# Structure-aware Reinforcement Learning for Protein Directed Evolution

**Zikun Nie**[1,2], **Suyuan Zhao**[1,2], **Yizhen Luo**[1,2], **Siqi Fan**[1], **Zaiqing Nie**[1,3,*]
[1]Institute of AI Industry Research (AIR), Tsinghua University
[2]Department of Computer Science and Technology, Tsinghua University
[3]PharMolix Inc.
{nzk24,zhaosy23,yz-luo22}@mails.tsinghua.edu.cn
zaiqing@air.tsinghua.edu.cn

## Abstract

Protein optimization remains a longstanding goal in life sciences. Existing machine learning–assisted directed evolution (MLDE) methods primarily rely on sequence-only features, overlooking the critical spatial constraints and co-evolutionary interactions encoded in protein structures. However, directly integrating structural information remains challenging due to the scarcity of reliable mutant structures. To address these issues, we propose **StructEvo**, a novel structure-aware reinforcement learning framework for protein directed evolution. StructEvo employs a delta-structure fusion encoder to approximate mutant structural features via feature differences, enabling effective end-to-end incorporation of spatial knowledge, without predicting coordinate-level structures for individual protein mutants. The vast mutation space is then decomposed into manageable subspaces through a structure-aligned hierarchical action network, while a geometric constraint further stabilizes delta feature learning. Our approach outperforms prior state-of-the-art methods by 9.2% and 16.3%, respectively, on two challenging optimization benchmarks, and further identifies an experimentally validated epistasis pattern in GFP, highlighting the importance of structural guidance for effective protein directed evolution.

## 1 Introduction

Proteins are fundamental functional units of biological systems. Compared to *de novo* protein design, optimizing existing proteins through mutations to improve desired functional properties (*i.e.*, *fitness*) offers greater practical value for real-world applications [1], ranging from antibody affinity maturation [2, 3] to enzyme catalytic engineering [4, 5]. In practice, the most widely adopted approach for protein optimization is directed evolution [6–8], which iteratively modifies proteins by random mutagenesis, expression, screening and selection. This process relies on large-scale *in vivo* expression and *in vitro* screening, making it costly and time-consuming [9].

Since directed evolution operates on amino acid sequences, existing machine learning-assisted directed evolution (MLDE) approaches [10–16] are predominantly sequence-oriented, focusing on modeling the rugged fitness landscape that maps protein sequences to fitness values. Early methods [14, 15] represent protein sequences by one-hot encodings or hand-crafted features, while recent approaches [13, 16] leverage protein language models (PLMs) to capture evolutionary priors. However, these sequence-only approaches often overlook the intrinsic spatial context in protein structures, limiting their ability to capture functionally relevant interactions. By explicitly encoding physical and geometric constraints, three-dimensional structures provide rich information about co-evolutionary interaction signals, spatial proximity, and functional motifs, which are essential for

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

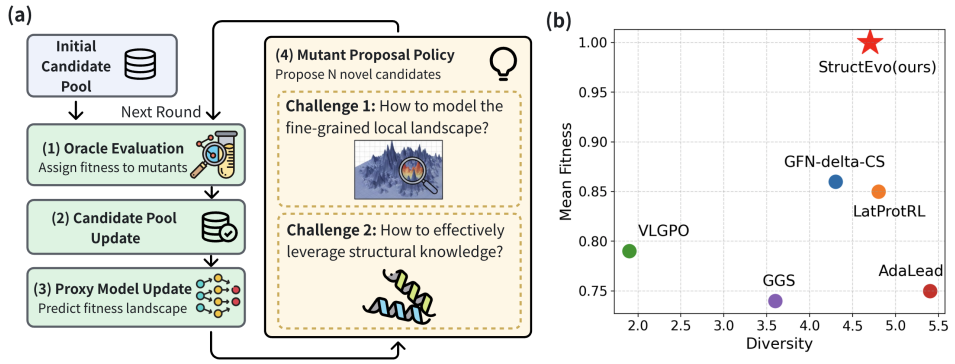


Figure 1: **(a) Illustration of the active learning pipeline for directed evolution.** Starting from an initial pool, candidates are evaluated by an oracle and then update the pool, after which a proxy is iteratively refined to guide candidate proposals. The figure highlights two key challenges for the proposal policy: how to model the fine-grained local fitness landscape, and how to effectively leverage structural knowledge. **(b) Performance comparison on GFP-hard benchmark.** StructEvo achieves the highest fitness while maintaining comparable diversity to other high-fitness baselines.

protein activity [17]. Motivated by this, we integrate protein structure as a complementary modality to sequence, facilitating more fine-grained geometric guidance for navigating local fitness landscapes.

However, leveraging structural knowledge presents two key challenges:

(1) *Reliable mutant structures are generally unavailable*. Though a representative experimentally resolved structure may be available for a protein family, mutant structures remain scarce in databases. Meanwhile, existing structure prediction models, such as the AlphaFold [18–21] and Protenix series [22–24], are computationally costly [25, 26] and often insensitive to point mutations, frequently producing nearly identical structures for similar sequences [27, 28]. Consequently, it remains impractical to directly capture structural variations at the coordinate level, especially in directed evolution settings where even a few selected mutations may induce noticeable backbone deviations. To address this, we model structural changes at the feature level by converting differences in PLM representations, which have been shown to contain rich mutational effects [29]. In this way, we transform the static structure into effective guidance during optimization.

(2) It remains unclear *how to effectively translate structural signals into mutation decisions*. Jointly optimizing mutation positions and substituted amino acids results in a large action space, making it difficult to learn reliable structural guidance. A key observation is that protein structure is more informative for *where to mutate* than for *mutate to what*, as it provides position-specific spatial context such as residue interactions in three-dimensional space. Motivated by this insight, we decompose mutation decisions into two sequential processes: position selection and amino acid refinement. This design enables structure to guide position decisions, while providing cleaner gradient signals in two low-dimensional subspaces.

Based on these insights, we propose **StructEvo**, a structure-aware reinforcement learning framework for protein directed evolution. StructEvo introduces a novel paradigm for incorporating structural knowledge into the mutation proposal policy, consisting of three components: (1) a *delta-structure fusion encoder* that approximates mutant structure features via delta representations and integrates sequence and structure information via a cross-attention module; (2) a *structure-aligned hierarchical action network* that decomposes action space and enables structure-guided mutation sampling; and (3) a *geometric constraint* applied on delta representations to improve consistency.

We validate our work through comprehensive experiments on two combinatorial mutation benchmarks and two more challenging full-length benchmarks. Across all benchmarks, StructEvo consistently outperforms strong baselines, and achieves relative fitness gains of up to 9.2% on AAV-hard and 16.3% on GFP-hard (Figure 1b), demonstrating its effectiveness for protein directed evolution. We further independently discovered a triple-mutation epistasis pattern in GFP validated by existing wet-lab experiments, highlighting the potential of our method for real-world applications. Our code and model are open-sourced at `https://github.com/Skyyyyyalker/StructEvo`.

Our contributions are summarized as follows:

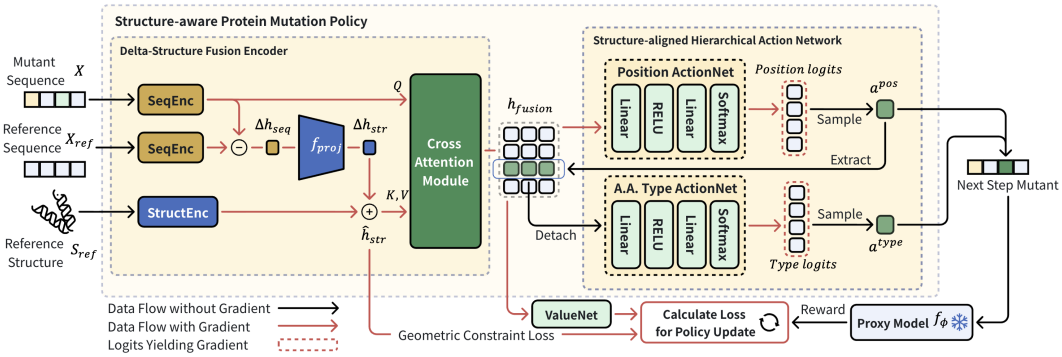


Figure 2: **Architecture of the StructEvo framework.** A delta-structure fusion encoder approximates mutant structure features via delta representations and integrates sequence and structure information. A structure-aligned hierarchical action network then decomposes action space and enables structure-guided mutation sampling. A geometric constraint loss is calculated to improve robustness. The proxy model remains frozen and provides reward signals during policy updates. A.A.: amino acid.

- We propose **StructEvo**, a structure-aware reinforcement learning framework that introduces a novel paradigm for incorporating structural knowledge into mutation proposal policy.
- We introduce a *delta-structure fusion encoder*, a *structure-aligned hierarchical action network* and a *geometric constraint* to enable effective structural guidance.
- We achieve relative fitness gains of 9.2% and 16.3% over the prior strongest baselines on two hard benchmarks, and identify a validated triple-mutation pattern in GFP, highlighting the importance of structural guidance for protein directed evolution.

## 2 Related work

**Machine learning–assisted directed evolution (MLDE).** MLDE aims to iteratively optimize a given starting protein sequence through mutations, where the key challenge is identifying beneficial variants in a rugged local fitness landscape under limited experimental budgets. *Evolutionary search-based methods* simulate natural selection through iterative mutation and recombination [11], with extensions that encourage proximal exploration via distance penalties [12]. *Energy-based methods* learn smooth fitness landscapes as energy models to guide sampling through gradient ascent [30–32]. *Generative model-based methods* explicitly model the mutant distributions using variational autoencoders [10, 33] or generative flow networks [34], with recent work further introducing controllable generation mechanisms [35, 36]. Existing MLDE methods primarily rely on sequence-only representations, limiting their ability to incorporate structural context during mutation selection.

**Reinforcement learning (RL) for protein optimization.** Recent studies increasingly formulate protein optimization as a sequential decision-making problem and leverage RL to guide mutation policies [37]. Wang et al. [38] generate new sequences through self-play and Monte Carlo tree search, while Wang et al. [16] draw physicochemical priors from an amino acid knowledge graph to guide mutation decisions. Lee et al. [13] further optimize mutants in a continuous latent space by treating mutations as small perturbations. Unlike prior methods, our approach introduces delta features to capture end-to-end structural variation within a hierarchical RL policy, enabling explicit guidance for proposing high-fitness mutants.

**Multi-modal protein fitness prediction**. This task is commonly referred to as *fitness landscape modeling*, focusing on predicting functional fitness values for given protein variants [39, 40]. Existing protein mutation prediction methods focus on learning representations and estimating fitness effects, but do not address the search problem of how to find better mutants. Beyond sequence representations, prior studies incorporate wild-type structure utilizing graph neural networks (GNNs) [41, 42] or inverse-folding models [43–45]. Recent studies further explore sequence-structure fusion through a joint vocabulary [46] or structure quantization module [47]. Additional modalities such as surface topology and multiple sequence alignment (MSA) are also leveraged [48, 49]. Compared to static fitness prediction, our work focuses on the more challenging task of proposing beneficial protein mutants in an active learning setting.

## 3 Task formulations

### 3.1 Directed evolution in an active learning setting

In both *in vitro* and *in silico* settings, directed evolution is often conducted in an *active learning* setting [38], where the mutants start from an initial pool and are then optimized iteratively over a total of $R$ rounds. Within each round, the optimization proceeds through four key components (Figure 1a):

**Oracle evaluation.** Let $\mathcal{V}$ denote the vocabulary of the 20 standard amino acids. A protein sequence of length $L$ is defined as $X = (x^1, ..., x^L) \in \mathbb{X} = \mathcal{V}^L$, where each residue $x^i \in \mathcal{V}$. A black-box *oracle* $\mathcal{F} : \mathbb{X} \to \mathbb{R}$ is a function that maps a protein sequence $X$ to a scalar fitness value $\mathcal{F}(X)$, thereby defining the fitness landscape. In MLDE, the oracle corresponds to costly wet-lab experiments for evaluating candidate sequences. Consequently, the number of oracle queries is strictly limited to $N$ per round, where $N \ll |\mathbb{X}|$.

**Candidate pool update.** A candidate pool $\mathcal{D}$ is iteratively updated to collect high-fitness mutants. The initial pool $\mathcal{D}_0 = \{(X_n, \mathcal{F}(X_n))\}_{n=1}^{|\mathcal{D}_0|}$ contains only a small set of low-fitness candidates. At the beginning of each round $i$, newly proposed mutants are evaluated and incorporated into the pool:

$$\mathcal{D}_i = \mathcal{D}_{i-1} \cup \{(X_n, \mathcal{F}(X_n))\}_{n=1}^{N}. \tag{1}$$

**Proxy model update.** Due to the limited number of oracle queries, a trainable *proxy* model $f_\phi(\cdot)$, parameterized by $\phi$, is used to approximate the landscape defined by the oracle. After updating the candidate pool, the proxy is iteratively fine-tuned on $\mathcal{D}_i$ by minimizing the mean squared error loss:

$$\mathcal{L}(\phi) = \frac{1}{|\mathcal{D}_i|} \sum_{X \in \mathcal{D}_i} \left(f_\phi(X) - \mathcal{F}(X)\right)^2. \tag{2}$$

**Proposal policy.** Guided by the fine-tuned proxy model $f_\phi$ and starting from the current candidate pool $\mathcal{D}_i$, a proposal policy generates a batch of $N$ novel mutants at each round. These mutants are then evaluated by the oracle and subsequently update the candidate pool for the next round.

The main objective of directed evolution is to *identify a batch of $K$ high-fitness protein mutants under a fixed budget of $R$ rounds*. A summary of all symbols is provided in Appendix A.1 for reference.

### 3.2 Reinforcement learning-based directed evolution

We formulate protein optimization as a reinforcement learning (RL) problem. By modeling sequential mutation trajectories as a Markov Decision Process (MDP), RL enables learning a policy that sequentially selects mutations to achieve high fitness.

**State.** At step $t$, the state $s_t = (x_t^1, ..., x_t^L)$ is a protein mutant with $L$ residues, where $x_t^i \in \mathcal{V}$.

**Action.** An action $a_t = (a_t^{\text{pos}}, a_t^{\text{type}})$ at step $t$ represents a single-point mutation applied to the current state $s_t$. The action $a_t$ can be factorized into two parts: the mutation position $a_t^{\text{pos}}$ and the type of substituted amino acid $a_t^{\text{type}}$. In our task, the transition function $P(s_{t+1} \mid s_t, a_t)$ is deterministic and the state is updated by:

$$x_{t+1}^i = \begin{cases} a_t^{\text{type}}, & i = a_t^{\text{pos}}, \\ x_t^i, & \text{otherwise.} \end{cases}$$

**Reward.** Reward signals are provided by the proxy model. In our work, we adopt a sparse reward setting, where the reward $r_t = r(s_t, a_t)$ is assigned only at trajectory termination. A trajectory terminates either when the policy finds a mutant better than the starting one, or when a predefined maximum number of steps $m_{\text{step}}$ is reached. Specifically, for a trajectory ending at step $T$, the reward is $r_T = f_\phi(s_{T+1})$, while intermediate rewards are set to $r_t = 0$ for $t < T$. We emphasize that rewards are generated solely by the proxy model, and the oracle is used only for final evaluation.

**Training Objective.** We aim to learn a policy $\pi(a_t \mid s_t)$ that maximizes the expected return over trajectories. Under the sparse reward setting, the training objective is formulated as:

$$\max_\pi \ \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{T} \gamma^t r_t \right] = \max_\pi \ \mathbb{E}_{\tau \sim \pi} \left[ \gamma^T f_\phi(s_{T+1}) \right], \tag{3}$$

where $\tau = (s_0, a_0, \ldots, s_T, a_T)$ denotes a mutation trajectory terminating at step $T$, and $\gamma$ is the discount factor. We optimize the policy using Proximal Policy Optimization (PPO) [50], which has demonstrated effectiveness in prior studies [13, 16]. The collected candidates are then ranked by proxy-predicted fitness and evaluated by the oracle for subsequent refinement in the next round.

Within the reinforcement learning framework, our work primarily focuses on improving the key component—the *mutation proposal policy*, as described in the following section.

# 4 Methodology

The main goal of our work is to incorporate structural knowledge into the protein mutation policy that maps states to actions. In this section, we present the proposed StructEvo framework, highlighting three main components: (1) a *delta-structure fusion encoder* that fuses sequential and structural features for a given mutant (Sec. 4.1), (2) a *structure-aligned hierarchical action network* that receives the fused feature and samples high-fitness mutations (Sec. 4.2), and (3) a *geometric constraint* between delta features to improve robustness and consistency (Sec. 4.3).

## 4.1 Delta-structure fusion encoder

The delta-structure fusion encoder integrates both sequence and structural knowledge to produce a fine-grained representation of protein mutants. As illustrated in Figure 2, it consists of a protein sequence encoder $\mathcal{E}_{\text{seq}}$, a protein structure encoder $\mathcal{E}_{\text{str}}$, a geometric projector $f_{\text{proj}}$ and a cross-attention module. We adopt ESM-2 [51], a protein language model (PLM) pretrained on large-scale evolutionary sequence data, to initialize the sequence encoder and capture mutant sequence features. For structure encoding, we employ ESM-IF [43] due to its simplicity and efficiency, as it captures both topology and geometric constraints of protein structures. Notably, the StructEvo framework is model-agnostic and easily extensible to different protein encoders.

**Formulation of delta sequence feature.** Since reliable mutant structures are generally unavailable, we adopt a *reference structure* $S^{\text{ref}}$ with the highest reliability as a structural scaffold for the corresponding mutant family (see Appendix D.1), with its sequence denoted as the *reference sequence* $X^{\text{ref}}$. Prior work [29] has shown that differences between PLM representations capture rich information of mutational effect. Therefore, we introduce the delta sequence feature $\Delta h_{\text{seq}}$ as:

$$\Delta h_{\text{seq}} = h_{\text{seq}} - h_{\text{seq}}^{\text{ref}} = \mathcal{E}_{\text{seq}}(X) - \mathcal{E}_{\text{seq}}(X^{\text{ref}}). \tag{4}$$

$X, X^{\text{ref}}$ denote the sequences of the current and the reference proteins, and $h_{\text{seq}}, h_{\text{seq}}^{\text{ref}}$ are the corresponding sequence representations, where $d_{\text{seq}}$ denotes its dimensionality.

**Projection into structural manifold.** We speculate that the mutational knowledge contained in the delta sequence feature reflects corresponding structural changes. Accordingly, we introduce a projector $f_{\text{proj}} : \mathbb{R}^{L \times d_{\text{seq}}} \to \mathbb{R}^{L \times d_{\text{str}}}$ to map the delta sequence feature from the sequence space to the structural manifold, yielding the delta structure feature $\Delta h_{\text{str}}$. Both the sequence and structural representations are aligned to residue-level embeddings of the protein backbone, enabling us to get the approximated mutant structural features $\hat{h}_{\text{str}}$ by adding the projected delta features to the reference structural features. These procedures are formulated as:

$$\Delta h_{\text{str}} = f_{\text{proj}}(\Delta h_{\text{seq}}), \quad \hat{h}_{\text{str}} = h_{\text{str}}^{\text{ref}} + \Delta h_{\text{str}} = \mathcal{E}_{\text{str}}(S^{\text{ref}}) + \Delta h_{\text{str}}. \tag{5}$$

Here, $S^{\text{ref}} \in \mathbb{R}^{L \times 3 \times 3}$ is the backbone coordinates of the reference structure, where each residue is represented by the three-dimensional positions of its N, C$\alpha$, and C atoms. To ensure sequence-structure consistency, we further impose a geometric constraint between delta features, as described below in Sec. 4.3.

**Structure-conditioned feature fusion.** To integrate sequence and structure information, we employ a cross-attention module where the mutant sequence feature $h_{\text{seq}}$ serves as the query, and the approximated structure feature $\hat{h}_{\text{str}}$ serves as the key and value. The asymmetric design allows sequence features to selectively query and attend to structural context. The resulting fusion feature $h_{\text{fusion}}$ encodes both sequential and structural information of the mutant, serving as input to the downstream hierarchical action network for mutation sampling.

### 4.2 Structure-aligned hierarchical action network

In RL formulation, the Markov decision process decomposes the vast mutation space into a sequence of single-step decision problems. Each action corresponds to a single-point mutation $a = (a^{\text{pos}}, a^{\text{type}})$, where $a^{\text{pos}} \in \{1, ..., L\}$ denotes the mutation position and $a^{\text{type}} \in \mathcal{V}$ denotes the substituted amino acid type. Here, $L$ is the protein length and $\mathcal{V}$ is the standard amino acid vocabulary. For the four-site benchmarks, mutation positions are restricted to the four designated sites. As depicted in Figure 2, the decision process is decomposed into two hierarchical steps:

**Mutation position sampling.** Since mutation position is more tightly coupled with the structural context than amino acid type, prioritizing position selection $a^{\text{pos}}$ enables the policy to better exploit structure-derived insights. A *position action network* $\pi_{\theta_1}$ receives the fusion feature $h_{\text{fusion}}$ as input and outputs a categorical distribution over mutation position subspace, from which the position action $a^{\text{pos}} \sim \pi_{\theta_1}(\cdot \mid h_{\text{fusion}})$ is sampled.

**Substituted amino acid sampling.** Once the position $a^{\text{pos}}$ is determined, the corresponding local representation at the specific position is then *extracted* and *detached* from the fusion feature. An *amino acid type action network* $\pi_{\theta_2}$ then operates on the detached context to sample the substituted amino acid $a^{\text{type}} \sim \pi_{\theta_2}(\cdot \mid h_{\text{fusion}}, a^{\text{pos}})$. Here, the *detach* operation blocks gradients backpropagating into the shared fusion encoder and the position network. This prevents gradients from the amino acid type network from directly updating the shared fusion representation, allowing the position network to receive a cleaner and more focused learning signal (see Appendix A.4).

### 4.3 Geometric constraint for delta features

**Geometric constraint.** In the delta-structure fusion encoder, the delta sequence feature is projected to obtain a corresponding delta structure feature. We aim to maintain consistency between structural variations and sequence perturbations, based on the intuition that small mutational effects captured by PLM should not lead to excessively large changes in structural features, and vice versa. To encourage this, we introduce a per-residue geometric constraint between the two delta features, formulated as:

$$\mathcal{L}_{\text{geo}} = \mathbb{E}_{\Delta h_{\text{str}}, \Delta h_{\text{seq}}} \left[ \frac{1}{L} \sum_{i=1}^{L} \left( \frac{\|\Delta h^i_{\text{str}}\|}{\sqrt{d_{\text{str}}}} - c * \frac{\|\Delta h^i_{\text{seq}}\|}{\sqrt{d_{\text{seq}}}} \right)^2 \right], \tag{6}$$

where $c$ is the scaling factor, and $\| \cdot \|$ denotes the $\ell_2$-norm. From another perspective, the geometric constraint is inspired by the notion of *Lipschitz continuity* [52], as it encourages the magnitude of structural feature variations to remain proportional to that of sequence perturbations, thus enabling a more stable and faithful mapping (see Appendix A.5).

**Overall training objective.** The overall RL training objective consists of two parts:

$$\mathcal{L}_{\text{total}} = \underbrace{\mathcal{L}_{\text{policy}} + \lambda_{\text{entropy}} \cdot \mathcal{L}_{\text{entropy}} + \lambda_{\text{value}} \cdot \mathcal{L}_{\text{value}}}_{\text{standard PPO loss}} + \underbrace{\lambda_{\text{geo}} \cdot \mathcal{L}_{\text{geo}}}_{\text{geometric constraint}}. \tag{7}$$

This objective jointly optimizes the mutation policy while enforcing consistency between delta sequence and structure features. The policy loss $\mathcal{L}_{\text{policy}}$, entropy loss $\mathcal{L}_{\text{entropy}}$, and value loss $\mathcal{L}_{\text{value}}$ are defined in the same way as in the standard PPO loss function [50]; see Appendix A.3 for details. Algorithm 1 summarizes the overall mutation proposal pipeline.

## 5 Experiments

In this section, we demonstrate that StructEvo is effective on directed evolution tasks across two combinatorial benchmarks (Sec. 5.1) and two more challenging full-length mutation benchmarks (Sec. 5.2). We further provide an in-depth analysis of our model (Sec. 5.3) and a case study that reveals high-fitness mutation patterns in green fluorescent protein (Sec. 5.4).

### 5.1 Optimization on combinatorial benchmarks

**Datasets.** We first evaluate StructEvo on two widely recognized four-site combinatorial datasets, GB1 and PhoQ. The GB1 dataset [53] consists of 149,361 variants at four epistatic sites (V39, D40,

Table 1: **Results on GB1 and PhoQ benchmarks**. We report average results and std across 5 independent runs. *: results are derived from Wang et al. [16]. n.a.: not reported in the original paper.

| | GB1 | | | PhoQ | | |
|---|---|---|---|---|---|---|
| Method | Max↑ | Mean↑ | NDCG↑ | Max↑ | Mean↑ | NDCG↑ |
| CMA-ES | 0.59±0.17 | 0.13±0.04 | 0.75±0.02 | 0.29±0.14 | 0.05±0.01 | 0.69±0.02 |
| BO | 0.67±0.10 | 0.17±0.02 | 0.77±0.01 | 0.29±0.06 | 0.05±0.01 | 0.70±0.01 |
| AdaLead | 0.64±0.13 | 0.30±0.09 | 0.74±0.02 | 0.27±0.11 | 0.14±0.02 | 0.67±0.02 |
| PEX | 0.66±0.03 | 0.36±0.02 | 0.76±0.01 | 0.40±0.03 | 0.10±0.02 | 0.71±0.02 |
| MLDE* | 0.68±n.a. | 0.20±n.a. | 0.79±n.a. | 0.36±n.a. | 0.10±n.a. | 0.79±n.a. |
| ftMLDE (EVmutation)* | 0.94±n.a. | 0.41±n.a. | 0.83±n.a. | 0.44±n.a. | 0.11±n.a. | 0.80±n.a. |
| ftMLDE (Transformer)* | 0.93±n.a. | 0.42±n.a. | 0.81±n.a. | 0.42±n.a. | 0.11±n.a. | 0.82±n.a. |
| EvoPlay* | 0.84±n.a. | 0.46±n.a. | 0.85±n.a. | 0.47±n.a. | 0.14±n.a. | 0.80±n.a. |
| CLADE* | 0.84±n.a. | 0.46±n.a. | 0.86±n.a. | 0.47±n.a. | 0.09±n.a. | 0.78±n.a. |
| CLADE2.0* | 0.94±n.a. | 0.49±n.a. | 0.88±n.a. | 0.40±n.a. | 0.15±n.a. | 0.81±n.a. |
| KnowRLM* | 0.97±0.06 | 0.56±0.02 | 0.88±0.02 | **0.66±n.a.** | 0.16±n.a. | 0.82±n.a. |
| **StructEvo (ours)** | **0.99±0.01** | **0.59±0.01** | **0.93±0.01** | 0.64±0.07 | **0.19±0.02** | **0.86±0.01** |

G41 and V54) of the GB1 domain of streptococcal protein G, with fitness reflecting binding to the Fc region of human IgG. Only 2.4% variants have higher fitness than the wild-type protein, highlighting the sparsity of the GB1 fitness landscape. The PhoQ dataset [54] contains 140,517 variants of the PhoQ histidine kinase mutated at four key interfacial residues (A284, V285, S288 and T289), with fitness reflecting both kinase and phosphatase responsiveness of PhoP/PhoQ regulatory system on signal transduction. Compared to GB1, the PhoQ fitness landscape is even sparser—98.7% mutants with fitness below 0.1 and 63.7% with zero, making it a particularly challenging task. All fitness values are min-max normalized to [0, 1]. Details of benchmarks are provided in Appendix C.1.

**Baselines.** We compare our method with various strong baselines, including (1) probability-based methods, BO [55] and CMA-ES [56]; (2) evolutionary search-based methods, AdaLead [11], PEX [12], MLDE [53] and two variants of ftMLDE [57]; (3) cluster-based methods, CLADE [14] and its advanced version CLADE2.0 [15], and (4) recent RL-based methods, EvoPlay [38] and KnowRLM [16]. Implementation details can be found in Appendix C.3.

**Metrics.** We report the *maximum* and *mean* fitness over a combined set of candidates, including $N$ initial mutants, $N * R$ proposed mutants across $R$ rounds, and the top $N$ mutants ranked by the final proxy. Here, $R$ denotes the total optimization rounds and $N$ is the number of proposed candidates per round. Given that the ground truth fitness is fully available in combinatorial tasks, we further evaluate the ranking quality of proxy by normalized discounted cumulative gain (NDCG). Formal definitions of metrics are provided in Appendix C.5.

**Experimental settings.** Following prior work [16], we set $R = 3$ and $N = 96$. The starting $N$ mutants are sampled by CLADE [14]. The oracle is the ground-truth dataset, with rare missing instances assigned a value of 0. All experiments are repeated over 5 independent runs.

**Results and analysis.** We present performance comparisons at round 3 in Table 1, with more detailed results of each round in Appendix E.1. We observe that: (1) StructEvo achieves state-of-the-art performance on GB1, outperforming the strongest RL-based baseline (KnowRLM) by relative gains of 5.7% on NDCG, demonstrating its strong capability in navigating fine-grained fitness landscapes and guiding mutant proposals. (2) Though the reference structure of PhoQ is predicted by AlphaFold2 [18] (with an average pLDDT of 82.94, see Appendix D.1) rather than experimentally resolved, StructEvo still remains competitive with KnowRLM on PhoQ benchmark, with relative gains of 4.9% on NDCG and 18.8% on mean fitness. This observation shows the robustness of our framework, while also indicating the importance of accurate structural information for landscape exploration. (3) Probabilistic and evolutionary methods perform poorly under extremely scarce data. The key strength of PEX, proximal preference, becomes less significant in the constrained four-site space. Cluster-based methods leverage prior knowledge to reduce aimless exploration, while RL-based methods further improve performance by explicitly modeling the sequential decision-making process. Hence, we emphasize the significance of combining structural priors for effective protein optimization in data-scarce settings.

Table 2: **Performance on AAV and GFP benchmarks**. The results include average and standard deviation across 5 independent runs. *: results are derived from Kirjner et al. [30]. †: results are derived from Lee et al. [13]. -: not reported in the original paper.

| | *AAV medium* | | | | *AAV hard* | | | |
|---|---|---|---|---|---|---|---|---|
| Method | Mean↑ | Max↑ | Diversity | Novelty | Mean↑ | Max↑ | Diversity | Novelty |
| CMA-ES | 0.05±0.00 | 0.40±0.01 | 20.9±0.5 | 17.2±0.5 | 0.05±0.00 | 0.32±0.03 | 20.4±0.7 | 18.6±0.5 |
| BO | 0.64±0.03 | 0.72±0.04 | 8.1±0.2 | 9.4±0.9 | 0.62±0.03 | 0.70±0.03 | 8.6±0.2 | 10.3±0.3 |
| PEX | 0.65±0.01 | 0.74±0.00 | 5.5±0.5 | 5.3±0.3 | 0.63±0.02 | 0.75±0.01 | 5.7±0.5 | 6.2±0.4 |
| GGS* | 0.51±0.01 | - | 4.0±0.2 | 5.4±0.5 | 0.60±0.02 | - | 4.5±0.5 | 7.0±0.0 |
| AdaLead | 0.74±0.04 | 0.80±0.05 | 5.2±0.9 | 7.6±0.4 | 0.76±0.03 | 0.81±0.03 | 4.0±0.4 | 8.9±0.6 |
| VLGPO | 0.69±0.01 | 0.76±0.01 | 3.1±0.2 | 5.7±0.1 | 0.71±0.01 | 0.78±0.01 | 3.3±0.2 | 6.9±0.1 |
| LatProtRL† | 0.71±0.02 | - | 5.4±0.8 | 6.0±0.2 | 0.66±0.01 | - | 6.0±0.8 | 7.0±0.6 |
| GFN-$\delta$-CS | 0.77±0.02 | 0.81±0.02 | 4.4±1.1 | 5.8±1.5 | 0.76±0.03 | 0.80±0.03 | 4.0±0.7 | 6.7±1.1 |
| **StructEvo** | **0.82±0.01** | **0.86±0.01** | 4.5±0.7 | 8.6±0.2 | **0.83±0.03** | **0.87±0.04** | 4.5±0.3 | 8.4±0.5 |
| | *GFP medium* | | | | *GFP hard* | | | |
| Method | Mean↑ | Max↑ | Diversity | Novelty | Mean↑ | Max↑ | Diversity | Novelty |
| CMA-ES | -0.04±0.01 | 0.58±0.00 | 172.7±2.1 | 154.3±4.9 | -0.03±0.00 | 0.16±0.03 | 173.8±2.0 | 218.6±7.6 |
| BO | 0.30±0.07 | 0.42±0.07 | 26.4±5.6 | 37.7±12.1 | 0.26±0.07 | 0.39±0.04 | 36.0±2.7 | 71.7±10.1 |
| PEX | 0.64±0.04 | 0.84±0.04 | 8.3±2.6 | 6.8±1.9 | 0.46±0.05 | 0.62±0.05 | 7.0±1.2 | 10.3±1.7 |
| GGS* | 0.76±0.01 | - | 3.7±0.2 | 5.0±0.1 | 0.74±0.00 | - | 3.6±0.0 | 6.9±0.1 |
| AdaLead | 0.93±0.01 | 0.99±0.02 | 5.2±0.9 | 9.5±1.2 | 0.75±0.04 | 0.81±0.04 | 5.4±0.6 | 13.5±1.1 |
| VLGPO | 0.93±0.03 | 1.02±0.01 | 2.8±0.2 | 5.3±0.0 | 0.79±0.02 | 0.95±0.00 | 1.9±0.0 | 6.2±0.0 |
| LatProtRL† | 0.93±0.00 | - | 4.6±0.4 | 5.5±0.1 | 0.85±0.01 | - | 4.8±0.3 | 7.0±0.4 |
| GFN-$\delta$-CS | 1.06±0.04 | 1.09±0.03 | 7.0±0.4 | 9.5±1.9 | 0.86±0.06 | 0.90±0.05 | 4.3±1.6 | 8.2±1.5 |
| **StructEvo** | **1.11±0.02** | **1.17±0.02** | 6.3±0.8 | 11.9±0.8 | **1.00±0.04** | **1.05±0.05** | 4.9±0.3 | 11.4±0.6 |

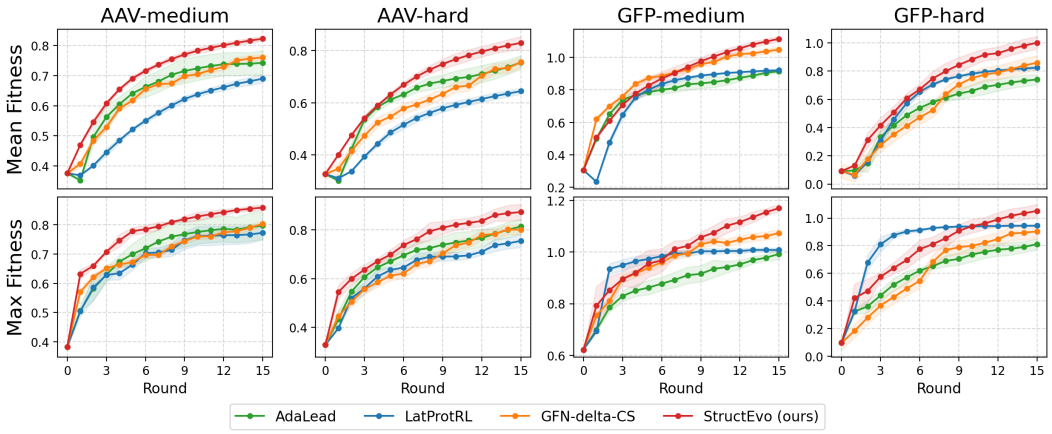


Figure 3: **Visualization of mean (top row) and maximum (bottom row) fitness across rounds.** The curves indicate the average results, and the shaded regions indicate the standard deviation.

### 5.2 Optimization on full-length benchmarks

**Datasets.** We further evaluate StructEvo on two full-length protein mutation benchmarks, *GFP* and *AAV*. Compared to combinatorial tasks, these are more challenging as mutations span the full sequence, leading to an exponentially larger search space. The GFP dataset [58] measures log-fluorescence intensity of 56,806 *Aequorea victoria* green fluorescent protein variants, with a mutation space size of $|\mathbb{X}| = 20^{237}$. The AAV dataset [59] focuses on engineering a 28-amino acid segment of the VP1 capsid protein of adeno-associated virus, comprising 44,156 variants from a space of $|\mathbb{X}| = 20^{28}$. Following Kirjner et al. [30], we partition both benchmarks into *medium* and *hard* regimes. The initial fitness distribution under the *hard* setting is more distant from the high-fitness range than the *medium* setting, leading to increased optimization difficulty. See more details in Appendix C.2.

**Baselines.** In addition to the baselines introduced for combinatorial benchmarks, we further compare StructEvo with (1) the energy model-based methods, GGS [30]; (2) latent-space optimization methods, LatProtRL [13] and VLGPO [35]; and (3) the GFlowNet-based method, GFN-$\delta$-CS [36]. Implementation details are provided in Appendix C.4. Since MLDEs [53, 57] and CLADEs [14, 15] need to enumerate all possible mutants to construct a global scoring vector, they may not be suitable

Table 3: **Results of ablation studies.** $X, X^{\text{ref}}$: mutant and reference sequence, $S^{\text{ref}}$: reference structure. w/o: without.

| | Settings | *Inputs* | | | Mean↑ | Max↑ | Time(s)↓ |
|---|---|---|---|---|---|---|---|
| | | $X$ | $X^{\text{ref}}$ | $S^{\text{ref}}$ | | | |
| *AAV hard* | StructEvo | ✓ | ✓ | ✓ | **0.83** | **0.87** | 0.337 |
| | (1) | ✓ | | ✓ | 0.80 | 0.84 | 0.323 |
| | (2) | ✓ | ✓ | | 0.75 | 0.79 | 0.297 |
| | (3) | ✓ | | | 0.75 | 0.81 | 0.303 |
| | (4) | w/o hierarchy | | | 0.77 | 0.81 | 0.583 |
| *GFP hard* | StructEvo | ✓ | ✓ | ✓ | **1.00** | **1.05** | 0.825 |
| | (1) | ✓ | | ✓ | 0.97 | 1.01 | 0.819 |
| | (2) | ✓ | ✓ | | 0.83 | 0.89 | 0.753 |
| | (3) | ✓ | | | 0.84 | 0.93 | 0.697 |
| | (4) | w/o hierarchy | | | 0.75 | 0.82 | 3.515 |

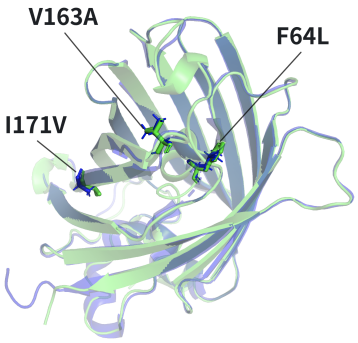


Figure 4: **A triple-mutation mutant in GFP.** The reference (green) and the mutant (blue) structure are rendered by PyMOL.

for full-length benchmarks, where the exponentially large mutation space size ($20^L$) makes such enumeration infeasible.

**Metrics.** We report the *mean* and *maximum* fitness of the top $N$ sequences proposed over $R$ rounds. Fitness values are evaluated by the oracle and min-max normalized using the ground truth dataset. Notably, normalized fitness may exceed 1.0, as the oracle can assign higher fitness to previously unseen mutants than the maximum observed in the ground-truth dataset. We additionally choose distance-based metrics, *diversity* and *novelty*, as described in Appendix C.6.

**Experimental settings.** Following prior work [30], we set the total rounds to $R = 15$, the number of candidates proposed per round, i.e., the evaluation budget, to $N = 128$. The oracle models are derived from Kirjner et al. [30]. Each method is run with 5 independent seeds.

**Results and analysis.** Table 2 and Figure 3 show comparisons between StructEvo and baselines on full-length benchmarks. We observe that: (1) StructEvo consistently achieves state-of-the-art fitness across all four scenarios, surpassing the strongest baseline by relative gains of 9.2% and 16.3% in mean fitness on the AAV and GFP *hard* settings, respectively. Furthermore, the larger improvements on *hard* settings compared to *medium* ones highlight the increasing importance of structural guidance when high-fitness data is severely limited. (2) While LatProtRL converges more quickly during the early rounds on GFP, StructEvo continues to improve across rounds and ultimately surpasses it. This indicates that structural information could provide sustained guidance for exploring more fine-grained high-fitness regions in the long-horizon optimization. Improving early-round efficiency remains an important direction for future work. (3) We note that higher diversity and novelty *do not* simply imply better performance, as pointed out in previous work [13]. For instance, CMA-ES attains the highest diversity and novelty but yields dysfunctional mutants with near-zero fitness. In contrast, StructEvo achieves the highest fitness while maintaining comparable or higher diversity and novelty than other high-fitness methods, reflecting a more effective trade-off between exploration and exploitation.

### 5.3 Ablation studies

We examine the impact of each component in StructEvo by: (1) removing the delta module by directly fusing mutant sequence with the static reference structure via cross-attention, (2) removing structural inputs and fusing two sequence features, (3) relying solely on mutant sequence, and (4) replacing the hierarchical action network with a flat network that jointly samples both actions. We also report the runtime cost per candidate for each setting. As shown in Table 3 and Appendix E.2, removing delta module degrades performance, while removing structural information leads to a larger drop (9.6% mean fitness on AAV-hard and 17.0% on GFP-hard). Substituting structure with reference sequence yields no improvements. Moreover, sampling in the joint space significantly degrades fitness and incurs much longer runtime, with both drawbacks amplified as protein length increases. These findings validate each component in our framework. We refer readers to more case studies and additional in-depth analyses of structure prediction models, model contribution isolation, proxy robustness, and geometric constraint sensitivity in Appendix F.

### 5.4 Case study: a triple-mutation epistasis in GFP

In GFP, we identify a triple-mutation pattern F64L/V163A/I171V (PDB: 8DPD; Figure 4). From 3,840 proposed mutants over 15 rounds, 271 mutants with this triple mutation achieve substantially higher fitness (average 0.91) than 1,834 single (0.07) and 427 double mutants (0.23), revealing a pronounced *epistasis effect*. Interestingly, this pattern has been validated and explained by wet-lab experiments [60, 61]: residue 64 is near the chromophore, while 163 and 171 are located in the proximal region of $\beta$-barrel; all mutations result in smaller side-chains, potentially facilitating tighter packing through synergistic effects. This promotes energy dissipation via photon emission and ultimately boosts fluorescence intensity. We further find a higher cosine similarity of this true mutant structure feature to the approximated feature (0.85) than to the reference (0.73), supporting the validity of delta module. Notably, none of the baselines recover this pattern, highlighting the advantage of structure in capturing long-range co-evolutionary signals that are typically missed by sequence-only approaches.

## 6 Conclusions

In this work, we propose StructEvo, a structure-aware reinforcement learning framework for protein directed evolution. By approximating mutant structure features via delta representations and decomposing the action space via a hierarchical action network, StructEvo introduces a new paradigm for incorporating structural knowledge into mutation proposal. Comprehensive experiments on both combinatorial and full-length benchmarks show consistent performance improvements over strong baselines, demonstrating the value of dynamic structural context for efficient protein optimization. Several limitations remain: (1) extending to protein complexes rather than monomers; (2) expanding single-objective fitness to multi-objective properties; and (3) integrating feedback from wet-lab experiments, which we leave for future work. While StructEvo shows potential in real-world applications, we emphasize safety concerns that *any protein variants generated by our method should undergo rigorous experimental validation and careful human inspection before practical use.*

## Acknowledgments

This research is supported by the Innovative Drug Research and Development National Science and Technology Major Project (No.2025ZD1803101), the Wuxi Research Institute of Applied Technologies, Tsinghua University (Grant 20242001120), and PharMolix Inc.

## References

[1] Misha Soskine and Dan S Tawfik. Mutational effects and the evolution of new protein functions. *Nature Reviews Genetics*, 11(8):572–582, 2010.

[2] Hans-Peter Roost, Martin F Bachmann, Andreas Haag, Ulrich Kalinke, Vladimir Pliska, Hans Hengartner, and Rolf M Zinkernagel. Early high-affinity neutralizing anti-viral igg responses without further overall improvements of affinity. *Proceedings of the National Academy of Sciences*, 92(5):1257–1261, 1995.

[3] Venkatasamy Manivel, Naresh C Sahoo, Dinakar M Salunke, and Kanury VS Rao. Maturation of an antibody response is governed by modulations in flexibility of the antigen-combining site. *Immunity*, 13(5):611–620, 2000.

[4] Dave W Anderson, Florian Baier, Gloria Yang, and Nobuhiko Tokuriki. The adaptive landscape of a metallo-enzyme is shaped by environment-dependent epistasis. *Nature Communications*, 12(1):3867, 2021.

[5] Richard J Fox and Gjalt W Huisman. Enzyme optimization: moving from blind evolution to statistical exploration of sequence–function space. *Trends in biotechnology*, 26(3):132–138, 2008.

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

[7] Michael S Packer and David R Liu. Methods for the directed evolution of proteins. *Nature Reviews Genetics*, 16(7):379–394, 2015.

[8] 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, 2021.

[9] Ling Yuan, Itzhak Kurek, James English, and Robert Keenan. Laboratory-directed protein evolution. *Microbiology and Molecular Biology Reviews*, 69(3):373–392, 2005.

[10] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pages 773–782. PMLR, 2019.

[11] 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. *arXiv preprint arXiv:2010.02141*, 2020.

[12] Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In *International Conference on Machine Learning*, pages 18520–18536. PMLR, 2022.

[13] Minji Lee, Luiz Felipe Vecchietti, Hyunkyu Jung, Hyun Joo Ro, Meeyoung Cha, and Ho Min Kim. Robust optimization in protein fitness landscapes using reinforcement learning in latent space. In *International Conference on Machine Learning*, pages 26976–26990. PMLR, 2024.

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

[15] 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, 2022.

[16] Yuhao Wang, Qiang Zhang, Ming Qin, Xiang Zhuang, Xiaotong Li, Zhichen Gong, Zeyuan Wang, Yu Zhao, Jianhua Yao, Keyan Ding, et al. Knowledge-aware reinforced language models for protein directed evolution. In *Forty-first International Conference on Machine Learning*, 2024.

[17] Hedi Hegyi and Mark Gerstein. The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *Journal of molecular biology*, 288 (1):147–164, 1999.

[18] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[19] Richard Evans, Michael O'neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *biorxiv*, pages 2021–10, 2021.

[20] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[21] Jun Cheng, Guido Novati, Joshua Pan, Clare Bycroft, Akvilė Žemgulytė, Taylor Applebaum, Alexander Pritzel, Lai Hong Wong, Michal Zielinski, Tobias Sargeant, et al. Accurate proteome-wide missense variant effect prediction with alphamissense. *Science*, 381(6664):eadg7492, 2023.

[22] ByteDance AML AI4Science Team, Xinshi Chen, Yuxuan Zhang, Chan Lu, Wenzhi Ma, Jiaqi Guan, Chengyue Gong, Jincai Yang, Hanyu Zhang, Ke Zhang, et al. Protenix-advancing structure prediction through a comprehensive alphafold3 reproduction. *BioRxiv*, pages 2025–01, 2025.

[23] Yuxuan Zhang, Chengyue Gong, Hanyu Zhang, Wenzhi Ma, Zhenyu Liu, Xinshi Chen, Jiaqi Guan, Lan Wang, Yanping Yang, Yu Xia, and Wenzhi Xiao. Protenix-v1: Toward high-accuracy open-source biomolecular structure prediction. *bioRxiv*, 2026. doi: 10.64898/2026.02.05.703733. URL `https://www.biorxiv.org/content/early/2026/02/22/2026.02.05.703733.1`.

[24] Yuxuan Zhang, Chengyue Gong, Jinyuan Sun, Jiaqi Guan, Milong Ren, Song Xue, Hanyu Zhang, Wenzhi Ma, Zhenyu Liu, Xinshi Chen, and Wenzhi Xiao. Protenix-v2: Broadening the reach of structure prediction and biomolecular design. *bioRxiv*, 2026. doi: 10.64898/2026.04.10.717613. URL `https://www.biorxiv.org/content/early/2026/04/11/2026.04.10.717613`.

[25] Jinpyo Kim, Mingi Kwon, and Jishen Zhao. Alphafold3 workload characterization: A comprehensive analysis of bottlenecks and performance scaling. In *2025 IEEE International Symposium on Workload Characterization (IISWC)*, pages 478–488. IEEE, 2025.

[26] Feiwen Zhu, Arkadiusz Nowaczynski, Rundong Li, Jie Xin, Yifei Song, Michal Marcinkiewicz, Sukru Burc Eryilmaz, Jun Yang, and Michael Andersch. Scalefold: Reducing alphafold initial training time to 10 hours. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.

[27] Marina A Pak, Karina A Markhieva, Mariia S Novikova, Dmitry S Petrov, Ilya S Vorobyev, Ekaterina S Maksimova, Fyodor A Kondrashov, and Dmitry N Ivankov. Using alphafold to predict the impact of single mutations on protein stability and function. *Plos one*, 18(3): e0282689, 2023.

[28] Federica Luppino, Swantje Lenz, Chi Fung Willis Chow, and Agnes Toth-Petroczy. Deep learning tools predict variants in disordered regions with lower sensitivity. *BMC genomics*, 26 (1):367, 2025.

[29] Yizhen Luo, Zikun Nie, Massimo Hong, Suyuan Zhao, Hao Zhou, and Zaiqing Nie. Mutaplm: Protein language modeling for mutation explanation and engineering. *Advances in Neural Information Processing Systems*, 37:79783–79818, 2024.

[30] 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 *ICLR*, 2024.

[31] Nathan C Frey, Daniel Berenberg, Karina Zadorozhny, Joseph Kleinhenz, Julien Lafrance-Vanasse, Isidro Hötzel, Yan Wu, Stephen Ra, Richard Bonneau, Kyunghyun Cho, et al. Protein discovery with discrete walk-jump sampling. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

[32] Thanh VT Tran, Nhat Khang Ngo, Viet Thanh Duy Nguyen, and Truong Son Hy. Latentde: Latent-based directed evolution accelerated by gradient ascent for protein sequence design. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*, 2024.

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

[34] 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.

[35] Lea Bogensperger, Dominik Narnhofer, Ahmed Allam, Konrad Schindler, and Michael Krauthammer. A variational perspective on generative protein fitness optimization. In *Forty-second International Conference on Machine Learning*, 2025.

[36] Hyeonah Kim, Minsu Kim, Taeyoung Yun, Sanghyeok Choi, Emmanuel Bengio, Alex Hernández-García, and Jinkyoo Park. Improved off-policy reinforcement learning in biological sequence design. In *Forty-second International Conference on Machine Learning*, 2025.

[37] 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*, 2019.

[38] Yi Wang, Hui Tang, Lichao Huang, Lulu Pan, Lixiang Yang, Huanming Yang, Feng Mu, and Meng Yang. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, 5(8):845–860, 2023.

[39] Christian Dallago, Jody Mou, Kadina E Johnston, Bruce J Wittmann, Nicholas Bhattacharya, Samuel Goldman, Ali Madani, and Kevin K Yang. Flip: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv*, pages 2021–11, 2021.

[40] Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood Van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, et al. Proteingym: Large-scale benchmarks for protein fitness prediction and design. *Advances in neural information processing systems*, 36:64331–64379, 2023.

[41] Pedro Hermosilla and Timo Ropinski. Contrastive representation learning for 3d protein structures. *arXiv preprint arXiv:2205.15675*, 2022.

[42] Can Chen, Jingbo Zhou, Fan Wang, Xue Liu, and Dejing Dou. Structure-aware protein self-supervised learning. *Bioinformatics*, 39(4):btad189, 2023.

[43] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pages 8946–8970. PMLR, 2022.

[44] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[45] Kevin K Yang, Niccolò Zanichelli, and Hugh Yeh. Masked inverse folding with sequence transfer for protein representation learning. *Protein Engineering, Design and Selection*, 36: gzad015, 2023.

[46] Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. In *ICLR*, 2024.

[47] Mingchen Li, Yang Tan, Xinzhu Ma, Bozitao Zhong, Huiqun Yu, Ziyi Zhou, Wanli Ouyang, Bingxin Zhou, Pan Tan, and Liang Hong. Prosst: Protein language modeling with quantized structure and disentangled attention. *Advances in Neural Information Processing Systems*, 37: 35700–35726, 2024.

[48] Zuobai Zhang, Pascal Notin, Yining Huang, Aurélie Lozano, Vijil Chenthamarakshan, Debora Marks, Payel Das, and Jian Tang. Multi-scale representation learning for protein fitness prediction. *Advances in Neural Information Processing Systems*, 37:101456–101473, 2024.

[49] Yang Tan, Ruilin Wang, Banghao Wu, Liang Hong, and Bingxin Zhou. From high-throughput evaluation to wet-lab studies: advancing mutation effect prediction with a retrieval-enhanced model. *Bioinformatics*, 41(Supplement_1):i401–i409, 2025.

[50] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[51] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[52] Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International conference on machine learning*, pages 264–273. PMLR, 2018.

[53] Zachary Wu, SB Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019.

[54] Anna I Podgornaia and Michael T Laub. Pervasive degeneracy and epistasis in a protein-protein interface. *Science*, 347(6222):673–677, 2015.

[55] James T Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

[56] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[57] 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, 2021.

[58] 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, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533 (7603):397–401, 2016.

[59] 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, 2021.

[60] Simon Delagrave, Rachael E Hawtin, Christopher M Silva, Mary M Yang, and Douglas C Youvan. Red-shifted excitation mutants of the green fluorescent protein. *Bio/technology*, 13(2): 151–154, 1995.

[61] Jean-Denis Pédelacq, Stéphanie Cabantous, Timothy Tran, Thomas C Terwilliger, and Geoffrey S Waldo. Engineering and characterization of a superfolder green fluorescent protein. *Nature biotechnology*, 24(1):79–88, 2006.

[62] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International conference on machine learning*, pages 8844–8856. PMLR, 2021.

[63] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in neural information processing systems*, 34:29287–29303, 2021.

[64] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. Proteinbert: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8): 2102–2110, 2022.

[65] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. In *International Conference on Machine Learning*, pages 16990–17017. PMLR, 2022.

[66] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[67] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[68] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

[69] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, pages 2962–2970, 2015.

[70] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. *Advances in neural information processing systems*, 25, 2012.

[71] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[72] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research*, 22(268):1–8, 2021.

# Appendix

## A Details of StructEvo

### A.1 Symbol definitions

Table 4: Definition of symbols used in our work.

| Category | Symbol | Definition |
|---|---|---|
| *Protein Representation* | $L$ | the protein length |
| | $X$ | the protein sequence |
| | $S$ | the protein structure |
| | $\mathbb{X}$ | the mutation space |
| | $\mathcal{V}$ | the amino acid vocabulary |
| | $x_i$ | the amino acid identity of residue $i$ |
| | $a_i, b_i, c_i$ | the three-dimensional coordinates of residue $i$ |
| | $\mathcal{E}_{seq}, \mathcal{E}_{str}(\cdot)$ | the sequence and structure encoder |
| | $h_{\text{seq}}, h_{\text{str}}, h_{\text{fusion}}$ | the sequence feature, structure feature, and the fusion feature |
| | $\Delta h_{\text{seq}}, \Delta h_{\text{str}}$ | the delta sequence and structure feature |
| | $\hat{h}_{\text{str}}$ | the approximated mutant structure feature |
| | $d_{\text{seq}}, d_{\text{str}}$ | the hidden size of sequence and structure features |
| *Active Learning* | $R$ | the number of total rounds |
| | $N$ | the number of proposed candidates per round |
| | $\mathcal{F}(\cdot)$ | the oracle function |
| | $f_\phi(\cdot)$ | the proxy model parameterized by $\phi$ |
| | $\mathcal{D}$ | the mutant candidate pool |
| | $\mathcal{D}_0$ | the initial mutant candidate pool |
| *Reinforcement Learning and Policy* | $\mathbb{A}$ | the action space |
| | $s_t$ | the state at step $t$ |
| | $a_t$ | the action at step $t$ |
| | $a_t^{pos}, a_t^{type}$ | the mutation position and substituted amino acid type at step $t$ |
| | $r_t$ | the reward calculated by the proxy at step $t$ |
| | $\gamma$ | the discount rate |
| | $\pi_{\theta_1}, \pi_{\theta_2}$ | the parameterized position and amino acid type action networks |
| | $\mathcal{T}, \tau$ | the collected trajectory |
| | $m_{step}$ | the maximum number of steps in a trajectory |

### A.2 Algorithm for StructEvo framework pipeline

We provide a full description of the StructEvo pipeline in Algorithm 1. The outer loop is the active learning process, where selected mutant candidates are evaluated by the oracle and then iteratively update the candidate pool, followed by proxy model fine-tuning. The inner loop shows the reinforcement learning procedure within each round, including how the mutation trajectories are collected, how rewards are assigned, and how the policy network is updated.

In the RL loop, the trajectory terminates once an improved variant is identified. This design choice simplifies the implementation and provides better control. More importantly, it ensures that each round focuses on relatively short-distance mutations, keeping the proxy model within its current trust region to provide reliable prediction guidance. Longer-distance mutations are achieved through multiple rounds.

The construction of the initial pool $\mathcal{D}_0$ differs slightly between the 4-site tasks and full-length tasks. For 4-site tasks, CLADE [14], an unsupervised hierarchical clustering strategy, is used to sample $K$ candidates from the combinatorial space as the initial pool. The initial sequence is then randomly selected from it. For the full-length tasks, the initial pools are already provided (i.e., initial training sets), and the initial sequence is randomly selected from the top $K$ candidates ranked by oracle fitness. For both task types, we set $K$ to be the same as the oracle budget $N$.

**Algorithm 1** The StructEvo pipeline

**Input:** Total rounds $R$, number of proposed candidates per round $N$, black-box oracle $\mathcal{F}$, starting candidates set $\mathcal{D}_0 = \{X_n, \mathcal{F}(X_n)\}_{n=1}^{|\mathcal{D}_0|}$, reference structure $S^{\text{ref}}$ and sequence $X^{\text{ref}}$, sequence encoder $\mathcal{E}_{\text{seq}}$, structure encoder $\mathcal{E}_{\text{str}}$, trajectory max length $m_{\text{step}}$, total max step $m_{\text{total}}$
Initialize current round $i \leftarrow 0$
**while** $i < R$ **do**
  *// active learning loop begin*
  Fine-tune the proxy model $f_\phi$ with validated candidates $\mathcal{D}_i$
  Initialize candidate buffer $\mathcal{D} \leftarrow \emptyset$
  Initialize trajectory buffer $\mathcal{T} \leftarrow \emptyset$
  Initialize total steps $t_{\text{total}} \leftarrow 0$
  **while** $t_{\text{total}} < m_{\text{total}}$ **do**
    *// RL loop begin*
    Sample an initial sequence $X_0 \in \mathcal{D}_i$
    Initialize state $s_0 \leftarrow X_0$, reward threshold $r_{\text{init}} \leftarrow f_\phi(s_0)$
    step $t \leftarrow 0$, trajectory $\tau \leftarrow \emptyset$, $\text{FLAG}_{\text{stop}} \leftarrow$ `False`
    **while** not $\text{FLAG}_{\text{stop}}$ **do**
      *// delta-structure fusion encoder*
      Delta sequence feature $\Delta h_{\text{seq},t} \leftarrow \mathcal{E}_{\text{seq}}(s_t) - \mathcal{E}_{\text{seq}}(X^{\text{ref}})$
      Delta structure feature $\Delta h_{\text{str},t} \leftarrow f_{\text{proj}}(\Delta h_{\text{seq},t})$
      Approximated mutant structure feature $\hat{h}_{\text{str},t} \leftarrow \mathcal{E}_{\text{str}}(S^{\text{ref}}) + \Delta h_{\text{str},t}$
      Calculate geometric constraint loss $\mathcal{L}_{\text{geo}}(\Delta h_{\text{str},t}, \Delta h_{\text{seq},t})$
      Fusion feature $h_{\text{fusion},t} \leftarrow \texttt{CrossAttention}(\mathcal{E}_{\text{seq}}(s_t), \hat{h}_{\text{str},t}, \hat{h}_{\text{str},t})$
      *// hierarchical action network*
      Sample mutation position $a_t^{\text{pos}} \sim \pi_{\theta_1}(\cdot \mid h_{\text{fusion},t})$
      Sample substituted amino acid type $a_t^{\text{type}} \sim \pi_{\theta_2}(\cdot \mid h_{\text{fusion},t}, a_t^{\text{pos}})$
      Apply mutation to obtain next state $s_{t+1} \leftarrow \texttt{transition}(s_t, (a_t^{\text{pos}}, a_t^{\text{type}}))$
      Assign reward $r_t \leftarrow f_\phi(s_{t+1})$
      **if** $r_t > r_{\text{init}}$ **or** $t \geq m_{\text{step}}$ **then**
        $\text{FLAG}_{\text{stop}} \leftarrow$ `True`
      **end if**
      **if** not $\text{FLAG}_{\text{stop}}$ **then**
        $r_t \leftarrow 0$  *// sparse reward for intermediate steps*
      **end if**
      $\tau \leftarrow \tau \cup \{s_t, a_t, r_t\}$
      $t \leftarrow t + 1$
      $t_{\text{total}} \leftarrow t_{\text{total}} + 1$
    **end while**
    Update $\mathcal{T} \leftarrow \mathcal{T} \cup \tau$
    Update $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t\}$
  **end while**
  Update policy by $\mathcal{L}_{\text{PPO}} + \lambda_{\text{geo}} \cdot \mathcal{L}_{\text{geo}}$ with collected trajectories $\mathcal{T}$
  Select $N$ novel candidates ranked by proxy $\{X_n\}_{n=1}^{N} \leftarrow \texttt{TopN}(\mathcal{D}, f_\phi(\mathcal{D}))$
  Evaluate $\{X_n\}_{n=1}^{N}$ by oracle $\mathcal{F}$
  Update candidate pool $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_i \cup \{X_n, \mathcal{F}(X_n)\}_{n=1}^{N}$
  $i \leftarrow i + 1$
**end while**
return proposed candidates $\mathcal{D}_R$

### A.3 The PPO loss function

The policy, entropy, and value losses follow the standard PPO formulation [50]. Specifically,

$$\mathcal{L}_{policy} = -\mathbb{E}_t \left[ \min \left( r_t(\theta) A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) A_t \right) \right],$$

$$\mathcal{L}_{value} = \mathbb{E}_t \left[ \left( V_\Phi(s_t) - R_t \right)^2 \right],$$

$$\mathcal{L}_{entropy} = -\mathbb{E}_t \left[ H(\pi_\theta(\cdot|s_t)) \right].$$

Here, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ denotes the policy probability ratio, $A_t$ is the estimated advantage, $V_\Phi(s_t)$ is the value function, $R_t$ is the return target, and $H(\pi_\theta(\cdot|s_t))$ represents the policy entropy. Other notations follow the original PPO paper.

### A.4 The detach operator

As introduced in Sec. 4.2, in the proposed structure-aligned hierarchical action network, the localized feature is extracted and detached from the fusion feature. We next show how the *detach* operation blocks gradient flow and leads to a cleaner learning signal. For simplicity, we use $h$ below to denote $h_{\text{fusion}}$ in the main text.

As introduced in Schulman et al. [50], the training objective of PPO consists of three components:

$$L_t^{\text{CLIP+VF+S}}(\theta) = \hat{\mathbb{E}}_t \Big[ L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S\big[\pi_\theta\big](s_t) \Big], \tag{8}$$

where the main objective $L_t^{\text{CLIP}}(\theta)$ is:

$$L_t^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \Big[ \min \left( r_t(\theta)\, \hat{A}_t,\ \text{clip}\left( r_t(\theta),\, 1-\epsilon,\, 1+\epsilon \right) \hat{A}_t \right) \Big], \tag{9}$$

with the probability ratio $r_t(\theta)$ defined as:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}. \tag{10}$$

In our hierarchical action network, the mutation action $a_t$ is decomposed into $a_t = (a_t^{\text{pos}}, a_t^{\text{type}})$, which is sampled sequentially through two action networks $\pi_{\theta_1}$ and $\pi_{\theta_2}$. Therefore, the probability of sampled action $a_t$ at given state $s_t$ corresponds to:

$$\pi_\theta(a_t \mid s_t) = p(a_t^{\text{pos}}, a_t^{\text{type}}) = \pi_{\theta_1}(a_t^{\text{pos}} \mid h) \cdot \pi_{\theta_2}(a_t^{\text{type}} \mid h, a_t^{\text{pos}}). \tag{11}$$

During policy optimization, gradients of the PPO objective are computed. We focus on the joint log-likelihood component $\nabla_{\theta_1,\theta_2} \log p(a^{\text{pos}}, a^{\text{type}})$ associated with the action network. Under the hierarchical regime, it factorizes as:

$$\begin{aligned}
&\nabla_{\theta_1,\theta_2} \log p(a^{\text{pos}}, a^{\text{type}}) && (12)\\
&= \nabla_{\theta_1,\theta_2} \log \left[ \pi_{\theta_1}(a^{\text{pos}} \mid h) \cdot \pi_{\theta_2}(a^{\text{type}} \mid h, a^{\text{pos}}) \right] && (13)\\
&= \nabla_{\theta_1,\theta_2} \left[ \log \pi_{\theta_1}(a^{\text{pos}} \mid h) + \log \pi_{\theta_2}(a^{\text{type}} \mid h, a^{\text{pos}}) \right] && (14)\\
&= \nabla_{\theta_1} \log \pi_{\theta_1}(a^{\text{pos}} \mid h) + \nabla_{\theta_2} \log \pi_{\theta_2}(a^{\text{type}} \mid h, a^{\text{pos}}), && (15)
\end{aligned}$$

where the sampling operation blocks the gradients of the amino acid decision back-propagating to the position action network. The gradients from $\pi_{\theta_2}$ cannot influence the preceding feature fusing module, since the detaching operation enforces $\nabla_h\, \pi_{\theta_2}(a^{\text{type}} \mid h, a^{\text{pos}}) = \mathbf{0}$. As a result, the position network $\pi_{\theta_1}$ is optimized to capture the overall contribution of selecting mutation sites, yielding a cleaner and more stable learning signal for position selection. This decomposition enables the structure-enhanced features to more directly and effectively guide the identification of functional mutation positions, thereby reducing learning complexity and improving data efficiency under limited experimental feedback.

### A.5 The geometric constraint

In Sec. 4.3, we introduce a per-residue geometric constraint loss between the delta sequence feature $\Delta h_{\text{seq}} \in \mathbb{R}^{L\times d_{\text{seq}}}$ and delta structure feature $\Delta h_{\text{str}} \in \mathbb{R}^{L\times d_{\text{str}}}$, formulated as:

$$\mathcal{L}_{\text{geo}}(\Delta h_{\text{str}}, \Delta h_{\text{seq}}) = \frac{1}{L}\sum_{i=1}^{L}\left(\frac{\|\Delta h^i_{\text{str}}\|}{\sqrt{d_{\text{str}}}} - c * \frac{\|\Delta h^i_{\text{seq}}\|}{\sqrt{d_{\text{seq}}}}\right)^2, \tag{16}$$

where $c$ is the scaling factor. Here, we provide an intuitive explanation of how the proposed geometric constraint encourages reference-centered norm consistency, analogous to a Lipschitz-type upper bound. We refer readers to Asadi et al. [52] for a more rigorous analysis on how Lipschitz continuity can promote smoother mappings and more stable learning signals in RL framework.

For simplicity, we denote the delta feature matrices as:

$$\mathbf{x} = \Delta h_{\text{seq}} = \{x_{ij}\}_{i=1,\dots,L}^{j=1,\dots,d_{\text{seq}}}, \quad \mathbf{y} = \Delta h_{\text{str}} = \{y_{ij}\}_{i=1,\dots,L}^{j=1,\dots,d_{\text{str}}}. \tag{17}$$

The corresponding $\ell_2$ norms are defined as:

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{L}\sum_{j=1}^{d_{\text{seq}}} x_{ij}^2}, \quad \|\mathbf{x}_i\| = \sqrt{\sum_{j=1}^{d_{\text{seq}}} x_{ij}^2}, \quad \|\mathbf{y}\| = \sqrt{\sum_{i=1}^{L}\sum_{j=1}^{d_{\text{str}}} y_{ij}^2}, \quad \|\mathbf{y}_i\| = \sqrt{\sum_{j=1}^{d_{\text{str}}} y_{ij}^2}. \tag{18}$$

Under this notation, the geometric constraint loss is formulated as:

$$\mathcal{L}_{\text{geo}} = \frac{1}{L}\sum_{i=1}^{L}(\frac{\|\mathbf{y}_i\|}{\sqrt{d_{\text{str}}}} - c\cdot\frac{\|\mathbf{x}_i\|}{\sqrt{d_{\text{seq}}}})^2. \tag{19}$$

We consider the idealized global minimum where $\mathcal{L}_{\text{geo}} = 0$. This implies:

$$\|\mathbf{y}_i\| = \left(c\cdot\frac{\sqrt{d_{\text{str}}}}{\sqrt{d_{\text{seq}}}}\right)\cdot\|\mathbf{x}_i\| \quad \text{for all} \quad i\in\{1,\dots,L\}. \tag{20}$$

We denote $c_0 = c\cdot\frac{\sqrt{d_{\text{str}}}}{\sqrt{d_{\text{seq}}}}$. Squaring both sides, we obtain:

$$\sum_{j=1}^{d_{\text{str}}} y_{ij}^2 = \|\mathbf{y}_i\|^2 = c_0^2\cdot\|\mathbf{x}_i\|^2 = c_0^2\cdot\sum_{j=1}^{d_{\text{seq}}} x_{ij}^2 \quad \text{for all} \quad i\in\{1,\dots,L\}. \tag{21}$$

Summing over $i$ from 1 to $L$, we obtain:

$$\sum_{i=1}^{L}\sum_{j=1}^{d_{\text{str}}} y_{ij}^2 = \sum_{i=1}^{L} c_0^2\cdot\|\mathbf{x}_i\|^2 = c_0^2\cdot\sum_{i=1}^{L}\sum_{j=1}^{d_{\text{seq}}} x_{ij}^2. \tag{22}$$

Thus,

$$\|\mathbf{y}\|^2 = c_0^2\cdot\|\mathbf{x}\|^2. \tag{23}$$

Therefore,

$$\|\mathbf{y}\| = c_0\cdot\|\mathbf{x}\|. \tag{24}$$

We define the projection mapping from delta sequence manifold $\mathbf{x} = \Delta h_{\text{seq}}$ to delta structure manifold $\mathbf{y} = \Delta h_{\text{str}}$ as $\mathcal{M} : \mathbf{x} \mapsto \mathbf{y}$. By definition, the mapping $\mathcal{M}$ is *$K$-Lipschitz continuous* if any input $\mathbf{x}$ satisfies:

$$\|\mathcal{M}(\mathbf{x}) - \mathcal{M}(\mathbf{x}^{\text{ref}})\| \le K\cdot\|\mathbf{x} - \mathbf{x}^{\text{ref}}\|. \tag{25}$$

In the context of our delta representation, any delta feature is defined as the difference $\Delta h_{\text{seq}} = h_{\text{seq}} - h^{\text{ref}}_{\text{seq}}$. Accordingly, the delta feature of the reference protein is $\mathbf{x}^{\text{ref}} = \Delta h^{\text{ref}}_{\text{seq}} = h^{\text{ref}}_{\text{seq}} - h^{\text{ref}}_{\text{seq}} = \mathbf{0}$. This simplifies Eq. 25 to:

$$\|\mathbf{y}\| = \|\mathcal{M}(\mathbf{x})\| \le K\cdot\|\mathbf{x}\|. \tag{26}$$

Comparing Eq. 24 and Eq. 26, we meet the upper bound of the Lipschitz continuity with the Lipschitz constant:

$$K = c_0 = c\cdot\frac{\sqrt{d_{\text{str}}}}{\sqrt{d_{\text{seq}}}}. \tag{27}$$

Thus, the geometric constraint loss $\mathcal{L}_{\text{geo}}$ enables a similar mechanism with Lipschitz continuity, encouraging consistent sensitivity in the projection mapping and leading to more stable learning signals for RL training process. We also conduct experiments to investigate the sensitivity of this geometric constraint, as detailed in Appendix F.7.

## B Protein fitness landscape modeling

Protein fitness landscape modeling aims to learn a fine-grained mapping from proteins to fitness values.

**Sequence-based models** [62, 63, 51, 64, 65] formulate protein residues as discrete tokens analogous to natural language, and are pre-trained on large-scale sequence databases via masked language modeling (MLM; 66), enabling zero-shot fitness prediction.

**Structure-based models** [41, 42] typically leverage Graph Neural Network (GNN; 67) to explicitly encode three-dimensional protein structures as nodes and edges. Inverse-folding models [43–45] leverage structural representations to predict protein sequence.

More recently, **hybrid models** have incorporated both sequence and structure to obtain more informative protein representations. Su et al. [46] introduced a concatenated structure-aware vocabulary, while Li et al. [47] proposed a structure quantization module with a disentangled attention mechanism for modality fusion.

## C Experimental details

### C.1 Combinatorial mutation benchmarks

The GB1 and PhoQ datasets are summarized at Table 5. Fitness of GB1 dataset measures the binding affinity of human protein GB1 domain to the antibody IgG-Fc, while fitness of PhoQ refers to kinase and phosphatase responsiveness of PhoP/PhoQ regulatory system. The key advantage of these datasets is that they are sufficiently large to almost exhaustively cover the theoretical four-site combinatorial space ($20^4 = 160,000$), thereby serving as reliable assessments for MLDE methods.

Table 5: **Description of GB1 and PhoQ benchmarks.** The last three columns report the proportion of mutants whose normalized fitness is lower than a given threshold, highlighting the sparsity and challenge of these benchmarks. aa: amino acids.

| Name | UniProt ID | Protein Length | Dataset Size | Mutation Sites | <0.3 | <0.1 | = 0 |
|---|---|---|---|---|---|---|---|
| **GB1** | P19909 | 56aa | 149,361 | V39,D40, G41,V54 | 99.30% | 97.30% | 19.74% |
| **PhoQ** | P23837 | 486aa | 140,517 | A284,V285, S288,T289 | 99.96% | 98.65% | 63.70% |

### C.2 Full-length mutation benchmarks

Following the settings of Kirjner et al. [30], both GFP and AAV tasks are partitioned into two levels, *medium* and *hard*, by controlling the initial variant pool. For *medium* level, initial mutants are sampled from the *20th–40th* fitness percentiles of the full dataset with a minimum pairwise Hamming distance of 6, resulting in 2,139 and 2,828 initial variants for AAV and GFP, respectively. For *hard* level, initial sequences are drawn from the bottom *30th* percentile with a minimum distance of 7, yielding 3,448 and 2,426 variants for AAV and GFP, respectively. Table 6 and 7 describe the statistic of four initial sets. Figure 5 provides a visualized comparison on fitness distributions among all data, medium initial sets and hard initial sets.

Table 6: **Description of initial sets in the AAV benchmark.** Mean Fitness is the mean normalized fitness of top 128 sequences in subsets.

| Level | Range(%) | Gap | Size | Mean Fitness |
|---|---|---|---|---|
| Medium | 20-40th | 6 | 2139 | 0.376 |
| Hard | <30th | 7 | 3448 | 0.326 |

Table 7: **Description of initial sets in the GFP benchmark.** Mean Fitness is the mean normalized fitness of top 128 sequences in subsets.

| Level | Range(%) | Gap | Size | Mean Fitness |
|---|---|---|---|---|
| Medium | 20-40th | 6 | 2828 | 0.232 |
| Hard | <30th | 7 | 2426 | 0.092 |

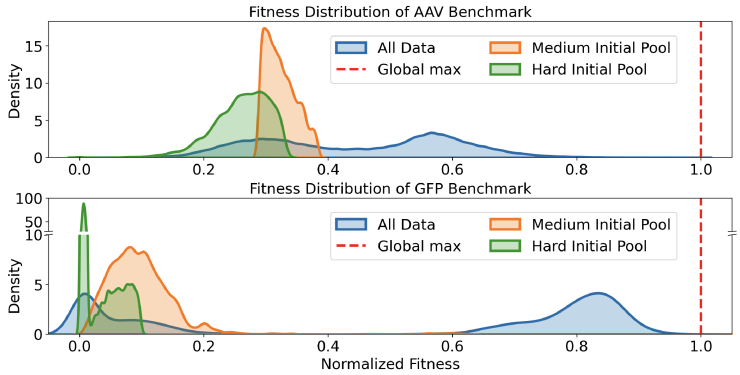


Figure 5: **Kernel density estimation (KDE) of fitness distributions** for AAV and GFP benchmarks.

### C.3 Baselines for combinatorial benchmarks

We evaluate various baselines on four-site combinatorial benchmarks, GB1 and PhoQ.

**CMA-ES** [56]. The Covariance Matrix Adaptation Evolution Strategy optimizes protein sequences through a continuous relaxation of the discrete mutation space, where each sequence is represented by a one-hot encoding and optimized in a continuous domain.

**BO** [55]. Bayesian optimization formulates protein directed evolution as a black-box optimization problem by updating an acquisition function after each round. The implementation of BO is derived from FLEXS benchmark [11].

**AdaLead** [11]. This work adopts an adaptive greedy search algorithm for directed evolution by iteratively selecting sequences whose fitness exceeds a dynamically updated threshold based on the previous batch, and generating new candidates via mutation and recombination of these high-performing variants.

**MLDE** [53]. This work first incorporates machine learning into the combinatorial directed evolution workflow by training an ensemble model to increase throughput with in-silico modeling.

**ftMLDE** [57]. This work evaluates how different protein encoding methods and training set partitioning strategies affect the performance of protein directed evolution, with EVmutation and MSATransformer adopted as our baselines following Wang et al. [16].

**CLADE** [14]. This work proposes an unsupervised hierarchical clustering strategy to explore the combinatorial mutation space, followed by in silico screening of selected mutants using a supervised proxy model.

**CLADE 2.0** [15]. This work extends CLADE by improving the initial sampling stage through ensembling multiple evolutionary scores, resulting in more robust performance with reduced sensitivity to hyperparameter choices.

**PEX** [12]. This work introduces proximal exploration, which biases evolutionary search toward high-fitness mutants with low mutation counts by applying a distance-based preference around the wild-type sequence.

**EvoPlay** [38]. This work models protein directed evolution by a neural network-guided Monte Carlo tree search (MCTS) procedure, which is more interpretable than optimization methods that performed in a latent space.

**KnowRLM** [16]. This work introduces an Amino Acid Knowledge Graph (AAKG) together with a dynamic window mechanism to guide mutation selection using physicochemical properties, optimizing by PPO.

Since MLDE methods [53, 57, 14, 15] need to enumerate all possible mutants to construct a scoring vector, they may not be suitable for full-length benchmarks, where the exponentially large mutation space ($20^L$) makes such enumeration infeasible. We additionally conduct proxy-based search baselines, including random search and brute-force search, to evaluate the contributions of both the proxy model and the search methods; the results are reported in Appendix F.3.

### C.4 Baselines for full-length benchmarks

This section introduces additional baselines for full-length benchmarks.

**GGS** [30]. This work employs a graph-based smoothing approach to train a smoothed proxy model, which defines a discrete energy function for mutant sampling via Gibbs-with-Gradients. Following prior work [13, 35] on the GFP and AAV benchmarks, we use the provided checkpoints of both the smoothed proxy and the oracle in this work.

**LatProtRL** [13]. This work encodes protein mutants into latent representations using a pretrained variational encoder–decoder and treats these latent features as states for PPO, with actions defined as small continuous perturbations in the latent space, which improves sample efficiency by reducing the dimension of the mutation space.

**VLGPO** [35]. This work introduces Variational Latent Generative Protein Optimization, which optimizes the sampling strategy in a continuous latent space using a flow-matching prior. As the original work demonstrates better performance when using an unsmoothed fitness predictor proposed in [30], we follow this choice and train the model for 15 rounds in an active learning setting.

**GFN-$\delta$-CS** [36]. GFN-AL [34] leverages Generative Flow Networks (GFlowNets; 68) within an active learning loop for protein optimization. This work further improves GFN-AL by introducing a controllable conservation factor $\delta$ to explicitly balance exploration and exploitation. In our experiments, we set $\delta = 0.01$ to prioritize fitness optimization.

Since these methods rely on pretraining an encoder–decoder or performing optimization in continuous latent spaces, we carefully considered their compatibility and, for fairness and clarity of comparison, did not include them as baselines for the combinatorial benchmarks.

We additionally evaluate classical proxy-based search methods for full-length benchmarks, including hill climbing, beam search and vanilla Gibbs sampling. The results are reported in Appendix F.3. To assess the effectiveness of reference structure input, we also develop simple structure-aware extensions of two strong existing baselines, VLGPO and GFN-$\delta$-CS. Implementation and experiment details are provided in Appendix F.4.

### C.5 Metrics for combinatorial benchmarks

The *mean* and *maximum* fitness are computed over the combined candidate set, which includes 96 mutants from initial set sampled by CLADE [14] via an unsupervised hierarchical clustering approach, 288 mutants proposed across three rounds ($N = 96$ per round), as well as the top 96 mutants selected by the proxy model after the final training round, resulting in a total of 480 candidates.

The normalized discounted cumulative gain (NDCG) is calculated between predictions and the ground truth fitness in the entire mutation space with size of $20^4 = 160,000$. We compute NDCG using `metrics.ndcg_score` implemented in scikit-learn framework [69].

### C.6 Metrics for full-length benchmarks

The *mean* and *maximum* fitness are computed on the top-N of the proposed sequences across 15 rounds, where $N$ is set to 128. The *diversity* of $N$ proposed sequences $\mathcal{D} = \{X_i\}_{i=1}^N$ is defined as mean pairwise Hamming distance [70] among sequences:

$$Diversity(\mathcal{D}) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \texttt{Hamming}(X_i, X_j). \tag{28}$$

The *novelty* of $N$ proposed sequences $\mathcal{D} = \{X_i\}_{i=1}^{N}$ is defined as the mean minimum Hamming distance from each proposed sequence to the initial set $\mathcal{D}_0$:

$$Novelty(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\left(\min_{X^* \in \mathcal{D}_0}\{\texttt{Hamming}(X_i, X^*)\}\right). \tag{29}$$

# D Implementation details

## D.1 Structural processing

Table 8: Information of the reference protein structures used in our work.

| Protein | Source | Structure ID | Chain ID | Position | Length |
|---|---|---|---|---|---|
| **GB1** | PDB | 2GI9 | A | 1–56 | 56 |
| **PhoQ** | AFDB | AF-Q8FIB8-F1-v6 | A | 188–376 | 189 |
| **AAV** | PDB | 3NG9 | A | 561–588 | 28 |
| **GFP** | PDB | 1GFL | A | 2–238 | 237 |

The sources of protein structures used in this work are listed in Table 8. Experimental structures of GB1, AAV and GFP are obtained from the Protein Data Bank (PDB) [71], the largest global database of biological macromolecular structures. As the crystal structure of PhoQ is unavailable in the PDB, we select a high-confidence predicted structure with average pLDDT of 82.94 from the AlphaFold Protein Structure Database (AFDB) [18] instead.

## D.2 Training details

**Model architecture.** We employed the open-source ESM-2 [51] as PLM for encoding protein sequences, and ESM-IF [43] as the inverse folding model for encoding structures. The *ValueNet*, required for computing the value loss in PPO, is implemented as a multi-layer MLP (following KnowRLM [16]). The *ActionNet* (described in Section 4.2) consists of a position action network followed by an amino acid type action network, each is implemented as a two-layer linear network. The *projector*, aiming to transform the sequence-derived delta features into the structural feature space, is implemented as a lightweight linear projection layer. Our code is developed within python 3.11 and PyTorch framework, running on a single NVIDIA A800 80G GPU.

**Proxy model training details.** For four-site combinatorial benchmarks, we adopt the structure-aware ProSST [47] as our proxy model. ProSST incorporates a structure quantization module and a transformer-based disentangled attention mechanism to better align sequence and structural features. We append a regression head to the backbone, with a hidden size of 128 and a scalar as output, to convert it into a fitness predictor. During all fine-tuning and inference process of ProSST, we utilize the reference structure coordinates as structural inputs. For full-length benchmarks, we use the provided checkpoints of the smoothed proxy in Kirjner et al. [30] by convention. The proxy is based on a convolutional neural network (CNN) architecture and encodes protein sequences to one-hot embeddings as inputs. For all benchmarks, the proxy model is fine-tuned on the newly proposed mutant candidates for 30 epochs after each round, using a batch size of 96 and a learning rate of $2 \times 10^{-3}$. The total compute time for an individual experimental run is around 2 hours.

**RL training details.** The reinforcement learning procedure is optimized using Proximal Policy Optimization (PPO) [50], supplemented by the `Stable-Baselines3` [72] framework. For all benchmarks, we train the PPO using 8 parallel environments, with a clipping range of 0.3, entropy coefficient of 0.0, discount factor of 0.99, learning rate of $3 \times 10^{-4}$, and batch size of 64. The max episode length $m_{step}$ is set to 3 for each round. The total number of training steps is set to 3,000 for combinatorial benchmarks and 15,000 for full-length benchmarks. For geometric constraint, the weight $\lambda_{\text{geo}}$ is set to 0.05 and the scaling factor $c$ is set to 1.

# E Additional experiment results

## E.1 Full results on 4-site benchmarks

We provide results on 3 optimization rounds at Table 9 and Table 10.

Table 9: **Results on GB1 benchmarks across 3 rounds**. The best scores **bolded** and the second best underlined. All baseline results are derived from Wang et al. [16].

| | Round 1 | | | Round 2 | | | Round 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | Max↑ | Mean↑ | NDCG↑ | Max↑ | Mean↑ | NDCG↑ | Max↑ | Mean↑ | NDCG↑ |
| MLDE | 0.650 | 0.183 | 0.767 | 0.680 | 0.217 | 0.794 | 0.684 | 0.203 | 0.789 |
| ftMLDE (EVmutation) | 0.725 | 0.233 | 0.791 | 0.770 | 0.280 | 0.814 | 0.935 | 0.414 | 0.833 |
| ftMLDE (Transformer) | 0.761 | 0.239 | 0.792 | 0.814 | 0.298 | 0.819 | 0.932 | 0.416 | 0.813 |
| EvoPlay | 0.837 | 0.433 | 0.826 | 0.834 | 0.457 | 0.839 | 0.840 | 0.460 | 0.847 |
| CLADE | 0.785 | 0.309 | 0.801 | 0.802 | 0.303 | 0.803 | 0.835 | 0.458 | 0.857 |
| CLADE 2.0 | 0.886 | 0.376 | 0.808 | 0.903 | 0.419 | 0.858 | 0.935 | 0.491 | 0.879 |
| KnowRLM | 0.931 | 0.494 | 0.851 | 0.972 | 0.534 | 0.862 | 0.972 | 0.562 | 0.884 |
| **StructEvo (ours)** | **0.942** | **0.514** | **0.870** | **0.973** | **0.540** | **0.902** | **0.990** | **0.587** | **0.932** |

Table 10: **Results on PhoQ benchmarks across 3 rounds**. The best scores **bolded** and the second best underlined. All baseline results are derived from Wang et al. [16].

| | Round 1 | | | Round 2 | | | Round 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | Max↑ | Mean↑ | NDCG↑ | Max↑ | Mean↑ | NDCG↑ | Max↑ | Mean↑ | NDCG↑ |
| MLDE | 0.309 | 0.069 | 0.753 | 0.364 | 0.087 | 0.775 | 0.361 | 0.095 | 0.791 |
| ftMLDE (EVmutation) | 0.297 | 0.072 | 0.754 | 0.431 | 0.114 | 0.807 | 0.436 | 0.115 | 0.804 |
| ftMLDE (Transformer) | 0.346 | 0.073 | 0.756 | 0.414 | 0.108 | 0.802 | 0.422 | 0.117 | 0.815 |
| EvoPlay | 0.443 | 0.119 | 0.782 | 0.444 | 0.135 | 0.786 | 0.474 | 0.143 | 0.804 |
| CLADE | 0.319 | 0.070 | 0.759 | 0.441 | 0.089 | 0.762 | 0.467 | 0.089 | 0.777 |
| CLADE 2.0 | 0.345 | 0.118 | 0.781 | 0.383 | 0.125 | 0.779 | 0.399 | 0.148 | 0.814 |
| KnowRLM | 0.486 | 0.129 | 0.821 | 0.532 | 0.152 | 0.816 | **0.658** | 0.157 | 0.819 |
| **StructEvo (ours)** | **0.530** | **0.158** | **0.844** | **0.581** | **0.169** | **0.848** | 0.640 | **0.186** | **0.856** |

## E.2 Full ablation results on full-length benchmarks

See Table 11 for full ablation results. Ablation settings are: (1) removing delta module by directly fusing mutant sequence with the static reference structure via cross-attention, (2) removing structure and fusing two sequence features, (3) relying solely on mutant sequence, and (4) replacing the hierarchical action network with a flat network that jointly samples both actions. The results on the two medium settings show trends consistent with those on the hard setting, as analyzed in Sec. 5.3.

# F Additional in-depth analysis

We discuss additional ablation effects in this section, including analysis on structure prediction models (Sec. F.1), proxy model robustness (Sec. F.2), and weights for geometric constraint (Sec. F.7).

## F.1 Analysis on structure prediction models

In this section, we assess the sensitivity of structure prediction models to point mutations on GFP mutants. We choose Protenix [22] as the representative structure prediction model for assessment. Compared to recent well-known models such as AlphaFold3, Protenix is completely open-source while achieving comparable performance across multiple structure prediction benchmarks.

Specifically, we perform stratified sampling on the GFP dataset by partitioning mutant sequences into 10 bins according to their fitness values, and randomly selecting 10 mutant sequences from each bin, resulting in a total of 100 sequences. The average Hamming distance of these 100 sequences is 7.5, indicating a moderate level of mutational diversity for evaluating model sensitivity.

Table 11: **Full results of ablation studies.** w/o: without.

| | Method | Mean↑ | Max↑ | Diversity | Novelty |
|---|---|---|---|---|---|
| *AAV medium* | **StructEvo** | **0.82**±0.01 | **0.86**±0.01 | 4.5±0.7 | 8.6±0.2 |
| | (1) w/o delta | 0.79±0.00 | 0.83±0.01 | 5.5±0.8 | 7.7±0.0 |
| | (2) w/o structure | 0.77±0.01 | 0.81±0.02 | 4.7±1.3 | 8.1±0.7 |
| | (3) w/o reference | 0.77±0.01 | 0.81±0.02 | 4.8±2.0 | 7.8±0.0 |
| | (4) w/o hierarchy | 0.78±0.01 | 0.80±0.01 | 5.3±1.3 | 7.9±0.6 |
| *AAV hard* | **StructEvo** | **0.83**±0.03 | **0.87**±0.04 | 4.5±0.3 | 8.4±0.5 |
| | (1) w/o delta | 0.80±0.03 | 0.84±0.05 | 4.9±0.6 | 8.3±0.7 |
| | (2) w/o structure | 0.75±0.03 | 0.79±0.04 | 4.9±1.8 | 8.6±0.7 |
| | (3) w/o reference | 0.75±0.01 | 0.81±0.01 | 7.5±1.4 | 8.7±0.4 |
| | (4) w/o hierarchy | 0.77±0.02 | 0.81±0.02 | 5.2±1.4 | 7.9±0.3 |
| *GFP medium* | **StructEvo** | **1.11**±0.02 | **1.17**±0.02 | 6.3±0.8 | 11.9±0.8 |
| | (1) w/o delta | 1.09±0.02 | 1.13±0.02 | 6.2±1.3 | 10.9±1.1 |
| | (2) w/o structure | 1.05±0.02 | 1.09±0.02 | 7.4±1.7 | 10.7±0.5 |
| | (3) w/o reference | 1.06±0.05 | 1.10±0.05 | 6.2±1.6 | 11.5±1.6 |
| | (4) w/o hierarchy | 0.91±0.01 | 0.98±0.01 | 10.5±2.5 | 7.4±1.4 |
| *GFP hard* | **StructEvo** | **1.00**±0.04 | **1.05**±0.05 | 4.9±0.3 | 11.4±0.6 |
| | (1) w/o delta | 0.97±0.05 | 1.01±0.05 | 4.7±0.3 | 11.0±1.1 |
| | (2) w/o structure | 0.83±0.04 | 0.89±0.02 | 5.7±1.2 | 11.7±0.8 |
| | (3) w/o reference | 0.84±0.06 | 0.93±0.05 | 5.9±0.9 | 11.3±1.2 |
| | (4) w/o hierarchy | 0.75±0.02 | 0.82±0.01 | 13.8±2.2 | 13.0±1.8 |

For each sequence, we predict its structure using the default Protenix inference configuration, including MSA construction via UniRef100, 5 Pairformer cycles, and 200 diffusion steps. We adopt the `protenix_base_default_v1.0.0` version and run all experiments on a single NVIDIA A800 80GB GPU. Due to the expensive computational costs, we only generate one structure sample per sequence. The entire MSA search and structure prediction process for 100 sequences takes approximately 13 hours, making full-scale structure prediction prohibitive within the RL training loop, which requires evaluating substantially more candidates.

Under this setting, we evaluate two aspects. First, we examine the Pearson and Spearman correlations (implemented by `scipy.stats`) between three commonly used confidence metrics and fitness values:

- *pLDDT (predicted Local Distance Difference Test)*: Higher pLDDT values indicate greater confidence in local structural predictions at the residue level.
- *GpDE (Global predicted Distance Error)*: Lower distance errors indicate more reliable predictions. Therefore, we use the negative values of GpDE scores to compute correlations.
- *pTM (predicted TM-score)*: Values closer to 1 indicate higher confidence in the overall structural prediction, capturing the reliability of global structure.

The results are shown in Figure 6. All three confidence metrics exhibit consistently weak correlations with fitness, indicating a poor alignment between structure prediction confidence and protein functionality. Notably, the highest observed correlation (0.29 for pLDDT-Pearson) remains below 0.3, suggesting that confidence scores are not sufficiently sensitive to the subtle structural variations induced by point mutations.

To further investigate the sensitivity of predicted structures, we conduct two analyses:

- *Reference-based difference*: We calculate the structure deviation between the 100 predicted mutant structures with the reference structure, and examine the correlation between the structural deviation and fitness value.
- *Pairwise difference*: For each pair in the 100 predicted mutant structures (a total of 4950 pairs), we examine the correlation between structural difference and the corresponding fitness difference. The fitness difference is defined as the absolute fitness gap.

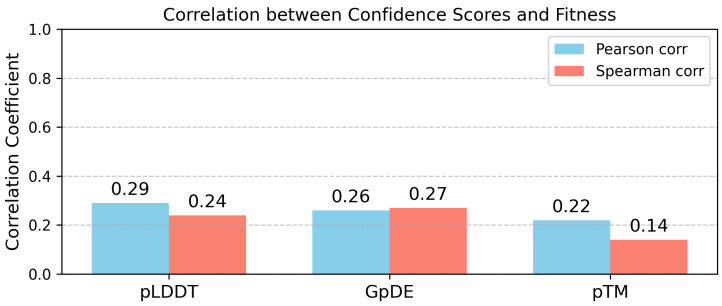


Figure 6: Correlation between Confidence Scores and Fitness.

In both analyses, the structural difference is measured by $C\alpha$-RMSD (root mean square deviation) with rigid-body alignment implemented by the `BioPython.Superimposer` module.

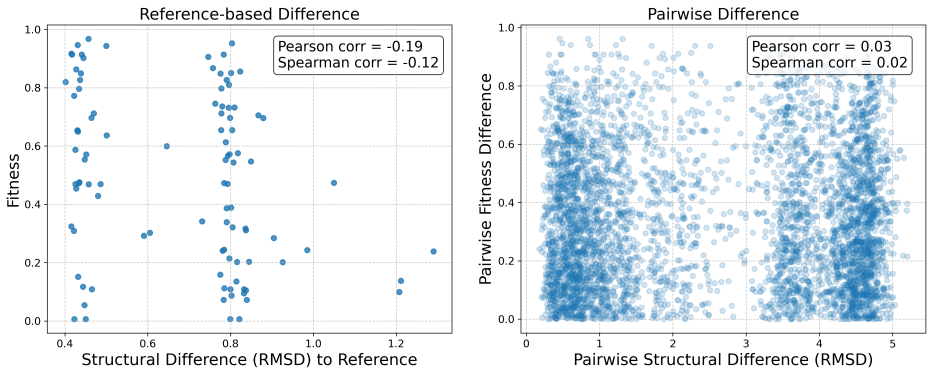


Figure 7: Samples of reference-based difference (left) and pairwise difference (right).

The results are shown in Figure 7. We find that both structural differences reveal negligible relationships with fitness values. In the reference-based setting, structural deviation from the reference shows weak negative correlations with fitness (Pearson -0.19, Spearman -0.12). In the pairwise setting, the correlation is much lower (Pearson 0.03, Spearman 0.02). The diffuse scatter distributions further indicate weak relationship between structural variations and functional changes. Notably, these correlations are even weaker than those of confidence scores. This suggests that current structure prediction outputs yield weak relationships to point mutations, limiting their utility for modeling fine-grained fitness landscape and guiding protein optimization.

Given the high computational costs of structure prediction, a potential alternative strategy is to periodically update the reference structure with newly predicted mutant structure, for example every $T$ steps or when the accumulated mutation distance exceeds a threshold. However, this approach requires careful evaluation of update schedules. We leave this exploration to future work.

### F.2 Robustness analysis of the proxy model

To examine the robustness of proxy model, we conduct experiments on the GFP-hard task under several settings:

- *Default*: The proxy model provides rewards during RL. At each round, the proxy selects top-N candidates, which are then evaluated by the oracle and used to iteratively refine the proxy.
- *Oracle-full*: The proxy is replaced by the oracle to provide rewards and can be queried without constraint. This represents an ideal upper bound of the proxy.
- *Oracle-limited*: The proxy is replaced by the oracle, but the number of queries is strictly limited as default setting (i.e., N per round).

The results are listed below:

Table 12: Proxy robustness analysis on GFP-hard task.

| | Mean↑ | Max↑ | Diversity | Novelty |
|---|---|---|---|---|
| *Default* | 1.00±0.04 | 1.05±0.05 | 4.9±0.3 | 11.4±0.6 |
| *Oracle-full* | 1.07±0.01 | 1.12±0.02 | 12.9±2.9 | 13.1±1.2 |
| *Oracle-limited* | 0.41±0.03 | 0.49±0.04 | 16.7±4.2 | 5.8±0.9 |

We observe that *oracle-full* achieves the best performance as an idealized upper bound setting. However, *this setting is impractical in MLDE, as oracle queries correspond to costly wet-lab experiment budgets*. In contrast, *oracle-limited* shows a significant performance drop, indicating that the RL agent collects insufficient trajectories under limited signals. This further demonstrates that the frequent rewards from the proxy model are crucial for effective evolution. We will include a more comprehensive analysis of proxy effects on different rounds and tasks in future work.

### F.3 Isolation of proxy and RL policy

To isolate the contribution of RL policy and proxy model, we conduct the following proxy-based searching methods as additional baselines:

- **Random**: randomly select single mutations each round without using the proxy.
- **Brute-force** (4-site benchmarks): rank all 160,000 candidates by proxy and select top N=96 for each round.
- **Hill climbing** (full-length benchmarks): use proxy to rank all single-mutation neighbors of each current sequence, and greedily select the best N candidates for the next round.
- **Beam search** (full-length benchmarks): use proxy to rank all single-mutation neighbors, while keeping a beam width of M=5 paths for each sequence.
- **Gibbs sampling** (full-length benchmarks): we conduct a *vanilla* Gibbs sampling that uses proxy to rank all single-mutation neighbors and sample new candidates from a Boltzmann distribution. One of our original baselines, GGS, is also an expansion of Gibbs sampling that adds graph-based smoothing.

For each method, we conduct 2 experimental settings:

- **Full-round**: Starting from the initial pool, we update the proxy using candidates selected by each method. We report performance at R rounds (R=3 for 4-site and R=15 for full-length).
- **Fixed-proxy**: We fix the fine-tuned proxy model at the last round in our main experiment and only perform the last round. This comparison can directly compare the effectiveness of each search strategy based on the same proxy model.

Each baseline is evaluated over three independent runs. Both average results and standard deviations are reported in Table 13 and 14.

Table 13: Comparison with proxy-based search baselines on 4-site benchmarks.

| Method | GB1 | | | PhoQ | | |
|---|---|---|---|---|---|---|
| | Mean | Max | NDCG | Mean | Max | NDCG |
| Random | 0.08±0.00 | 0.22±0.12 | - | 0.00±0.00 | 0.14±0.02 | - |
| Brute-force (full-round) | 0.37±0.01 | 0.60±0.01 | 0.78±0.01 | 0.10±0.01 | 0.23±0.02 | 0.71±0.01 |
| Brute-force (fixed-proxy) | 0.45±0.01 | 0.97±0.02 | - | 0.09±0.01 | 0.55±0.12 | - |
| **StructEvo (ours)** | **0.59±0.01** | **0.99±0.01** | **0.93±0.01** | **0.19±0.02** | **0.64±0.07** | **0.86±0.01** |

The results show that: (1) On the 4-site benchmarks, exhaustive brute-force method selects the global optima under the proxy model, yet still yields lower oracle fitness than StructEvo. This provides direct evidence of prediction errors, where simply maximizing the proxy predicted fitness does not

Table 14: Comparison with proxy-based search baselines on full-length benchmarks.

| Method | AAV-hard | | | | GFP-hard | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Max | Diversity | Novelty | Mean | Max | Diversity | Novelty |
| Random | 0.08±0.01 | 0.22±0.02 | 22.7±0.3 | 11.3±0.3 | 0.04±0.05 | 0.24±0.07 | 39.6±1.2 | 14.5±0.1 |
| Hill climbing (full-round) | 0.56±0.03 | 0.68±0.03 | 4.9±0.2 | 7.7±0.2 | 0.68±0.03 | 0.77±0.03 | 4.8±0.3 | 7.5±0.4 |
| Beam search (full-round) | 0.62±0.03 | 0.68±0.03 | 4.2±0.2 | 7.9±0.3 | 0.76±0.03 | 0.78±0.03 | 4.4±0.2 | 9.3±0.4 |
| Gibbs sampling (full-round) | 0.51±0.03 | 0.60±0.03 | 12.9±0.3 | 8.0±0.4 | 0.43±0.03 | 0.47±0.03 | 15.8±0.3 | 13.1±0.6 |
| Hill climbing (fixed-proxy) | 0.72±0.00 | 0.77±0.01 | 5.8±0.1 | 7.9±0.1 | 0.91±0.00 | 0.95±0.01 | 4.8±0.2 | 11.7±0.1 |
| Beam search (fixed-proxy) | 0.72±0.01 | 0.79±0.01 | 5.7±0.1 | 7.8±0.1 | 0.92±0.00 | 0.95±0.01 | 4.6±0.2 | 11.0±0.1 |
| Gibbs sampling (fixed-proxy) | 0.69±0.03 | 0.72±0.03 | 6.6±0.2 | 8.1±0.4 | 0.87±0.02 | 0.97±0.04 | 6.1±0.2 | 12.4±0.4 |
| **StructEvo (ours)** | **0.83±0.03** | **0.87±0.04** | 4.5±0.3 | 8.4±0.5 | **1.00±0.04** | **1.05±0.05** | 4.9±0.3 | 11.4±0.6 |

necessarily lead to better navigation on the true landscape. In contrast, *the RL search strategy induces a more constrained search distribution that partly mitigates proxy overestimation*.

(2) We observe the same trend on the full-length tasks. StructEvo consistently outperforms all three simple search baselines, further indicating that the RL policy makes an independent contribution beyond proxy quality. Notably, Gibbs sampling achieves the highest diversity but lower fitness, suggesting that much of this exploration is of low value although it explores a broader region of sequence space. In contrast, *StructEvo achieves a more effective balance between exploration and exploitation*.

(3) Performance drops substantially in the full-round setting compared with the fixed-proxy setting. This is further consistent with our observation, that *policy improvement and proxy refinement are mutually complementary in the active-learning loop*. The RL policy collects more informative candidates to fine-tune a better proxy, while the refined proxy provides more reliable guidance for RL search.

### F.4 Comparison of structure-aware baselines

To evaluate more directly the advantage of the reference structure inputs, we implemented simple structure-aware extensions of two strong existing baselines, VLGPO and GFN-$\delta$-CS, providing them with the same per-residue reference structure feature encoding by ESM-IF as used in StructEvo:

- **VLGPO + Struct**: the structural features are normalized and used as weight for the original fitness guidance loss during sampling.
- **GFN-$\delta$-CS + Struct**: the structural features are normalized and used as weight for the mutation action logits produced by the original LSTM network.

We note that there does not exist mature peer-reviewed structure-aware MLDE baselines for fair comparison. A few of MLDE studies involve protein structural information, but they are neither evaluated on standard benchmarks nor provide reproducible code. To the best of our knowledge, StructEvo is the first structure-aware RL-based method on MLDE.

Table 15: Comparison with structure-aware baselines on full-length benchmarks.

| Method | AAV-hard | | | | GFP-hard | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Max | Diversity | Novelty | Mean | Max | Diversity | Novelty |
| VLGPO | 0.71±0.01 | 0.78±0.01 | 3.3±0.2 | 6.9±0.1 | 0.79±0.02 | 0.95±0.00 | 1.9±0.0 | 6.2±0.0 |
| VLGPO + Struc | 0.72±0.01 | 0.79±0.00 | 3.3±0.1 | 7.0±0.1 | 0.81±0.02 | 0.95±0.03 | 1.8±0.0 | 6.2±0.1 |
| GFN-$\delta$-CS | 0.76±0.03 | 0.80±0.03 | 4.0±0.7 | 6.7±1.1 | 0.86±0.06 | 0.90±0.05 | 4.3±1.6 | 8.2±1.5 |
| GFN-$\delta$-CS + Struc | 0.77±0.06 | 0.84±0.08 | 5.8±0.2 | 7.1±2.5 | 0.84±0.04 | 0.87±0.05 | 6.4±1.9 | 13.7±3.8 |
| **StructEvo (ours)** | $\mathbf{0.83 \pm 0.03}$ | $\mathbf{0.87 \pm 0.04}$ | 4.5±0.3 | 8.4±0.5 | $\mathbf{1.00 \pm 0.04}$ | $\mathbf{1.05 \pm 0.05}$ | 4.9±0.3 | 11.4±0.6 |

The results are listed in Table 15. We observe that StructEvo still consistently achieves the highest mean and maximum fitness on both benchmarks, indicating that *the improvement cannot be attributed merely to the additional input structure, but also arise from how structural guidance is integrated into the RL search policy*. Besides, incorporating structure information into two baselines yields modest performance gains in most cases, while the slight drop of GFN-$\delta$-CS on GFP-hard benchmark may be associated with broader exploration of simple position-wise logits reweighting. More robust integration of structural knowledge could be explored in future work.

### F.5 Single-mutation settings on 4-site benchmark

To explore the purely additive setting, which is more common in real-world protein engineering, we conducted the *single-mutation initial library experiments* on the 4-site GB1 benchmark. The initial library is reconstructed to all 19×4=76 single mutants, with mean and maximum fitness of 0.07 and 0.44, respectively. Accordingly, we set the oracle budget (i.e., the number of proposed candidates per round) to N=76 instead of 96. We evaluated two strong baselines, KnowRLM and CLADE 2.0, as well as the following additive baselines, which sample new candidates from the single-mutation neighbors of current sequences at each round:

- **Random**: randomly selects N=76 single-mutation neighbors per round without proxy scoring.
- **Hill climbing**: greedily chooses mutants with the highest proxy scores.
- **Beam search**: maintains K=5 mutants with top proxy scores and randomly selects one for the next round.
- **Gibbs sampling**: samples new candidates from a Boltzmann distribution of proxy scores.

The results are summarized in Table 16:

Table 16: Results of single-mutation settings on GB1 benchmark.

| Method | Round 1 | | | Round 2 | | | Round 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Max | Mean | NDCG | Max | Mean | NDCG | Max | Mean | NDCG |
| Random | 0.44±0.02 | 0.07±0.00 | - | 0.44±0.02 | 0.07±0.01 | - | 0.44±0.02 | 0.07±0.01 | - |
| Hill | 0.49±0.02 | 0.18±0.00 | 0.73±0.00 | 0.52±0.04 | 0.19±0.01 | 0.75±0.00 | 0.58±0.01 | 0.21±0.01 | 0.76±0.00 |
| Beam | 0.47±0.00 | 0.18±0.00 | 0.76±0.00 | 0.48±0.03 | 0.23±0.01 | 0.76±0.01 | 0.49±0.03 | 0.24±0.01 | 0.76±0.01 |
| Gibbs | 0.49±0.03 | 0.21±0.02 | 0.76±0.00 | 0.54±0.06 | 0.26±0.05 | 0.77±0.01 | 0.58±0.00 | 0.29±0.04 | 0.78±0.01 |
| CLADE 2.0 | **0.55±0.07** | 0.18±0.01 | **0.77±0.01** | **0.60±0.11** | 0.23±0.03 | 0.78±0.01 | **0.70±0.09** | 0.27±0.01 | 0.80±0.01 |
| KnowRLM | **0.51±0.03** | 0.24±0.04 | **0.78±0.01** | **0.62±0.02** | 0.33±0.03 | 0.78±0.00 | 0.66±0.04 | 0.36±0.01 | 0.80±0.00 |
| **StructEvo** | **0.55±0.02** | **0.29±0.00** | **0.77±0.00** | **0.66±0.02** | **0.39±0.01** | **0.80±0.00** | **0.72±0.00** | **0.42±0.01** | **0.85±0.01** |

We observe that, under the single-mutation initialization setting, StructEvo still outperforms or achieves comparable performance to all baselines across all rounds, improving mean fitness from 0.29 to 0.42. It also achieves the best proxy model ranking quality with an NDCG of 0.85, surpassing KnowRLM and CLADE 2.0 (both 0.80).

The additive baselines reach limited improvements, as their heavy reliance on proxy scores leads to proxy bias and inaccurate local ranking. In contrast, StructEvo learns an adaptive mutation policy from historical evolutionary trajectories, which enables a better balance between exploration of promising regions and exploitation of high-fitness mutants. Note that the identical results of Random baseline across rounds are due to the cumulative evaluation, where the randomly sampled triple and quadruple mutants are worse than single or double mutants. This further demonstrates the scarcity of valuable mutants in the high dimensional mutation space.

### F.6 Additional case studies of delta features

Our ablations and case studies have shown that the delta module improves MLDE performance. To further validate this approximation, we additionally collected several GFP mutants with experimentally resolved or high-confidence predicted structures, as listed in Table 17. For each mutant, we leveraged a structure encoder (ESM-IF in our work) to compute $h_{\text{true}}$ (the feature of the true mutant structure from PDB or AFDB), $h_{\text{ref}}$ (the feature of the reference structure), and $h_{\text{approx}}$(the approximated structural feature encoded by the delta-structure fusion encoder, i.e., $\hat{h}_{\text{str}}$ in Section 4.1), and then report:

- $Dist.$: Levenshtein distance of amino acid mutations from mutant to reference sequence.
- $Sim_{\text{r}}$: cosine similarity of $h_{\text{true}}$ and $h_{\text{ref}}$.
- $Sim_{\text{a}}$: cosine similarity of $h_{\text{true}}$ and $h_{\text{approx}}$.
- $Sim_{\text{d}}$: cosine similarity of $(h_{\text{true}} - h_{\text{ref}})$ and $(h_{\text{approx}} - h_{\text{ref}})$.

Table 17: More case study of delta features.

| Mutant ID | Source | $Dist.$ | $Sim_{\mathrm{r}}$ | $Sim_{\mathrm{a}}$ | $Sim_{\mathrm{d}}$ |
|---|---|---|---|---|---|
| 5N9O | PDB (experimentally resolved) | 5 | 0.69 | 0.95 | 0.97 |
| 8DPD | PDB (experimentally resolved) | 17 | 0.73 | 0.85 | 0.96 |
| AF-Q8WTD0-F1 | AFDB (AF2 predicted, pLDDT=96.88) | 43 | 0.26 | 0.77 | 0.84 |

Across all three cases, the true mutant structural representations are substantially closer to the approximated mutant representations than the reference. The high Delta Similarity $Sim_{\mathrm{d}}$ (0.84-0.97) further indicates that the projected delta structure yields meaningful structural changes.

Instead of reconstructing coordinate-level mutant structures, the goal of delta feature is to incorporate mutation-associated structural signals inferred from sequence differences *into the end-to-end optimization process*, especially under MLDE settings where mutant structures are unavailable. In this way, the model directly learns how these structural signals provide guidance for navigating the fitness landscape and therefore selecting high-fitness variants.

### F.7 Sensitivity analysis of the geometric constraint weight

As introduced in Sec. 4.3, the overall RL loss consists of PPO loss and geometric constraint loss:

$$\mathcal{L}_{\text{total}} = \underbrace{\mathcal{L}_{\text{policy}} + \lambda_{\text{entropy}} \cdot \mathcal{L}_{\text{entropy}} + \lambda_{\text{value}} \cdot \mathcal{L}_{\text{value}}}_{\text{standard PPO loss}} + \underbrace{\lambda_{\text{geo}} \cdot \mathcal{L}_{\text{geo}}}_{\text{geometric constraint}} . \tag{30}$$

To investigate the effect of the geometric constraint, we vary its weight $\lambda_{\text{geo}}$ and conduct experiments on GFP-hard task. Each setting is evaluated over 5 independent runs. The results are shown in Table 18.

Table 18: Hyperparameter analysis of geometric constraint weight on GFP-hard task.

| | Mean↑ | Max↑ | Diversity | Novelty |
|---|---|---|---|---|
| $\lambda_{\text{geo}} = 0$ | 0.984±0.081 | 1.017±0.072 | 5.101±1.087 | 9.648±1.795 |
| $\lambda_{\text{geo}} = 0.01$ | 0.987±0.052 | 1.031±0.036 | 4.878±0.312 | **11.427**±1.134 |
| $\lambda_{\text{geo}} = 0.05$ (default) | **0.998**±0.041 | **1.047**±0.046 | 4.868±0.336 | 11.425±0.602 |
| $\lambda_{\text{geo}} = 0.1$ | 0.970±0.041 | 1.010±0.043 | 5.289±0.358 | 10.596±0.846 |
| $\lambda_{\text{geo}} = 0.5$ | 0.925±0.043 | 0.991±0.031 | **6.452**±1.534 | 10.677±0.785 |
| $\lambda_{\text{geo}} = 1$ | 0.931±0.034 | 0.974±0.027 | 5.115±0.970 | 9.706±0.115 |

We observe that: (1) increasing $\lambda_{\text{geo}}$ generally improves training stability, as reflected by reduced standard deviations across mean and max fitness metrics, indicating more stable optimization. (2) $\lambda_{\text{geo}} = 0.05$ achieves the best performance on fitness metrics, while both larger and smaller values degrade performance. This is likely because larger weights overly restrict policy updates, whereas smaller weights provide insufficient regularization, leading to noisier gradients and less stable optimization. (3) We found that novelty decreases with larger $\lambda_{\text{geo}}$, likely because stronger constraint limits exploration and favors more proximal candidates. Overall, we select $\lambda_{\text{geo}} = 0.05$ as the default setting reported in the main experiments.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state the scope and contributions of both communities of protein optimization and computer science in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations at Sec. 6 .

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper has disclosed all the necessary information to reproduce the main experimental results, including model architecture (Sec. 4), algorithm (Sec. A.2), and implementation details (Sec. D).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our code in the GitHub repository: `https://github.com/Skyyyyyalker/StructEvo`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings are presented in Sec. 5.1 and Sec. 5.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report standard deviation on results across 5 independent runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the needed computer resources in Sec. D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conduct our work in accordance with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The societal impacts are in Sec. 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release a high-risk pretrained generative model; generated candidates require oracle evaluation, wet-lab validation, and human inspection before use.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original paper and clearly state assets details at Appendix C.1 and C.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.