LatentDE: Latent-based Directed Evolution accelerated by Gradient Ascent for Protein Sequence Design

Thanh V. T. Tran[*,1] **Nhat Khang Ngo**[*,†,1] **Viet Thanh Duy Nguyen**[1] **Truong Son Hy**[‡,2]
[1] FPT Software AI Center, Vietnam [2] University of Alabama at Birmingham, United States

Abstract

Directed evolution has been the most effective method for protein engineering that optimizes biological functionalities through a resource-intensive process of screening or selecting among a vast range of mutations. To mitigate this extensive procedure, recent advancements in machine learning-guided methodologies center around the establishment of a surrogate sequence-function model. In this paper, we propose Latent-based Directed Evolution (LDE), an evolutionary algorithm designed to prioritize the exploration of high-fitness mutants in the latent space. At its core, LDE is a regularized variational autoencoder (VAE), harnessing the capabilities of the state-of-the-art Protein Language Model (pLM), ESM-2, to construct a meaningful latent space of sequences. From this encoded representation, we present a novel approach for efficient traversal on the fitness landscape, employing a combination of gradient-based methods and directed evolution. *In-silico* evaluations conducted on eight protein sequence design tasks demonstrate the superior performance of our proposed LDE over previous baseline algorithms. Our implementation is publicly available at `https://github.com/HySonLab/LatentDE`.

1 Introduction

Protein engineering aims to design proteins with specific functions, determined by their amino acid sequences that fold into three-dimensional structures [15, 11], creating a fitness landscape that maps sequences to functions [52]. Inspired by natural evolution, *directed evolution* [2] iteratively mutates and selects protein variants to optimize desired functions.

Recent machine learning (ML) methods have been studied to improve the sample efficiency of this evolutionary search [31, 50, 32, 13]. However, these methods often rely on costly, repetitive rounds of mutagenesis and validation in a vast, discrete sequence space. The search space of protein sequences is discrete and combinatorially large, and most protein have low fitness [6], making searching algorithms prone to local optima and false positives [6, 12, 5]. Latent space optimization (LSO) offers a more efficient alternative by using continuous, low-dimensional representations, but its integration with directed evolution remains underexplored.

Protein design is a black-box model-based optimization (MBO) problem, where the objective is to identify sequences maximizing fitness. While online MBO iteratively proposes designs based on feedback, its effectiveness is limited by the availability of experimental data and computational resources. Offline MBO, which relies on a static dataset, is more efficient but can struggle with

[*] Equal contribution
[†] Now at Mila, McGill University
[‡] Correspondence at thy@uab.edu

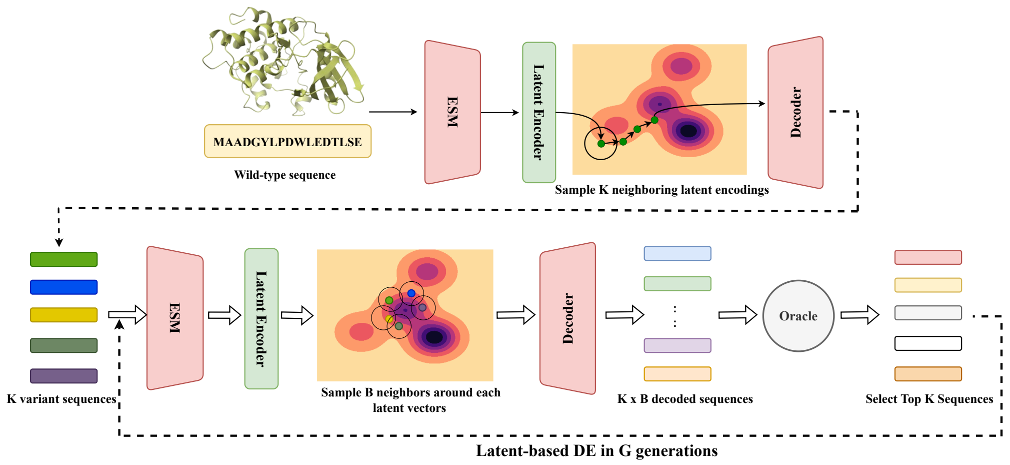

Figure 1: Overview of our proposed method. **Top**: Encoding of the wild-type sequence into latent variables and gradient ascent towards high-fitness regions. **Bottom**: Latent-based directed evolution performed over G generations, sampling B neighbors around the latent variables for evaluation by a black-box oracle.

out-of-distribution inputs, causing the learned model $\hat{f}(x)$ to predict fitness inaccurately [25], thereby fooling the discovery algorithm to produce non-desirable designs.

To address these challenges, we propose **L**atent-based **D**irected **E**volution (LDE), the first method to integrate directed evolution within a latent space. Leveraging the latent space of ESM-2 [26],LDE learns to reconstruct and predict the fitness value of the input sequences in the form of a variational autoencoder (VAE) regularized by supervised signals. Post-training, combining the strengths of both MBO approaches, LDE first uses gradient ascent to guide latent representations toward high-fitness regions and then integrates directed evolution by introducing noise to explore locally. The noised latent representations are decoded into sequences and evaluated by the truth oracles, and top samples are selected for the next round of the algorithm, enabling an efficient gradient-free sampling strategy.

2 Method

Optimizing the latent variables of generative models has proven effective in various drug and protein design tasks [19, 40]. Our work introduces a novel approach that utilizes directed evolution to efficiently discover optimal protein sequences directly from their latent representations. The theoretical background for Directed Evolution (DE) and Latent Space Optimization (LSO), which form the basis of our approach, is discussed in detail in Appendix A. We begin with the problem formulation in Section 2.1. Then, we present our pre-trained regularized variational autoencoders (VAEs) [22] in Section 2.2 and how to perform directed evolution, which is accelerated by gradient ascent, in the latent space of VAEs in Section 2.3. We describe our model architecture in Appendix B.

2.1 Problem Formulation

We aim to find high-fitness protein sequences s in the sequence space $\mathcal{V}^L$, where $\mathcal{V}$ is the vocabulary of amino acids and L is the sequence length. The goal is to maximize a black-box protein fitness function $\mathcal{O} : \mathcal{V}^L \mapsto \mathbb{R}$, which can only be evaluated through wet-lab experiments:

$$x^* = \text{argmax}_{x \in \mathcal{V}^L} \mathcal{O}(x). \tag{1}$$

For in-silico evaluation, we use an oracle $\mathcal{O}_\psi$ parameterized by ψ to approximate $\mathcal{O}$. Given a training subset $\mathcal{D}_t$, our task is to generate sequences $\hat{x}$ that optimize the fitness predicted by $\mathcal{O}_\psi$. We focus on designing sequences starting from a wild-type sequence and iteratively generating candidates with improved properties, as illustrated in Figure 1.

2.2 Regularized Variational Autoencoder

Given a dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^N$ of protein sequences $x_i \in \mathcal{V}^L$ with fitness values $y_i \in \mathbb{R}$, we train a VAE with an encoder $\phi : \mathcal{V}^L \mapsto \mathbb{R}^{d_h}$ and a decoder $\theta : \mathbb{R}^d \mapsto \mathcal{V}^L$. The VAE encodes each sequence into a latent vector $z_i \in \mathbb{R}^d$ sampled from $\mathcal{N}(\mu_i, \sigma_i^2)$ and reconstructs it back to $\hat{x}_i \in \mathcal{V}^L$. The training objective combines cross-entropy loss $\mathcal{C}(\hat{x}_i, x_i)$ and Kullback-Leibler (KL) divergence:

$$\mathcal{L}_{vae} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{C}(\hat{x}_i, x_i) + \frac{\beta}{N}\sum_{i=1}^{N} D_{\mathrm{KL}}(\mathcal{N}(\mu_i, \sigma_i^2)\|\mathcal{N}(0, I_d)). \tag{2}$$

where β controls the disentanglement of the VAE's latent space.

Latent Space Regularization The KL-divergence regularizer in Equation (2) encourages the encoder ϕ to produce a latent space close to the unit Gaussian prior. However, forcing the approximate posterior $q_\phi(z|x)$ to be unit Gaussian may lead to the fact that the encoder network ϕ may encode zero information into the latent space [9], resulting in less meaningful latent representations. This is not a good phenomenon for our latent-based directed evolution algorithm that aims at locally searching for high-fitness proteins by iteratively modifying the reference sequences (see Appendix A.1). Regarding learning protein fitness landscape, the percentage of high-fitness sequences in datasets is relatively small compared to those with low fitness scores. Consequently, the latent representations of high-fitness proteins may be hidden within a dense of low-fitness ones [8], leading to inefficient searching processes. To address this, we jointly train the VAE with a fitness predictor $f : \mathbb{R}^d \mapsto \mathbb{R}$ to enhance the separation of high-fitness representations:

$$\mathcal{L} = \mathcal{L}_{vae} + \frac{1}{N}\sum_{i=1}^{N}||f(z_i) - y_i||_2^2, \tag{3}$$

where y_i is the experimental fitness of sequence x_i, and $z_i = \phi(x_i)$. This L_2 regularization aligns latent representations with their fitness scores.

Algorithm 1 Latent-based directed evolution accelerated by gradient ascent

Input: x^{wt}, encoder ϕ, decoder θ, fitness predictor f, # iterations T, # generations G, beam size B, oracle $\mathcal{O}$.

```
1: $\mathcal{P}_0 \leftarrow \emptyset$
2: for $i = 1$ to $K$ do
3:     Sample $z \sim \mathcal{N}(\mu_\phi(x^{wt}), \sigma_\phi(x^{wt})^2)$
4:     $z_T \leftarrow$ gradient_ascent$(z, f, T)$ in Equation (5)
5:     $\hat{x} \leftarrow \theta(z_T)$
6:     $P_0 \leftarrow P_0 \cup \{(\hat{x}, \mathcal{O}(\hat{x}))\}$
7: end for
8: for $g = 1$ to $G$ do
9:     $P_g \leftarrow \emptyset$
10:    for $k = 1$ to $K$ do
11:        for $b = 1$ to $B$ do
12:            Sample $z_{k,b} \sim \mathcal{N}(\mu_\phi(x_k), \sigma_\phi(x_k)^2)$
13:            $p \sim \mathcal{U}[0, 1)$
14:            if $p >$ threshold then
15:                Inject noise to $z_{k,b}$ based on Equation (6)
16:            end if
17:            $\hat{x}_{k,b} \leftarrow \theta(z_{k,b})$
18:            $P_g \leftarrow P_g \cup \{(\hat{x}_{k,b}, \mathcal{O}(\hat{x}_{k,b}))\}$
19:        end for
20:        $P_g \leftarrow P_g \cup P_{g-1}$
21:        $P_g \leftarrow$ topK$(P_g)$
22:    end for
23: end for
```

2.3 Latent-Based Directed Evolution

Inspired by nature's evolutionary process, where subtle mutations in a protein's sequence enhance fitness, our method explores the latent space around the wild-type sequence through iterative sampling from the pretrained encoder ϕ.

A key challenge is that the latent representation z of x^{wt} may reside in low-fitness regions, slowing convergence. To address this, we use gradient ascent to initialize the population $\mathcal{P}_0$, guiding the search toward higher fitness areas. Algorithm 1 outlines our approach, with further details on the evolutionary process and sampling techniques provided in Appendix C.

2.4 Efficient Sampling and Optimization

Traditional directed evolution is effective but computationally intensive for protein design. Our latent-based algorithm improves sampling efficiency by compressing long sequences into low-dimensional latent representations via VAEs, simplifying the mutation process as noise addition in latent space. This approach, unlike complex mutation operators [3, 44], requires no domain knowledge and is more computationally efficient. The gradient ascent benefits from VAE regularization, creating a smoother optimization landscape and reducing the risk of local optima. Furthermore, even if the optimization converges to a local optimum, latent-based directed evolution (DE) can explore surrounding regions through controlled random perturbations for a broader search, enabling a more comprehensive search of promising areas.

3 *In-slico* Experiments

We conducted a comprehensive set of experiments to evaluate the effectiveness of our proposed LDE in protein sequence design. The experimental setup, including datasets, implementation details, baseline algorithms, oracles, and evaluation metrics, is described in Appendix D. Complete results of these experiments are provided in Appendix E.

Table 1: Maximum fitness scores across eight protein datasets. Shaded rows indicate the result of ablation studies.

	avGFP	AAV	TEM	E4B	AMIE	LGK	Pab1	UBE2I
x^{wt}	1.408	-6.778	-0.015	0.774	-2.789	-1.260	0.014	-0.262
AdaLead [38]	3.323	-1.545	0.248	-0.373	-1.483	-0.047	0.382	2.765
DyNA PPO [1]	5.331	-2.817	0.570	-0.575	-2.790	-0.060	0.183	2.630
CbAS [5]	5.187	-2.800	0.481	-0.658	-1.784	-0.056	0.276	2.693
CMA-ES [16]	5.125	-3.267	0.590	-0.658	-2.790	-0.086	0.254	2.527
COMs [43]	3.544	-3.533	0.472	-0.860	-20.182	-0.087	0.156	2.086
PEX [33]	3.796	2.378	0.252	4.317	-0.364	0.009	1.326	3.578
GFN-AL [18]	5.028	-4.444	0.654	-0.831	-37.360	-5.738	1.399	3.850
GGS [23]	3.368	2.442	1.121	-1.147	-3.364	-0.972	0.059	4.101
LDE (ours)	**8.058**	**2.636**	**1.745**	**5.120**	-0.103	**0.018**	1.548	**4.297**
– w/o GA	6.407	2.148	1.220	4.597	**-0.099**	-0.531	**1.592**	3.254
– w/o DE	3.677	0.919	-0.024	3.052	-0.701	-1.597	0.285	0.766

Comparison with Baseline Algorithms As shown in Table 1, LDE outperforms other algorithms across eight protein benchmarks, especially for longer sequences (e.g., avGFP with 237 amino acids). This highlights LDE's advantage in efficiently exploring complex, high-dimensional protein spaces. While AAV contains shorter sequences (up to 28 amino acids), LDE's latent representations still enhance optimization. The diversity and novelty metrics in Tables 5 and 6 provide further insights into the exploration-exploitation trade-offs of various methods. It is important to note that maximizing these metrics does not always indicate better overall performance, as different algorithms have different objectives (e.g., optimizing fitness scores, maximizing diversity) [39, 33, 23]. For example, GFN-AL and COMs achieve high diversity but low fitness scores in the AMIE dataset.

Impact of Gradient Ascent and Directed Evolution We conduct an ablation study to demonstrate how each component of our method contributes to the performance. In particular, we compare LDE with its variants, including (1) without gradient ascent (w/o GA): we do not use GA as the warm-up step for the latent-based DE and (2) without directed evolution (w/o DE): we only run gradient ascent to optimize fitness of the sequences. As shown in shaded rows of Table 1, removing gradient ascent (i.e., w/o GA) notably reduces the performance of the evolution algorithm. Similarly, performing gradient ascent without DE only leads to sub-optimal solutions.

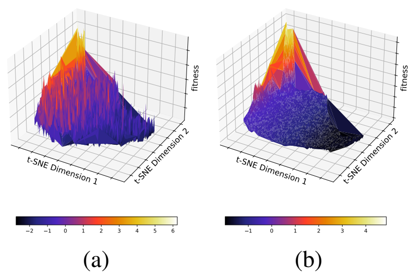

Figure 2: The fitness landscape approximated by regularized VAEs of E4B proteins.

Latent Space Visualization Figures 2a and 2b illustrate the approximated fitness landscapes of protein sequences in the E4B family. These suggest that applying VAE regularization, as in Equation (3), organizes latent representations by fitness. Figure 2a demonstrates that our VAEs generate meaningful latent representations based on true fitness values, while Figure 2b effectively approximates a smooth fitness landscape, enabling effective gradient-based optimization methods. These results highlight the effectiveness of our latent-based directed evolution approach.

Effect of Latent Dimension To have a comprehensive understanding of latent-based DE, we study the effect of latent dimensions on the evolution process. Table 2 shows that a large latent dimension can lead to worse performance in optimizing protein fitness, while smaller latent sizes can result in sub-optimal fitness. This is an expected behavior in LSO where high dimensions may impose challenges for optimization algorithms and lower dimensions can not encode sufficient information about the input data. Furthermore, we observe that novelty and diversity are independent of good fitness and vary specifically for different fitness landscapes.

Table 2: Reported metrics of LDE with different latent sizes on two datasets. All experiments use learning rate $\alpha = 0.002$ and exploration step size $\gamma = 3$.

Dataset	Latent Dim	Fitness	Novelty	Diversity
E4B (L = 102)	128	4.433	2.040	5.322
	256	4.597	0.615	0.968
	320	**5.120**	2.038	4.504
	512	3.052	0.862	0.968
TEM (L = 286)	128	1.408	17.109	44.906
	256	1.220	267.013	1.783
	320	**1.745**	62.508	61.652
	512	-0.024	269.992	0.044

Active Learning Protein fitness datasets are often fragmented due to the cost of wet lab experiments [12], and even then, they only capture a small protein of real-world protein behavior. This limitation can trap machine learning models, which learn from training samples, in local optima, hindering their generalizability and accuracy [5]. To overcome this issue, in this work, we propose using active learning to update the fitness landscape approximated by our regularized VAEs. Indeed, Algorithm 2 demonstrates our method in detail.

To validate our proposed method, we conduct additional experiments on the four smallest benchmark datasets, each containing fewer than $8,000$ training data points: TEM, AMIE, LGK, and UBE2I. For this demonstration, we set the number of active learning rounds N to 10 and decrease the number of directed evolution iterations G to 5. Additionally, in each active learning loop, the regularized VAE M is fine-tuned in 30 epochs. As outlined in Table 3, LDE demonstrates improved performance when combined with active learning. These results empirically validate our hypothesis and confirm the efficacy of our proposed method.

Table 3: Max fitness scores on four smallest protein datasets.

Model	TEM	AMIE	LGK	UBE2I
LDE	1.095	-0.558	-0.005	2.976
− w/ active	**2.167**	**-0.015**	**0.022**	**3.698**

4 Conclusion and Future Work

In this work, we present Latent-based Directed Evolution (LDE), a novel method that combines directed evolution with gradient ascent in a regularized VAE latent space to efficiently optimize and design protein sequences. This approach leverages deep representation learning capabilities of generative models to significantly speed up the evolutionary process, achieving superior results compared to traditional methods solely operating in sequence space. LDE holds significant promise for accelerating protein engineering and drug discovery efforts, and we invite further research on integrating it with in vitro protein characterization for real-world validation.

On-going Direction However, our present work is not complete and has limitations, including its reliance on a single fitness predictor that could destabilize the optimization if poorly calibrated. To address this, we are planning to use ensembles of surrogate models with risk-aware strategies to handle uncertainty, along with robustness checks and sensitivity analyses to examine model stability and parameter impact. Previous studies [30, 47] have explored this issue and shown promising results. A potential extension would involve integrating structural information [45, 42] into the optimization process, ensuring that the generated structures closely resemble the wild-type to maintain functionality. Additionally, LDE could be adapted for multi-objective optimization. These ongoing efforts will enhance the robustness, applicability, and effectiveness of our method, connecting computational predictions with real-world biological outcomes.

References

[1] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2020.

[2] Frances H. Arnold. Directed evolution: Creating biocatalysts for the future. *Chemical Engineering Science*, 51(23):5091–5102, 1996.

[3] Frances H Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998.

[4] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In Stefan Riezler and Yoav Goldberg, editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics.

[5] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 773–782. PMLR, 09–15 Jun 2019.

[6] David H Brookes, Amirali Aghazadeh, and Jennifer Listgarten. On the sparsity of fitness functions and implications for learning. *Proceedings of the National Academy of Sciences*, 119(1):e2109649118, 2022.

[7] Drew H. Bryant, Ali Bashir, Sam Sinai, Nina K. Jain, Pierce J. Ogden, Patrick F. Riley, George M. Church, Lucy J. Colwell, and Eric D. Kelsic. Deep diversification of an aav capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, Jun 2021.

[8] Egbert Castro, Abhinav Godavarthi, Julian Rubinfien, Kevin Givechian, Dhananjay Bhaskar, and Smita Krishnaswamy. Transformer-based protein generation with regularized latent space optimization. *Nature Machine Intelligence*, 4:1–12, 09 2022.

[9] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017.

[10] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[11] Cyrus Chothia. Principles that determine the structure of proteins. *Annual Review of Biochemistry*, 53(1):537–572, June 1984.

[12] Christian Dallago, Jody Mou, Jody Mou, Kadina Johnston, Bruce Wittmann, Nicholas Bhattacharya, Samuel Goldman, Ali Madani, and Kevin Yang. Flip: Benchmark tasks in fitness landscape inference for proteins. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

[13] Patrick Emami, Aidan Perreault, Jeffrey Law, David Biagioni, and Peter St. John. Plug and play directed evolution of proteins with gradient-based discrete mcmc. *Machine Learning: Science and Technology*, 4(2):025014, April 2023.

[14] Elad Firnberg, Jason W. Labonte, Jeffrey J. Gray, and Marc Ostermeier. A Comprehensive, High-Resolution Map of a Gene's Fitness Landscape. *Molecular Biology and Evolution*, 31(6):1581–1592, 02 2014.

[15] N Go. Theoretical studies of protein folding. *Annual Review of Biophysics and Bioengineering*, 12(1):183–210, June 1983.

[16] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[17] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with GFlowNets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9786–9801. PMLR, 17–23 Jul 2022.

[18] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[19] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2323–2332. PMLR, 10–15 Jul 2018.

[20] Kadina E Johnston, Clara Fannjiang, Bruce J Wittmann, Brian L Hie, Kevin K Yang, and Zachary Wu. Machine learning for protein engineering. *arXiv preprint arXiv:2305.16634*, 2023.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[22] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[23] Andrew Kirjner, Jason Yim, Raman Samusevich, Shahar Bracha, Tommi S Jaakkola, Regina Barzilay, and Ila R Fiete. Improving protein optimization with smoothed fitness landscapes. In *The Twelfth International Conference on Learning Representations*, 2023.

[24] Justin R. Klesmith, John-Paul Bacik, Ryszard Michalczyk, and Timothy A. Whitehead. Comprehensive sequence-flux mapping of a levoglucosan utilization pathway in e. coli. *ACS Synthetic Biology*, 4(11):1235–1243, September 2015.

[25] Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5126–5137. Curran Associates, Inc., 2020.

[26] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[27] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[28] John Maynard Smith. Natural selection and the concept of a protein space. *Nature*, 225(5232):563–564, February 1970.

[29] Daniel Melamed, David L. Young, Caitlin E. Gamble, Christina R. Miller, and Stanley Fields. Deep mutational scanning of an rrm domain of the saccharomyces cerevisiae poly(a)-binding protein. *RNA*, 19(11):1537–1551, September 2013.

[30] Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. Improving black-box optimization in VAE latent space using decoder uncertainty. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[31] Yuchi Qiu, Jian Hu, and Guo-Wei Wei. Cluster learning-assisted directed evolution. *Nature Computational Science*, 1(12):809–818, December 2021.

[32] Yuchi Qiu and Guo-Wei Wei. Clade 2.0: Evolution-driven cluster learning-assisted directed evolution. *Journal of Chemical Information and Modeling*, 62(19):4629–4641, September 2022.

[33] Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18520–18536. PMLR, 17–23 Jul 2022.

[34] Karen S. Sarkisyan, Dmitry A. Bolotin, Margarita V. Meer, Dinara R. Usmanova, Alexander S. Mishin, George V. Sharonov, Dmitry N. Ivankov, Nina G. Bozhanova, Mikhail S. Baranov, Onuralp Soylemez, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor A. Kondrashov. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, May 2016.

[35] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[36] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[37] Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek Abdelzaher. ControlVAE: Controllable variational autoencoder. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8655–8664. PMLR, 13–18 Jul 2020.

[38] Sam Sinai, Richard Wang, Alexander Whatley, Stewart Slocum, Elina Locane, and Eric D. Kelsic. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *CoRR*, abs/2010.02141, 2020.

[39] Zhenqiao Song and Lei Li. Importance weighted expectation-maximization for protein sequence design. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32349–32364. PMLR, 23–29 Jul 2023.

[40] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20459–20478. PMLR, 17–23 Jul 2022.

[41] Lea M. Starita, Jonathan N. Pruneda, Russell S. Lo, Douglas M. Fowler, Helen J. Kim, Joseph B. Hiatt, Jay Shendure, Peter S. Brzovic, Stanley Fields, and Rachel E. Klevit. Activity-enhancing mutations in an e3 ubiquitin ligase identified by high-throughput mutagenesis. *Proceedings of the National Academy of Sciences*, 110(14):E1263–E1272, 2013.

[42] Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. In *The Twelfth International Conference on Learning Representations*, 2024.

[43] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10358–10368. PMLR, 18–24 Jul 2021.

[44] Thanh V. T. Tran and Truong Son Hy. Protein design by directed evolution guided by large language models. *bioRxiv*, 2024.

[45] Michel van Kempen, Stephanie S. Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron L. M. Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature Biotechnology*, 42(2):243–246, Feb 2024.

[46] Yajie Wang, Pu Xue, Mingfeng Cao, Tianhao Yu, Stephan T. Lane, and Huimin Zhao. Directed evolution: Methodologies and applications. *Chemical Reviews*, 121(20):12384–12444, Oct 2021.

[47] Yanzheng Wang, TIANYU SHI, and Jie Fu. Sample-efficient antibody design through protein language model for risk-aware batch bayesian optimization. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[48] Jochen Weile, Song Sun, Atina G Cote, Jennifer Knapp, Marta Verby, Joseph C Mellor, Yingzhou Wu, Carles Pons, Cassandra Wong, Natascha van Lieshout, Fan Yang, Murat Tasan, Guihong Tan, Shan Yang, Douglas M Fowler, Robert Nussbaum, Jesse D Bloom, Marc Vidal, David E Hill, Patrick Aloy, and Frederick P Roth. A framework for exhaustively mapping functional missense variants. *Molecular Systems Biology*, 13(12):957, 2017.

[49] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[50] Bruce J. Wittmann, Yisong Yue, and Frances H. Arnold. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Systems*, 12(11):1026–1045.e7, November 2021.

[51] Emily E. Wrenbeck, Laura R. Azouz, and Timothy A. Whitehead. Single-mutation fitness landscapes for an enzyme on multiple substrates reveal specificity is globally encoded. *Nature Communications*, 8(1):15695, Jun 2017.

[52] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the XI International Congress of Genetics*, 8:209–222, 1932.

A Preliminaries

A.1 Directed Evolution Theory

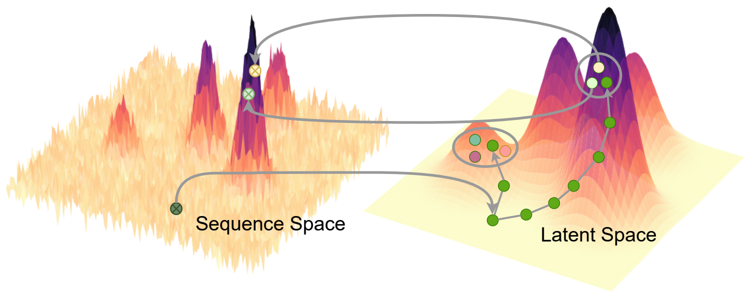

Figure 3: We perform directed evolution within the smooth and continuous latent space of a generative model. Our algorithm begins by identifying high-fitness regions through gradient ascent. Within these regions, we strategically sample a defined number of neighboring points (gray circles) to create a diverse population for the evolutionary process. This population then undergoes iterative selection and mutation within the latent space, ultimately converging to sequences with enhanced fitness.

Directed evolution (DE) is a widely used approach in protein engineering, focusing on identifying optimal protein sequences from a large set of unlabeled candidates $\mathcal{S}$ with minimal experimental validation [20, 46]. DE involves iterative cycles of mutation and selection, where protein variants with enhanced properties are chosen for subsequent rounds. This process effectively explores local regions of high-fitness proteins within the vast sequence space, capitalizing on the clustering of functional proteins observed in [28]. As depicted in the left part of Figure 3, DE begins with wild-type sequences, applying targeted mutations to discover variants with improved functions.

A.2 Latent Space Optimization

Latent space optimization (LSO) is a model-based technique that operates within the latent space $\mathcal{Z}$ of deep generative models. Here, an objective function $f : \mathcal{Z} \mapsto \mathbb{R}$ predicts objective values directly from latent representations. LSO is particularly effective when $\mathcal{Z}$ is low-dimensional and continuous, simplifying optimization challenges typically encountered in discrete, high-dimensional spaces. This transformation enables the application of techniques like gradient ascent and Bayesian optimization (BO) [36]. Moreover, f is trained using an encoder $\phi : \mathcal{X} \mapsto \mathcal{Z}$, which maps an input sample $x \in \mathcal{X}$ to its corresponding latent representation $z \in \mathcal{Z}$.

B Implementation Details

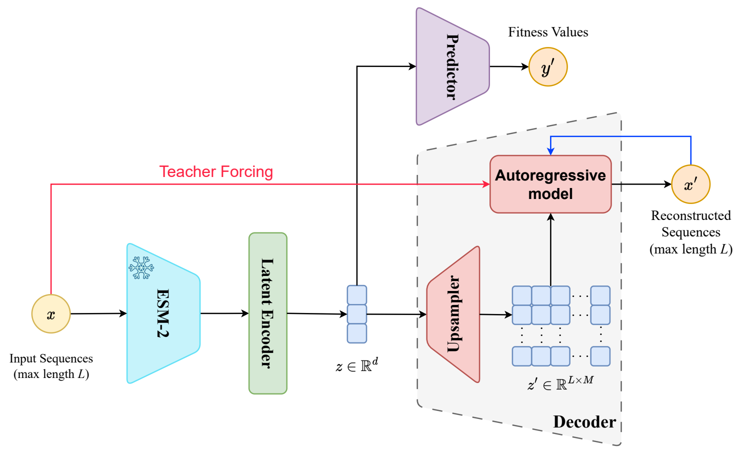

Figure 4: Schematic illustration of the VAE model used in the study.

B.1 VAEs' Architecture

In this section, we go into detail regarding the architecture of the VAE used in our study. As mentioned in the main text, our regularized VAE consists of an encoder, a predictor, and a decoder.

Encoder Figure 4 depicts that the encoder incorporates a pre-trained ESM-2 [26] followed by a latent encoder to compute the latent representation z. In our study, we leverage the powerful representation of the pre-trained 30-layer ESM-2 by making it the encoder of our model. Given an input sequence $x = \langle x_0, x_1, \cdots, x_L \rangle$, where $x_i \in \mathcal{V}$, the transformer-based ESM-2 computes representations for each token x_i in x, resulting in a token-level hidden representation $H = \langle h_0, h_1, \cdots, h_L \rangle, h_i \in \mathbb{R}^{d_h}$. We calculate the global representation $h \in \mathbb{R}^{d_h}$ of x via a weighted sum of its tokens:

$$h = \sum_{i=1}^{L} \frac{\omega^T \exp(h_i)}{\sum_{i=1}^{L} \omega^T \exp(h_i)} h_i. \tag{4}$$

here, ω is a learnable global attention vector. Then, two multi-layer perceptrons (MLPs) are used to compute $\mu = \text{MLP}_1(h)$ and $\log \sigma = \text{MLP}_2(h)$, where the latent dimension is d. Finally, a latent representation $z \in \mathbb{R}^d$ is sampled from $\mathcal{N}(\mu, \sigma^2)$, which is further proceeded to the decoder to reconstruct the sequence $\hat{x}$. We use an auxiliary MLP as a **fitness predictor** that maps the latent z to the fitness score, i.e. $y' = \text{MLP}(z)$. The hidden dimensional size of the predictor is set to 512, with the dropout of 0.2. We set $d_h = 1280$ and $d = 320$ for the main experiments in our study.

Decoder Inspired from [35], we construct our decoder as the combination of two components: an 'upsampler' component comprising 3 layers of transposed convolutions with stride of 2 to upsample the latent vector z to a sequence matching the output sequence's length; and an autoregressive component consisting of an attention-based GRU [10, 27] with 512 units. To cope with the optimization difficulties reported when training VAE with powerful autoregressive decoders [4, 35], we follow [4] by applying 40% dropout to the amino acid context supplied as input to the GRU during training. This encourages the network to depend on the information conveyed through the upsampled latent code along with the conditional information in the masked amino acid sequence to make predictions. Additionally, we apply teacher forcing [49] with a ratio of 50% for faster convergence.

B.2 Training Configurations

We employ the ControlVAE mechanism to prevent KL vanishing and enhance the diversity of generated data during training. Across all tasks, we configure the coefficients K_p and K_i of the P term and I term to 0.01 and 0.0001, respectively. For the LGK benchmark, the desired KL-divergence C is set to 40, while for all other tasks, $C = 20$ is utilized. The batch size for each task is determined to be as large as possible, as long as the total steps in one epoch for each task are higher than 100.

C Detailed Methodology

Gradient Ascent (GA) At the beginning of our algorithm, the wild-type protein sequence x^{wt} is encoded by the encoder ϕ to produce its mean $\mu_\phi(x^{wt})$ and log variance $\log \sigma_\phi(x^{wt})$. A latent representation z is then sampled from $\mathcal{N}(\mu_\phi(x^{wt}), \sigma_\phi(x^{wt})^2)$. Subsequently, we perform gradient ascent with a learning rate of α in T iterations to move z to high-fitness regions as:

$$z_{t+1} = z_t + \alpha \nabla_z f(z)|_{z=z_t}, 0 \le t < T, \tag{5}$$

where $\nabla_z f(z)|_{z=z_t}$ is the gradient of the fitness predictor f with respect to z_t. After T iterations, we add z_T to the initial population $\mathcal{P}_0$. The procedure is iteratively executed until K pairs of decoded sequences along with their respective fitness scores are obtained. At this stage, the fitness scores are computed by the latent predictor f, allowing fast computation and efficient latent sampling (see lines 1 to 7).

Evolutionary Process From lines 8 to 23, a latent-based directed evolution is conducted in G generations to generate K protein sequences with high fitness values. For each candidate x_k, we sample B latent variables from its posterior distribution, where each is denoted as $z_{k,b} \sim \mathcal{N}(\mu_\phi(x_k), \sigma_\phi(x_k)^2)$. These latent representations are then decoded into sequences by the decoder θ, i.e. $\hat{x}_{k,b} = \theta(z_{k,b})$. Wet-lab experiments then evaluate the decoded sequences to obtain their fitness scores. All sequences generated in generation g are added into $\mathcal{P}_g$. Finally, at the end of g, the top K sequences are selected from both $\mathcal{P}_{g-1}$ and $\mathcal{P}_g$.

Random Exploration Although sampling around the local areas of high-fitness latent codes can guarantee the superiority of the generated sequences, the search process may be prone to be trapped in these local regions after a certain number of generations, thereby hindering the exploration of potentially promising sequences, which are unseen before. As a result, we randomly add white noise to the latent variables when $p \sim \mathcal{U}[0, 1)$ is higher than a threshold. As demonstrated in line 14 of Algorithm 1, the formula is defined as:

$$z_l = z + (\gamma - \delta g)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{6}$$

where $\gamma \in \mathbb{R}$ denotes the step size that controls the exploration rate, and δ is the annealing factor at the generation g of the evolution process. We hypothesize that when g gets close to G, i.e., the total number of generations, the population tends to contain superior samples; thus, we slow down the exploration to avoid degeneration at the end of the algorithm.

D Experimental Setup

Table 4: Detailed information and statistics of the eight protein datasets.

Dataset	Organism	Protein	Optimization Target	Length	Size	Percentiles 0.25	0.50	0.75
avGFP [34]	Aequorea victoria	GFP	Brightness	237	$51,715$	1.428	3.287	3.161
AAV [7]	Homo sapiens	VP1	AAV viabilities	28	$42,330$	-3.964	-0.840	1.321
TEM [14]	Escherichia coli	TEM-1 β-Lactamase	Thermodynamic stability	286	$5,199$	0.049	0.444	0.934
E4B [41]	Mus musculus	UBE4B	Ubiquitin ligase activity	102	$91,032$	-1.830	-0.984	-0.093
AMIE [51]	Escherichia coli	Amidase	Hydrolysis activity	341	$6,417$	-1.228	-0.666	-0.263
LGK [24]	Lipomyces starkeyi	Levoglucosan kinase	Levoglucosan utilization	439	$7,633$	-0.871	-0.562	-0.394
Pab1 [29]	Saccharomyces cerevisiae	Poly(A)-binding	mRNA binding	75	$36,389$	-0.116	-0.022	0.036
UBE2I [48]	Homo sapiens	UBE2I	Growth rescue rate	159	$3,022$	0.068	0.492	0.766

Datasets Following [33] and [39], we assess the performance of our method across eight protein engineering benchmarks: (1) Green Fluorescent Protein (**avGFP**), (2) Adeno-Associated Viruses (**AAV**), (3) TEM-1 β-Lactamase (**TEM**), (4) Ubiquitination Factor Ube4b (**E4B**), (5) Aliphatic Amide Hydrolase (**AMIE**), (6) Levoglucosan Kinase (**LGK**), (7) Poly(A)-binding Protein (**Pab1**), (8) SUMO E2 Conjugase (**UBE2I**). The comprehensive dataset information, including the protein name, organism, optimization target, sequence length, data size, and data percentiles, is provided in Table 4. Each dataset represents distinct optimization target, which we simplify as a singular term *"fitness"* when reporting the benchmark results. Detailed descriptions of the data are provided in the Supplementary Material.

Implementation Details The model training is conducted using a single NVIDIA A100 card, employing the Adam optimization algorithm [21] with a learning rate of 2e-4. Each dataset is randomly split into training and validation sets at a ratio of 9:1. To control the disentanglement property in the latent representation, we adopt the strategy proposed by [37] and set the expected KL values to be 20. We train the VAE model for 130 epochs and choose the best checkpoint for later inference. The experiments are run five times, and the average scores are reported. For inference, we perform gradient ascent as the warm-up phase for $T = 500$ iterations with the learning rate $\alpha \in [0.001, 0.01]$. The latent-based directed evolution involves $G = 10$ iterative processes, with the candidate number set to $K \times B = 128$. In our implementation, we set the number of samples and beam size to $K = 128$ and $B = 1$, respectively. Finally, for the random exploration, we try multiple combinations of annealing factor $\delta = 0.1$ and exploration step size $\gamma \in [1.5, 6]$ and report the best outcomes.

Baseline Algorithms We compare our method against the following representative baselines: (1) **AdaLead** [38] is an advanced implementation of model-guided evolution. (2) **DyNA PPO** [1] applies proximal policy optimization to search sequences on a learned landscape model. (3) **CbAS** [5] is a probabilistic modeling framework and uses an adaptive sampling algorithm. (4) **CMA-ES** [16] is a famous evolutionary search algorithm. (5) **COMs** [43] is conservative objective models for offline MBO. (6) **PEX** [33] is a model-guided sequence design algorithm using proximal exploration. (7)

GFN-AL [18] applies GFlowNet to design biological sequences. (8) **GGS** [23] is a graph-based smoothing method to optimize protein sequences. To ensure precise evaluation, we re-execute and re-evaluate all baseline methods using the same oracle. For the implementation from (1) to (4), we employ the open-source implementation provided by [38]. Regarding other baseline methods, we utilize the codes released by their respective authors. We were unable to evaluate [39] due to unrunnable public code.

Oracles To ensure unbiased evaluation and avoid circular use of oracles, we use two separate oracles for each fitness dataset: (1) the optimization oracle that guides the model optimization and (2) the evaluation oracle that assesses the performance of the methods. Firstly, we establish the optimization oracle $\mathcal{O}(\cdot)$, utilized for latent-based directed evolution, by leveraging features generated by the pre-trained 33-layer ESM-2 with a dimension of 1280. Subsequently, we fine-tune an Attention1D model to predict fitness values based on these representations. As for the evaluation oracle $\mathcal{E}(\cdot)$, which acts as the "ground-truth" evaluator, we employ the trained oracle provided by [33] to assess all methods, including our approach and other baselines.

Evaluation Metrics We use three metrics defined in [17] to evaluate our method and compare with other baselines: (1) **MFS**: maximum fitness score, (2) **Diversity**, (3) **Novelty**. The metrics are computed as follows:

- **MFS** = $\max(\{\mathcal{E}(p_i)\}_{i=1}^{N})$,
- **Diversity** = $\dfrac{\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} d(p_i,p_j)}{N(N-1)}$,
- **Novelty** = $\dfrac{\sum_{i=1}^{N}\min_{s_j\in\mathcal{D}} d(p_i,s_j)}{N}$,

where $d(\cdot,\cdot)$ is the Levenshtein distance, and $\mathcal{D}$ is the initial dataset (i.e., training dataset). It is crucial to emphasize that greater diversity and novelty do *not* equate to superior performance, but offer insights into the exploration and exploitation trade-offs exhibited by different methods.

E Additional Experimental Results

Table 5: Diversity across eight protein datasets. This table provides insight into the exploration and exploitation trade-off among methods.

Models	avGFP	AAV	TEM	E4B	AMIE	LGK	Pab1	UBE2I	Average
AdaLead	9.814	6.904	8.071	7.412	6.898	7.430	7.441	7.439	7.676
DyNA PPO	204.376	114.229	157.964	140.617	171.123	205.098	185.145	179.212	169.721
DbAS	205.717	115.426	159.524	142.044	172.539	206.819	186.761	180.758	171.199
CbAS	205.709	115.437	159.502	142.033	172.526	206.823	186.766	180.769	171.196
CMA-ES	173.365	96.882	134.633	119.388	145.151	174.274	157.098	152.058	144.106
COMs	70.319	45.761	73.191	68.398	100.731	126.623	115.242	109.650	88.739
PEX	7.048	4.782	6.692	6.625	6.388	7.031	7.012	7.219	6.600
GFN-AL	6.253	0.501	1.234	29.057	46.577	11.248	27.134	3.253	28.231
GGS	4.630	12.712	7.385	10.577	14.692	16.151	16.730	3.011	10.744
LDE (ours)	94.646	1.604	61.652	4.504	35.762	108.053	9.040	14.666	41.806

E.1 Differences between optimization oracle and evaluation oracle

In Table 7, we present a comparison of results for designs using the optimization oracle $\mathcal{O}(\cdot)$ (Opt. Oracle) and the evaluation oracle $\mathcal{E}(\cdot)$ (Eval. Oracle). The table demonstrates that, with the exception of Pab1, all other benchmarks experienced a decrease in performance when using the evaluation oracle. This decline is expected, as different oracle architectures result in different approximate fitness scores. Therefore, as we optimize the results based on the optimization oracle and solely utilize the evaluation oracle to assess the final performance of the method, the score produced by the optimization oracle should be higher.

Table 6: Novelty across eight datasets. This table provides insight into the exploration and exploitation trade-off among methods.

Models	avGFP	AAV	TEM	E4B	AMIE	LGK	Pab1	UBE2I	Average
AdaLead	13.486	17.805	41.405	48.934	41.167	78.868	73.526	78.548	47.967
DyNA PPO	201.702	111.566	156.784	139.227	170.368	205.815	185.686	179.801	168.869
DbAS	201.838	111.544	156.858	139.222	170.066	205.439	185.366	179.545	168.735
CbAS	201.825	111.549	156.845	139.172	170.031	205.404	185.354	179.549	168.716
CMA-ES	202.155	111.467	156.968	139.126	157.701	193.991	175.414	170.746	163.446
COMs	184.177	98.831	111.931	101.930	123.590	150.657	134.298	129.795	129.401
PEX	4.323	1.930	4.461	4.748	3.031	8.614	4.356	4.779	4.530
GFN-AL	220.632	2.452	255.939	29.062	326.255	413.951	64.046	145.458	192.728
GGS	3.552	2.061	4.029	8.081	9.989	20.990	8.380	2.862	7.493
LDE (ours)	91.863	0.657	62.508	2.037	10.423	65.145	2.875	126.495	45.250

Table 7: Comparison of optimization and evaluation oracles on the max fitness scores across eight protein benchmarks.

	avGFP	AAV	TEM	E4B	AMIE	LGK	Pab1	UBE2I
Opt. Oracle	15.266	2.736	5.986	5.594	0.327	0.939	0.786	7.405
Eval. Oracle	8.058	2.636	1.745	5.120	-0.103	0.018	1.548	4.297

E.2 Autoregressive vs. Non-autoregressive

In addition to the autoregressive decoder utilized in LDE, we report the performance of a non-autoregressive decoder implemented following the architecture proposed by [8]. This decoder comprises four 1-dimensional convolutional layers with ReLU activations and batch normalization layers are incorporated between convolutional layers, except for the final layer. As outlined in Table 8, it is observed that the autoregressive decoder consistently outperforms the non-autoregressive decoder across all tasks. This paves the way for further study on how different types of decoders affect the latent-based evolutionary algorithms.

Table 8: Maximum fitness scores on eight protein datasets of two decoder versions of LDE.

	avGFP	AAV	TEM	E4B	AMIE	LGK	Pab1	UBE2I	Average
Non-autoregressive LDE	3.733	1.368	1.095	3.123	-0.558	-0.005	0.089	2.976	1.430
(Autoregressive) LDE	**8.058**	**2.636**	**1.745**	**5.120**	**-0.103**	**0.018**	**1.548**	**4.297**	**3.204**

E.3 Active Learning

In particular, we perform an outer active learning loop with N rounds to iteratively update the latent space, as well as the simulated landscape produced by the encoder ϕ and the latent fitness predictor f as mentioned in Section 2.2. For each round i, we fine-tune the VAE model M with the dataset $\mathcal{D}_{i-1}$. The fine-tuned model is then used in Algorithm 1 to explore sequences with higher fitness scores. In our study, we avoid the circular use of oracles by using the optimization oracle $\mathcal{O}(\cdot)$ during the optimization process to guide the exploration, and the evaluation oracle $\mathcal{E}(\cdot)$ is used to evaluate of the final population. We remove all duplicated samples in $\mathcal{P}$ and use them as training samples $\mathcal{D}_i$ for the next round in our algorithm.

Algorithm 2 Active Learning with Latent-Based DE

```
Input: a VAE $M = (\phi, \theta, f)$, training dataset $\mathcal{D}_t$, number of rounds $N$,
       optimization oracle $\mathcal{O}$, number of epochs $n$.
1: Train $M$ on $\mathcal{D}_t$
2: $\mathcal{D}_0 \leftarrow \emptyset$
3: for $i = 1$ to $N$ do
4:     Run Algorithm 1 with oracle $\mathcal{O}$ and $M$ to
       find population $\mathcal{P} = \{(x, \mathcal{O}(y)) | x \in \mathcal{V}^L, y \in \mathbb{R}\}$.
5:     $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup$ RemoveDuplicate($\mathcal{P}$)
6:     Update $M$ on the data $\mathcal{D}_i$ in $n$ epochs
7: end for
Return $P_G$
```