top5	0.3457
New-to-nature top13	0.3395
New-to-nature top4	0.3390
All top6	0.3316
New-to-nature top14	0.3179
All top7	0.3162
New-to-nature top15	0.3091
All top12	0.3079
All top9	0.3046
All top14	0.3008
Non-native top16	0.3007
All top16	0.3007
New-to-nature top16	0.3007
All top13	0.3000
All top8	0.2983
Non-native top15	0.2978
All top15	0.2978
All top11	0.2943
Non-native top14	0.2938
All top10	0.2901
Non-native top12	0.2887
Non-native top11	0.2843
Non-native top13	0.2802

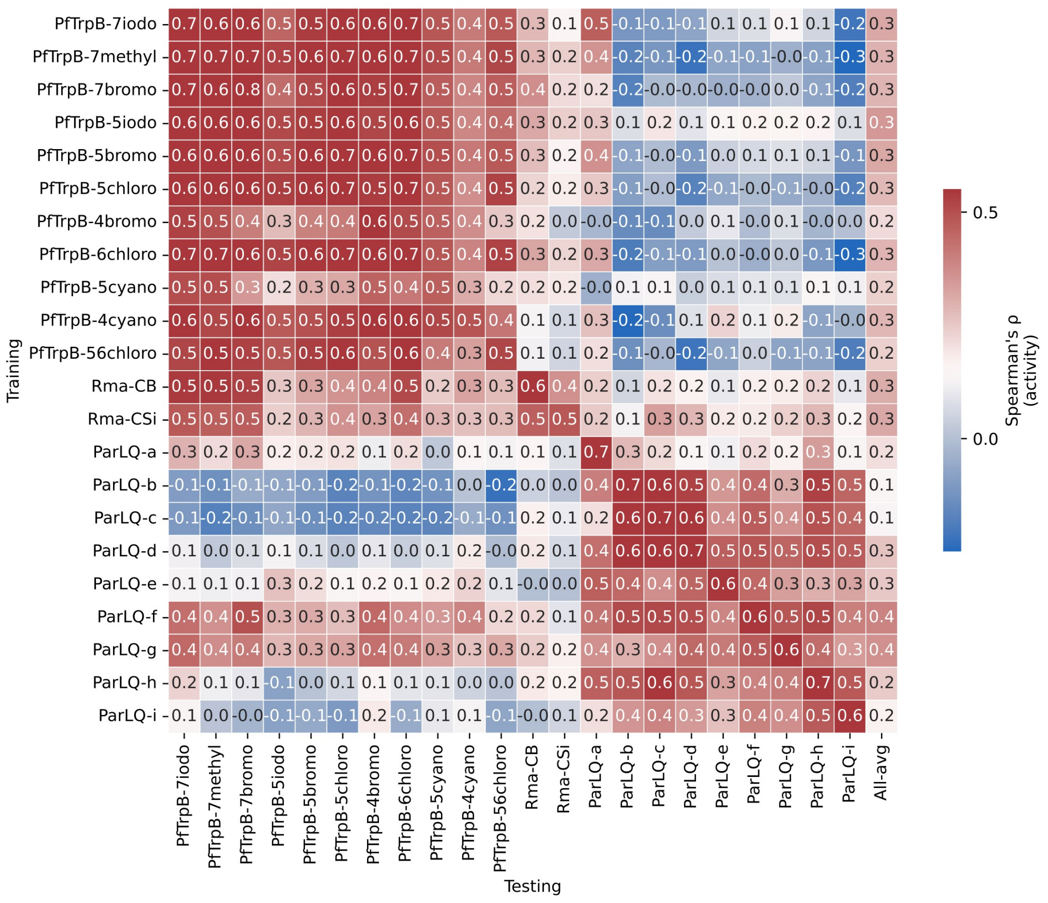


Figure C14. *Linear regression model trained on one library with 16 ZS and tested on all.*

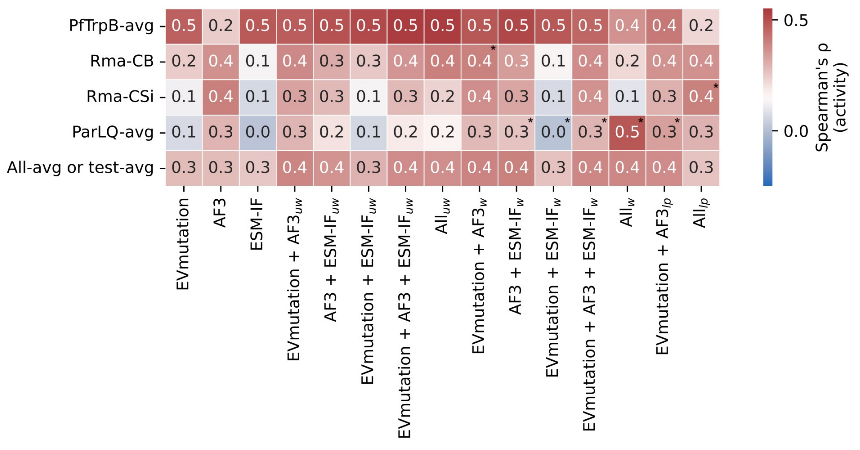


Figure C15. *Different model and ZS predictor combinations for ensembling. uw refers to an unweighted combination. w refers weighted linear combination trained on the best dataset and tested on the rest. lp refers to piecewise linear models trained on the best dataset and tested on the rest. The * symbol indicates the training set, which is excluded from the test-avg calculation.*

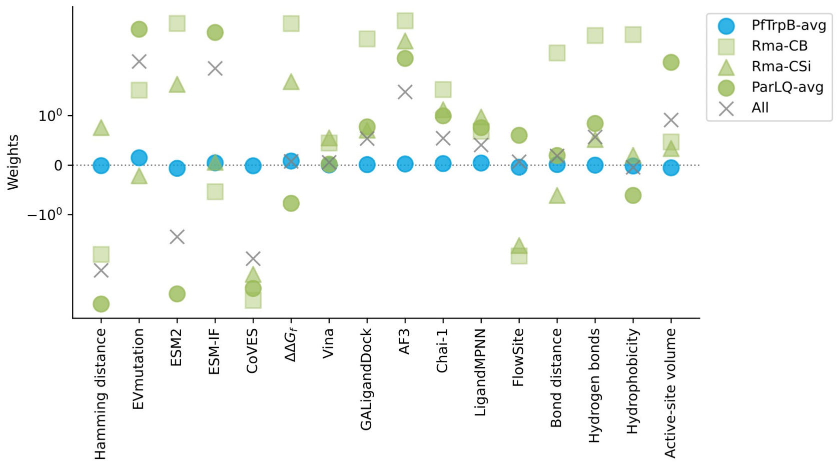


Figure C16. *Averaged weights for linear regression model trained on one library with 16 ZS and tested on all.*

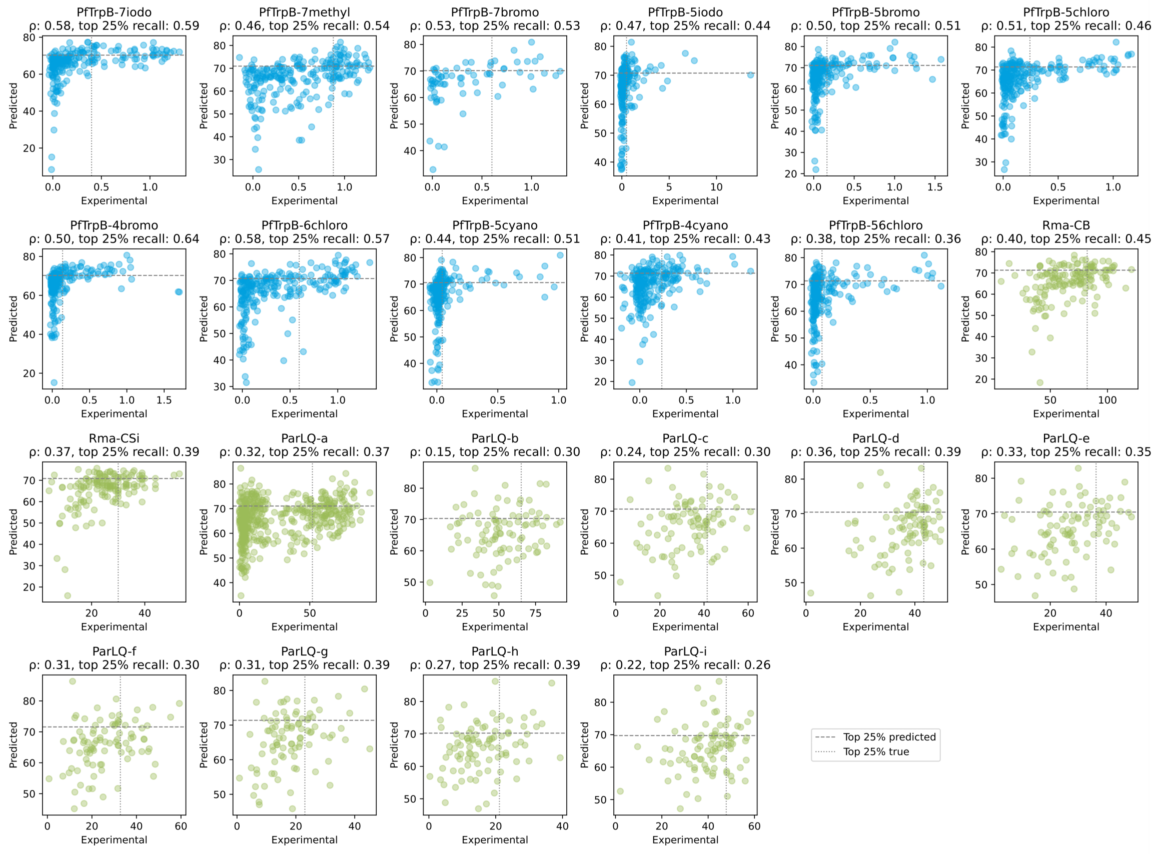


Figure C17. *Scatter plot for EVmutation + AF3$_w$.*

C.4.3 Additional tables

Table C6. *Bitscore and sequence counts for* Pf*TrpB,* Rma *cyt* c*, and ParLQ. The* ***bold*** *row indicates the chosen MSA covering all the targeted sites.*

Enzyme	Bitscore	Sequences
*Pf*TrpB	0.1	74795
	0.3	5996
	0.5	5935
	0.7	4647
Rma cyt *c*	0.1	Job exceeded resources
	0.3	79025
	0.5	3042
	0.7	1940
ParLQ	0.1	15086
	0.3	875
	0.5	343
	0.7	343

Table C7. *Activity and selectivity Spearman's correlation.*

Library	Spearman's ρ
ParLQ-a	0.9610
ParLQ-b	0.7527
ParLQ-c	0.9335
ParLQ-d	0.9326
ParLQ-e	0.8971
ParLQ-f	0.8197
ParLQ-g	0.7097
ParLQ-h	0.9257
ParLQ-i	0.7618
Rma-CB	0.6438
Rma-CSi	0.4554

Table C8. *Correlation between ZS predictions for activity and for selectivity, both measured by Spearman's correlation.*

ZS predictor	Spearman's ρ	p-value
Hamming distance	0.9455	1.12e-05
EVmutation	0.7818	0.0045
ESM-2	0.8545	0.0008
ESM-IF	0.7273	0.0112
CoVES	0.6818	0.0208
$\Delta\Delta G_f$	0.8727	0.0005
Vina	0.8091	0.0026
GALigandDock	0.9182	6.66e-05
AF3	0.3091	0.3550
Chai-1	0.9545	4.99e-06
LigandMPNN	0.8273	0.0017
FlowSite	0.6545	0.0289
Bond distance	0.5636	0.0710
Hydrogen bonds	0.4909	0.1252
Hydrophobicity	0.9364	2.21e-05
Active-site volume	0.9000	0.0002

Table C9. *Tanimoto similarity of atom-pair fingerprints for PfTrpB non-native substrates.*

Indole Analogs	Similarity to Indole
7iodo	0.6053
7methyl	0.6053
7bromo	0.6053
5iodo	0.6000
5bromo	0.6000
5chloro	0.6000
4bromo	0.5500
6chloro	0.5366
5cyano	0.4898
4cyano	0.4894
56chloro	0.3333

Table C10. *Correlation between predictors and substrate similarity to the native substrate.*

ZS predictor	Spearman's ρ	p-value
Hamming distance	0.3890	0.2371
EVmutation	0.6390	0.0343
ESM-2	0.6390	0.0343
ESM-IF	0.5371	0.0884
CoVES	0.1574	0.6439
$\Delta\Delta G_f$	0.6390	0.0343
Vina	0.3334	0.3164
GALigandDock	0.5371	0.0884
AF3	0.3982	0.2251
Chai-1	0.3241	0.3308
LigandMPNN	0.3982	0.2251
FlowSite	0.1574	0.6439
Bond distance	0.0185	0.9569
Hydrogen bonds	0.5464	0.0820
Hydrophobicity	0.3612	0.2751
Active-site volume	0.0370	0.9139

Table C11. *Calculated reaction energy barrier (kcal/mol).*

Chemistry	Energy barrier
ParLQ	~ 9
Rma-CB	~ 11
Rma-CSi	~ 12

Table C12. *Calculated reaction energy ΔG (kcal/mol) considering all substrates and products for new-to-nature chemistries.*

Chemistry	ΔG
ParLQ-a	-44.6075
ParLQ-b	-44.7320
ParLQ-c	-44.5280
ParLQ-d	-46.3590
ParLQ-e	-45.8877
ParLQ-f	-45.5328
ParLQ-g	-46.0807
ParLQ-h	-75.2704
ParLQ-i	-45.5365
Rma-CB	-54.8434
Rma-CSi	-62.6220

Table C13. *Correlation between reaction energy and ZS predictor performance.*

ZS predictor	Spearman's ρ	p-value
Hamming distance	0.4455	0.1697
EVmutation	0.1273	0.7092
ESM-2	0.3000	0.3701
ESM-IF	-0.0545	0.8734
CoVES	0.3455	0.2981
$\Delta\Delta G_f$	0.2818	0.4011
Vina	0.2000	0.5554
GALigandDock	-0.1000	0.7699
AF3	0.4909	0.1252
Chai-1	0.3818	0.2466
LigandMPNN	0.3091	0.3550
FlowSite	0.0273	0.9366
Bond distance	-0.0364	0.9155
Hydrogen bonds	0.1364	0.6893
Hydrophobicity	0.1273	0.7092
Active-site volume	0.0455	0.8944

C.5 Bibliography for Appendix C

1. Buller, A. R. *et al.* Directed evolution of the tryptophan synthase β-subunit for stand-alone function recapitulates allosteric activation. *Proc. Natl. Acad. Sci.* **112**, 14599–14604 (2015).
2. Romney, D. K., Murciano-Calles, J., Wehrmüller, J. E. & Arnold, F. H. Unlocking reactivity of TrpB: A general biocatalytic platform for synthesis of tryptophan analogues. *J. Am. Chem. Soc.* **139**, 10769–10776 (2017).
3. Almhjell, P. J., Boville, C. E. & Arnold, F. H. Engineering enzymes for noncanonical amino acid synthesis. *Chem. Soc. Rev.* **47**, 8980–8997 (2018).
4. Boville, C. E. *et al.* Engineered biosynthesis of β-alkyl tryptophan analogues. *Angew. Chem. Int. Ed.* **57**, 14764–14768 (2018).
5. Brandenberg, O. F., Fasan, R. & Arnold, F. H. Exploiting and engineering hemoproteins for abiological carbene and nitrene transfer reactions. *Curr. Opin. Biotechnol.* **47**, 102–111 (2017).
6. Renata, H., Wang, Z. J. & Arnold, F. H. Expanding the enzyme universe: Accessing non-natural reactions by mechanism-guided directed evolution. *Angew. Chem. Int. Ed.* **54**, 3351–3367 (2015).
7. Coelho, P. S., Brustad, E. M., Kannan, A. & Arnold, F. H. Olefin cyclopropanation via carbene transfer catalyzed by engineered cytochrome P450 enzymes. *Science* **339**, 307–310 (2013).
8. Kan, S. B. J., Lewis, R. D., Chen, K. & Arnold, F. H. Directed evolution of cytochrome c for carbon–silicon bond formation: Bringing silicon to life. *Science* **354**, 1048–1051 (2016).
9. Kan, S. B. J., Huang, X., Gumulya, Y., Chen, K. & Arnold, F. H. Genetically programmed chiral organoborane synthesis. *Nature* **552**, 132–136 (2017).
10. Kille, S. *et al.* Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. *ACS Synth. Biol.* **2**, 83–92 (2013).
11. Yang, J. *et al.* Active learning-assisted directed evolution. *Nat. Commun.* **16**, 714 (2025).

12. Ding, K. *et al.* Machine learning-guided co-optimization of fitness and diversity facilitates combinatorial library design in enzyme engineering. *Nat. Commun.* **15**, 6392 (2024).
13. Almhjell, P. J. *et al.* The β-subunit of tryptophan synthase is a latent tyrosine synthase. *Nat. Chem. Biol.* **20**, 1086–1093 (2024).
14. Tinoco, A. *et al.* Origin of high stereocontrol in olefin cyclopropanation catalyzed by an engineered carbene transferase. *ACS Catal.* **9**, 1514–1524 (2019).
15. Calvó-Tusell, C., Liu, Z., Chen, K., Arnold, F. H. & Garcia-Borràs, M. Reversing the enantioselectivity of enzymatic carbene N−H insertion through mechanism-guided protein engineering. *Angew. Chem. Int. Ed.* **62**, e202303879 (2023).
16. Li, F.-Z. *et al.* Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes. 2024.10.24.619774 Preprint at https://doi.org/10.1101/2024.10.24.619774 (2024).
17. Bento, A. P. *et al.* An open source chemical structure curation pipeline using RDKit. *J. Cheminformatics* **12**, 51 (2020).
18. Landrum, G. RDKit Documentation.
19. O'Boyle, N. M. *et al.* Open Babel: An open chemical toolbox. *J. Cheminformatics* **3**, 33 (2011).
20. Hanwell, M. D. *et al.* Avogadro: An advanced semantic chemical editor, visualization, and analysis platform. *J. Cheminformatics* **4**, 17 (2012).
21. Case, D. A. *et al.* AmberTools. *J. Chem. Inf. Model.* **63**, 6183–6191 (2023).
22. Dauparas, J. *et al.* Atomic context-conditioned protein sequence design using LigandMPNN. Preprint at https://doi.org/10.1101/2023.12.22.573103 (2023).
23. Stärk, H., Jing, B., Barzilay, R. & Jaakkola, T. Harmonic self-conditioned flow matching for multi-ligand docking and binding site design. Preprint at https://doi.org/10.48550/arXiv.2310.05764 (2024).
24. Adasme, M. F. *et al.* PLIP 2021: Expanding the scope of the protein–ligand interaction profiler to DNA and RNA. *Nucleic Acids Res.* **49**, W530–W534 (2021).
25. Kyte, J. & Doolittle, R. F. A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.* **157**, 105–132 (1982).

26. Hopp, T. P. & Woods, K. R. A computer program for predicting protein antigenic determinants. *Mol. Immunol.* **20**, 483–489 (1983).

27. Eisenberg, D., Schwarz, E., Komaromy, M. & Wall, R. Analysis of membrane and surface protein sequences with the hydrophobic moment plot. *J. Mol. Biol.* **179**, 125–142 (1984).

28. Tien, M. Z., Meyer, A. G., Sydykova, D. K., Spielman, S. J. & Wilke, C. O. Maximum allowed solvent accessibilites of residues in proteins. *PLoS ONE* **8**, e80635 (2013).

29. Sriramulu, D. K. & Lee, S.-G. Combinatorial effect of ligand and ligand-binding site hydrophobicities on binding affinity. *J. Chem. Inf. Model.* **60**, 1678–1684 (2020).

30. Tian, W., Chen, C., Lei, X., Zhao, J. & Liang, J. CASTp 3.0: Computed atlas of surface topography of proteins. *Nucleic Acids Res.* **46**, W363–W367 (2018).

31. Carhart, R. E., Smith, D. H. & Venkataraghavan, R. Atom pairs as molecular features in structure-activity studies: Definition and applications. *J. Chem. Inf. Comput. Sci.* **25**, 64–73 (1985).

32. Neese, F. The ORCA program system. *WIREs Comput. Mol. Sci.* **2**, 73–78 (2012).

33. Garcia-Borràs, M. *et al.* Origin and control of chemoselectivity in cytochrome c catalyzed carbene transfer into Si–H and N–H bonds. *J. Am. Chem. Soc.* **143**, 7114–7123 (2021).

34. Huang, X. *et al.* A biocatalytic platform for synthesis of chiral α-trifluoromethylated organoborons. *ACS Cent. Sci.* **5**, 270–276 (2019).

35. Wei, Y., Tinoco, A., Steck, V., Fasan, R. & Zhang, Y. Cyclopropanations via heme carbenes: Basic mechanism and effects of carbene substituent, protein axial ligand, and porphyrin substitution. *J. Am. Chem. Soc.* **140**, 1649–1662 (2018).