Oracle-Budgeted Protein Design with Stochastic Search and Iterative Imitation

Ashima Khanna-Reiter[1] **Dominik G. Grimm**[1,2]

[1] Technical University of Munich, TUM Campus Straubing for Biotechnology and Sustainability, Machine Learning for Sustainable Processes and Materials, Germany
[2] Technical University of Munich, TUM School of Computation, Information and Technology, Germany
ashima.khanna@tum.de, dominik.grimm@tum.de

Abstract

Optimizing protein sequences under tight oracle budgets requires methods that can explore a vast combinatorial mutational search space. These methods must also remain robust in the presence of surrogate noise and propose mutations on positions where substitutions are predicted to have favorable effects. We introduce SILO (**S**elf-improvement **I**mitation **L**earning **O**ptimization), a trajectory-level approach for oracle-budgeted protein design. SILO combines a hierarchical edit policy with incremental stochastic beam search without replacement (SBS) to explore diverse candidate trajectories. A UCB-based proxy ensemble, complemented by a proxy-derived alanine-scan-inspired fitness score (AFS), guides candidate selection for *in silico* oracle evaluation. The policy is then updated by next-action cross-entropy imitation on the round's best oracle-labeled trajectories. Across eight *in silico* protein fitness landscapes and five strong baselines from prior work, SILO achieves competitive performance within the evaluated oracle budget, often demonstrating faster early-stage improvement than other reproduced baselines. Two-sided paired t-tests show positive mean differences in favor of SILO across all eight landscapes for both maximum fitness and top-100 mean fitness, with statistically significant improvements after Holm correction on two and four landscapes, respectively. Under low-labeled data and noisy-proxy conditions, SILO shows competitive, task-dependent performance. Ablations identify SBS as the primary driver of performance, while AFS guidance and iterative imitation provide complementary, task-dependent benefits within the oracle-budgeted optimization loop. Code is available at: https://github.com/grimmlab/SILO.git

1 Introduction

Protein engineering enables the development of proteins with tailored functional properties, with applications ranging from therapeutics to enzyme engineering [34, 29, 23]. This engineering can be framed as a sequential decision making optimization problem under a strict evaluation budget, where the goal is to propose mutations to an initial (often wildtype) sequence that maximize a "fitness" function, quantifying a desired property such as thermostability, binding affinity, or catalytic efficiency. The problem is challenging due to the combinatorial size of the sequence space [13] and the prevalence of epistasis, which yields rugged, sparse, and highly non-convex fitness landscapes [10]. Locally plausible edits can produce highly non-additive outcomes, making efficient exploration critical.

A range of learning- and search-based methods address this problem. Reinforcement learning (RL) approaches optimize a generative policy via on-policy algorithms such as proximal policy optimization (PPO) [37] using learned reward models [2, 24]. Self-play approaches based on AlphaZero [39] combine policy-value networks with Monte Carlo Tree Search [21], enabling

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

structured exploration through look-ahead search [47, 25]. Generative methods such as GFlowNets [3] instead learn to sample diverse high-reward candidates via off-policy training [15] on existing datasets.

Despite these advances, sequence optimization under limited evaluation budgets faces three persistent challenges. First, value-based optimization can be unstable, sensitive to hyperparameters [37, 12], and prone to error propagation when value estimates are computed over noisy surrogate rewards [1]. Although off-policy generative methods improve stability [15], they remain sensitive to the quality and coverage of training data. This can lead to distributional shifts and poor generalization to unseen regions [38]. Moreover, their performance degrade under unreliable reward estimates when evaluating out-of-distribution generated candidates [17, 44]. Second, mutational proposals are typically position-agnostic, and edits at conserved positions can compromise protein function. Third, efficient exploration under strict evaluation budgets requires prioritizing high-value regions while avoiding redundant evaluations. Together, these challenges motivate approaches that reduce reliance on noisy surrogate estimates for policy learning, promote diverse exploration of mutation trajectories, and prioritize candidates with favorable substitution effects for oracle evaluation.

Recent works in neural combinatorial optimization (NCO) have explored self-improvement imitation learning (SIL) [27, 31, 7] as an alternative to value-based optimization. SIL iteratively improves a policy by supervised next-action prediction on trajectories that produced the highest-quality solutions found so far, relying on strong sampling procedures to generate candidates. Crucially, because the training signal comes from the actions of oracle-labeled top performers rather than from value estimates over a noisy surrogate, SIL sidesteps the value-propagation failures that destabilize RL under approximate rewards. While effective in settings such as routing problems with cheap evaluation, adapting SIL to oracle-budgeted settings with limited or noisy feedback is non-trivial. Evaluations must be allocated carefully, and the sampling procedure must surface candidates that are both promising under the surrogate and have mutations that do not disrupt protein function.

Main contributions Motivated by the challenges above, we introduce SILO, a trajectory-level protein sequence optimization approach that integrates diverse trajectory sampling, proxy-guided candidate selection, and iterative policy adaption. We evaluate SILO in an active learning setting under limited oracle budgets. Our contributions are:

1. **Hierarchical mutation search and trajectory-level learning.** SILO combines (i) a hierarchical edit policy that decomposes each mutation into a position choice followed by a residue choice, (ii) incremental stochastic beam search without replacement (SBS) [22] for diverse trajectory sampling, (iii) iterative policy adaptation from top oracle-evaluated trajectories using self-improvement imitation learning (SIL) [31]. Candidates are prioritized for oracle evaluation using fitness scores derived from UCB-based proxy ensemble, complemented by a proxy-derived alanine-scan-inspired fitness score (AFS) [20] as an auxillary signal.
2. **Sequence optimization under budgeted oracle calls.** We adapt these components into an iterative optimization process for protein design under strict oracle budgets within an active learning setup. Our ablations demonstrate that SBS drives the majority of the performance with AFS and SIL-based policy update providing additional, task-dependent adaptation.
3. **Empirical evaluation under standard and stress-test conditions.** Across eight *in silico* protein fitness landscapes and five reproduced baselines, SILO demonstrates strong performance within the evaluated oracle budget, frequently achieving faster early-stage improvement than other reproduced baselines. While paired mean differences favor SILO across all eight landscapes for maximum fitness and mean fitness of top-100 sequences, Holm-adjusted two-sided paired *t*-test identify statistically significant improvements on two landscapes for maximum fitness and four for top-100 mean fitness. Additional low-labeled data and noisy-proxy settings demonstrate competitive, task-dependent performance under challenging conditions.

2 Problem formulation

We consider protein sequence optimization over a discrete sequence space. Let $x \in \mathcal{V}^L$ denote a protein sequence of length L, where $\mathcal{V}$ is a finite vocabulary of $|\mathcal{V}| = 20$ (the standard amino acids). The objective is to maximize an oracle fitness function $O : \mathcal{V}^L \to \mathbb{R}$, measuring a desired property such as binding affinity, thermostability, or activity. We assume that oracle evaluations are expensive and only available through limited queries. The optimization proceeds in an active-

learning setting, following prior protein design works [2, 15]. We assume access to an initial dataset $\mathcal{D}_0 = \{(x_i, O(x_i))\}_{i=1}^{M_0}$ of $M_0 = |\mathcal{D}_0|$ oracle-labeled sequences, and initialize a starting sequence as $x_{\text{start}} = \text{argmax}_{x \in \mathcal{D}_0} O(x)$. At each active-learning round $n \in \{1, \ldots, N\}$, we select a batch $B_n \subset \mathcal{V}^L$ of K candidate sequences for oracle evaluation, where K is the per-round oracle query budget and each candidate differs from x_{start} by at most d_{max} mutations. After observing their oracle fitness values, the dataset is updated as $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{(x, O(x)) : x \in B_n\}$, giving a total of at most NK new oracle evaluations. Because direct oracle evaluation is limited, we train a proxy model f_ϕ on the currently available oracle-labeled dataset by minimizing

$$\mathcal{L}(\phi) = \mathbb{E}_{(x,O(x)) \sim \mathcal{D}_{n-1}} \left[(O(x) - f_\phi(x))^2 \right] \tag{1}$$

The proxy guides candidate selection before oracle evaluation, while final performance is always measured by the oracle O. The goal is to design sequences with higher fitness than the initial sequence within the budget NK.

3 Active learning with SILO

We follow an active learning loop, namely iterating over four steps, as illustrated in Figure 1 and outlined in Algorithm 1: (i) updating the proxy model f_ϕ by training on current dataset D_{n-1} by minimizing loss as shown in Equation 1; (ii) generating mutants using policy π_θ by sampling action trajectories using incremental SBS [22]; (iii) evaluating the selected candidates with the oracle and appending the new sequence-score pairs to the dataset; (iv) updating the policy π_θ by training on action trajectories derived from oracle-labeled top performing sequences using a batch-wise cross entropy objective.

We cast candidate generation as a Markov decision process (MDP) over edit trajectories, which allows policy training via cross-entropy on action sequences rather than value estimation over generated sequences. At each round we re-select the starting sequence x_{start} as the highest-oracle-scoring sequence in the current dataset $\mathcal{D}_{n-1}$, anchoring the search on the best candidate found so far. In the following subsections, we describe important components within SILO.

Algorithm 1: Active Learning with SILO.

Input: policy π_θ, proxy f_ϕ, oracle O, initial dataset D_0, oracle budget K, active learning rounds N, starting sequence x_{start}

for $n \leftarrow 1$ **to** N **do**
- Train f_ϕ on D_{n-1}: $\mathbb{E}_{(x,O(x)) \sim D_{n-1}}[(O(x) - f_\phi(x))^2]$
- Select starting sequence: $x_{start} \leftarrow \text{argmax}_{x \in \mathcal{D}_{n-1}} O(x)$
- SAMPLED $\leftarrow$ Sample mutated sequences from π_θ using incremental SBS with beam size β and mutational budget d_{max}
- Score $x \in$ SAMPLED using a defined objective function $S(x)$
- CANDIDATES $\leftarrow$ Select top K unique sequences from SAMPLED via greedy selection
- Evaluate CANDIDATES with oracle O and update dataset: $D_n \leftarrow \mathcal{D}_{n-1} \cup \{x, O(x)\}$: x $\in$ CANDIDATES
- BESTFOUND $\leftarrow$ top P sequences from CANDIDATES ranked by $O(x)$, together with their action trajectories
- **for** *each training step* **do**
  - Sample a batch of m history-next action pairs $(h^{(t)}, a^{(t)})$ uniformly from BESTFOUND trajectories
  - Update policy π_θ by minimizing the cross-entropy loss:
  - $\mathcal{L}(\theta) = -\frac{1}{m} \sum_{(h^{(t)}, a^{(t)})} \log \pi_\theta(a^{(t)} \mid h^{(t)})$

3.1 Markov decision process formulation

We formulate candidate optimization as a MDP, where an agent iteratively edits a protein sequence through a sequence of actions. A state $x^{(t)} \in \mathcal{V}^L$ corresponds to the edited sequence at step t, with $x_0 = x_{start}$ and $t \in \{0, \ldots, d_{\max} - 1\}$. At each step, the agent selects an action $a^{(t)}$ from a hierarchical action space $\mathcal{A} = \mathcal{A}_0 \times \mathcal{A}_1$, sampled in two stages:

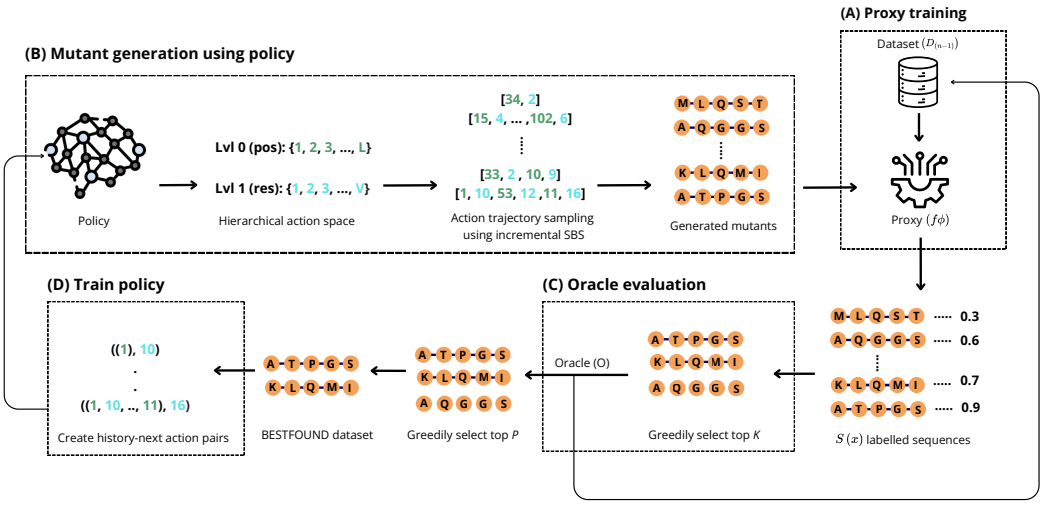

Figure 1: **Active learning with SILO.** We define the following four steps within our active learning loop: (A) A proxy ensemble model $f_\phi(x)$ is trained on the current dataset D_{n-1}. (B) The policy π_θ generates candidate mutants by sampling action trajectories (position then residue) with incremental stochastic beam search. The proxy scores the generated sequences with the objective $S(x)$ (Equation 5). (C) Top *K* candidates are selected for oracle evaluation. (D) The top P oracle-labeled sequences yield action trajectories used to train π_θ via next-action cross-entropy imitation. The loop repeats over N active learning rounds.

- **Level-0 action** $a_0^{(t)} \in \{1, ..., L\}$ chooses a position to modify.
- **Level-1 action** $a_1^{(t)} \in \mathcal{V}$ chooses a replacement amino acid, conditioned on the selected position $i = a_0^{(t)}$.

The environment transitions deterministically to a new state $x^{(t+1)}$, by substituting the amino acid at position i in $x^{(t)}$ with $a_1^{(t)}$. Each episode terminates after d_{max} steps, returning the final sequence $x^{(d_{max})}$, where d_{max} is the per-round mutational budget defined in Section 2. This hierarchical action space decomposes each mutation into a position choice followed by an amino acid choice, providing a factorized representation for sequential mutation trajectory generation.

3.2 Policy network architecture

Following the architecture used by GraphXForm for molecular design [32], we adapt it to the protein design space. Our policy is a lightweight transformer-based decision module (Figure 2). Given an input sequence x, we obtain contextualized per-residue embeddings from a frozen ESM Cambrian encoder [43], which encodes evolutionary priors. These embeddings are then processed by a stack of transformer layers [46] using FlashAttention [9] for memory efficiency, producing a latent representation $H \in \mathbb{R}^{(L \times D)}$, where D is the latent dimension.
Two learnable policy heads operate on H. A level-0 head h_0 produces logits over sequence positions in $\mathbb{R}^L$, and a level-1 head h_1 produces position-conditioned logits over the amino acid vocabulary in $\mathbb{R}^{(L \times |\mathcal{V}|)}$. The final residue distribution is obtained by selecting the row corresponding to the chosen position. This factorization reduces the joint action space of size $L \times |\mathcal{V}|$ to two decisions and naturally mirrors the hierarchical structure of the action space, enabling efficient parameter sharing while decomposing decision-making into position selection followed by amino acid selection.

3.3 Training algorithm

We train the policy via SIL [31], adapting it to the budgeted oracle setting. The key idea is to treat the policy's own highest fitness solutions as pseudo-expert demonstrations for cross-entropy imitation, without external supervision or value-function estimation.
Within each active learning round, the current policy π_θ samples a set of candidate trajectories of

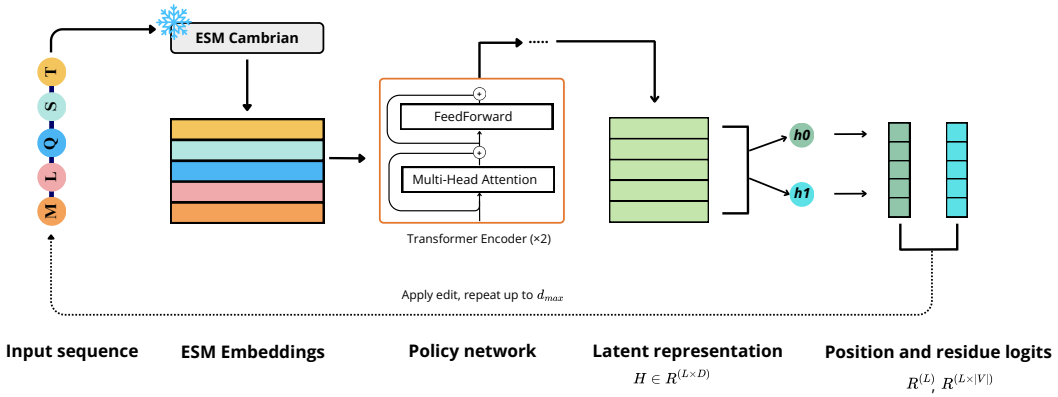

Figure 2: **Overview of the policy architecture.** An input sequence is encoded by a frozen ESM Cambrian model and the embeddings are processed by a transformer based policy. Two policy heads use latent representations from the policy to emit logits: h_0 over positions ($\mathbb{R}^L$) and h_1 over residues per position ($\mathbb{R}^{(L \times |\mathcal{V}|)}$). The policy is invoked iteratively over d_{max} edit steps, updating the input sequence after each step.

mutated sequences. Each trajectory $\tau = (a^{(0)}, ..., a^{(d_{max}-1)})$ corresponds to a sequence of actions that converts x_{start} into a final candidate $x^{d_{max}}$. The resulting candidate sequences are scored by the proxy under the objective $S(x)$ (Equation 5), and the top K unique sequences are evaluated by the oracle O. From these oracle-evaluated sequences, we select top P by oracle fitness as the BESTFOUND set, whose corresponding action trajectories serve as imitation targets.

We update π_θ on history-next action pairs $\{h^{(t)}, a^{(t)}\}$ drawn from BESTFOUND trajectories, where $h^{(t)} = (a^{(0)}, \ldots, a^{(t-1)})$ denotes the sequence of past actions taken until $t-1$ step and $a^{(t)}$ denotes the corresponding next action at step t. The policy is then trained by minimizing the cross-entropy loss. In practice, we optimize Equation 2 by uniformly sampling batches of history-next action pairs across BESTFOUND trajectories rather than processing full trajectories. The improved policy is then used to generate candidates in the next round. This creates a closed-loop self-improvement imitation learning process, in which the policy progressively concentrates probability mass on regions of the sequence space yielding higher oracle fitness. Unlike within the original applications of SIL [31], where calling an objective function for evaluation is cheap and the full candidate pool can be re-ranked, our oracle budget motivates a proxy-scored selection step (Section 3.5) that interposes between trajectory sampling and oracle evaluation.

$$\mathcal{L}(\theta) = -\mathbb{E}_{\tau \sim \text{BESTFOUND}} \left[\sum_{t=0}^{d_{max}-1} log \pi_\theta(a^{(t)} \mid h^{(t)}) \right] \tag{2}$$

3.4 Sampling

We sample diverse candidate trajectories using incremental SBS [22]. SBS is a stochastic variant of beam search that draws trajectories without replacement by perturbing log-probabilities with Gumbel noise (Equation10 in Appendix D.1). Starting from $x^{(0)} = x_{start}$, SBS expands partial action trajectories by sampling from π_θ. At each step, candidate continuations are ranked by perturbed log-probabilities and the top β trajectories are retained, forming a beam. To further improve efficiency and exploration, SIL adopts an incremental formulation of SBS, which allows sampling trajectories sequentially while dynamically updating the underlying distribution. After sampling a complete trajectory, its probability mass is removed from π_θ, and the remaining probabilities are renormalized, so that subsequent trajectories are drawn from the remaining probability space. This yields a diverse set of trajectories over the action space.

3.5 Objective function

Each candidate sequence x is scored using the proxy ensemble f_ϕ, which provides a mean prediction $\mu_\phi(x)$ and an uncertainty estimate $\sigma_\phi(x)$. The first scoring term reflects the predicted fitness of x

itself, computed as an upper-confidence-bound (UCB) acquisition function [41]:

$$\text{UCB}_{\text{proxy}}(x) = \mu_\phi(x) + \gamma_1 \sigma_\phi(x), \tag{3}$$

where γ_1 controls the exploration-exploitation factor. The second term incorporates an alanine-scan-inspired fitness signal (AFS), a proxy-derived score motivated by experimental alanine scanning [8, 16] and originally introduced by Kmicikiewicz et al. [20]. Experimental alanine scanning probes functionally relevant positions by substituting them with alanine, a neutral amino acid that perturbs side-chain interactions. Our AFS score does not rely on experimental alanine-substitution measurements but uses proxy predictions to provide an auxiliary signal of potential mutational tolerance. For a set of mutated positions M in x selected by the agent, we construct an alanine-substituted variant $A(x)$ by replacing all positions in M with alanine, and score it with the same proxy ensemble f_ϕ:

$$\text{UCB}_{\text{AFS}}(x) = \mu_\phi(A(x)) + \gamma_2 \sigma_\phi(A(x)), \tag{4}$$

where γ_2 controls the exploration-exploitation weight on the AFS term. A high predicted fitness of the alanine-substituted variant $A(x)$ suggests that the selected positions may tolerate neutral substitutions. However, this interpretation depends on the proxy accurately predicting the fitness of alanine-substituted variants and does not constitute direct evidence of functional tolerance. While prior work [20] uses AFS as a filter to identify non-disruptive substitutions, we instead use it as a complementary signal combined with the proxy UCB, allowing UCB score of alanine-substituted variants to contribute directly to candidate ranking.
The final candidate score is the unweighted sum of both terms:

$$S(x) = \text{UCB}_{\text{proxy}}(x) + \text{UCB}_{\text{AFS}}(x) \tag{5}$$

After scoring all sampled candidates, we greedily select the top K unique sequences for oracle evaluation.

4 Experiments

4.1 General setup

We examine the performance of SILO based on the following general setup as specified by [20], which serves as a basis of all subsequent experiments. The number of active learning rounds are $N = 10$, and the query budget to the oracle O per round is $K = 128$. The proxy f_ϕ architecture is an ensemble of three one-dimensional convolutional neural networks, as specified in [40]. Details on training f_ϕ per round can be found in Appendix B.2. These constraints remain consistent across all comparison partners. We set $\gamma_1 = 0.1$ and $\gamma_2 = 1$ in the objective $S(x)$ (Equation 5), directly adopting the corresponding UCB coefficients from ProSpero without task specific tuning. A sensitivity analysis in Table 32 further examines the effect of varying these coefficients on optimization performance. At each round, we instantiate $M = 5$ search instances with mutational budget $d_{max} \in \{1, 2, 3\}$, consistent with the low-order mutations typically done in experimental directed evolution to enable cumulative optimization. Full hyperparameters are reported in Appendix B.5

Protein engineering benchmarks We adopt the same eight benchmark datasets and evaluation protocol as used in ProSpero (see Appendix B.1) [20] for all methods. For AAV, we used ground-truth scores from FLEXS [40], while for all other tasks we follow Ren et al. [35] and use TAPE [33] as the *in silico* oracle O. Following Kim et al. [17] and Kmicikiewicz et al. [20], we replace the experimentally-measured fitness values in D_0 with oracle-derived scores. This isolates optimization performance from the regression error between the *in silico* oracle and the original experimental measurements, ensuring that the proxy is trained against the same target that the optimization algorithms are evaluated on.

Baselines We compare SILO against five baselines: (i) PEX and AdaLead, which are evolutionary algorithms [35, 40]; (ii) GFN-AL-δCS, which is an off-policy RL method with δ conservation parameter [17]; (iii) MLDE, a machine-learning-assisted directed evolution approach [45]; (iv) ProSpero, a biologically constrained search method based on inference-time guidance of a pre-trained generative model [20]. We focus our analysis on the top baselines from the "Fitness optimization" experiment in original ProSpero publication, selected based on their overall performance. For completeness, we report both our reproduced results and the originally reported performance from ProSpero in Tables 1 and 33, respectively.

Evaluation metrics We report four metrics, with primary emphasis on (i) **maximum fitness**, derived from the sequence with the highest fitness attained amongst all evaluated sequences, and (ii) **mean fitness**, computed as the average fitness of the top-100 evaluated sequences across all active learning rounds. We additionally report (iii) **novelty**, the average Hamming distance between top-100 oracle-evaluated sequences and the wildtype sequence, and (iv) **diversity**, the average pairwise Hamming distance within the top-100 oracle sequences. Novelty and diversity characterize the exploration profile of each method rather than its optimization quality.

4.2 Experiment 1: Sequence optimization task

Setup We evaluate all methods under the active learning setup as described in subsection 4.1. In this experiment, we focus on the core objective of protein fitness optimization: identifying high-fitness variants under a fixed oracle budget. We compare methods using the maximum oracle fitness and mean fitness of top-100 sequences across all rounds, measuring both peak performance and the quality of the best generated candidate set.

Results Across eight protein fitness landscapes and five reproduced baselines, SILO achieves the highest observed maximum and top-100 mean fitness on all eight landscapes within the evaluated oracle budget (Tables 1, 4). As shown in Figure 3 and Table 10, SILO often discovers high-fitness sequences in earlier rounds, indicating a more efficient use of the available oracle budget.
To assess the statistical support of these improvements, we conduct two-sided paired *t*-test over five aligned random seeds, comparing SILO against the strongest reproduced baseline selected separately for each landscape and metric. For maximum fitness, the paired differences favor SILO on all eight landscapes, with statistically significant improvements after Holm correction on AAV and AMIE (Table 8). UBE2I also exhibits a positive paired difference and a significant unadjusted p-value, but does not remain significant after correction. For top-100 mean fitness, all eight paired differences favor SILO, with statistically significant improvements on AAV, AMIE, TEM, and UBE2I after Holm correction (Table 9). These findings indicate strong performance within the evaluated oracle budget, although the statistical evidence varies across landscapes and metrics.
We additionally evaluate sequence novelty relative to the wildtype sequence and diversity within the top-100 candidate set. SILO achieves the highest novelty on TEM and the second-highest on five other landscapes (Table 5). However, its top-100 candidate set exhibit lower mean diversity than other comparison partners across all eight landscapes (Table 6), indicating that gains in observed fitness do not necessarily coincide with greater within-set diversity.
Our primary comparisons use reproduced baselines evaluated under a consistent experimental protocol, enabling controlled, seed-aligned comparisons. We further report originally published ProSpero results in Table 33 under Appendix B.3, noting differences, particularly in the scores obtained for Pab1 for AdaLead.

Table 1: Maximum fitness attained by all methods on 8 benchmark datasets across 10 active learning rounds. Reported values represent the mean ± standard deviation over 5 independent runs. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task. Underline: second-best

Method	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
PEX	0.656 ± 0.009	0.245 ± 0.004	8.049 ± 0.037	3.604 ± 0.037	0.035 ± 0.000	1.799 ± 0.324	**1.233 ± 0.002**	2.991 ± 0.002
AdaLead	0.697±0.011	0.236±0.009	8.034±0.062	3.585±0.002	0.037±0.002	1.653±0.165	1.229±0.001	2.987±0.002
GFN-AL-δCS	0.681±0.015	0.228±0.003	8.052±0.055	3.593±0.004	0.031±0.003	1.592±0.327	1.228±0.000	2.991±0.000
MLDE	0.551 ± 0.001	0.237 ± 0.002	7.869 ± 0.035	3.586 ± 0.002	0.025 ± 0.001	0.936 ± 0.001	1.228 ± 0.000	2.985 ± 0.005
ProSpero	0.676 ± 0.012	0.242 ± 0.005	8.143 ± 0.021	3.615 ± 0.004	0.043 ± 0.001	1.653 ± 0.404	1.230 ± 0.001	2.991 ± 0.002
SILO	**0.747 ± 0.014**	**0.262 ± 0.004**	**8.169 ± 0.052**	**3.619 ± 0.002**	**0.045 ± 0.003**	**1.894 ± 0.348**	**1.233 ± 0.003**	**2.997 ± 0.003**

4.3 Experiment 2: Low-labeled data regime

Setup We evaluate all methods under the active learning setup as explained in Section 4.1, but reduce the size of the initial labeled dataset D_0. Specifically, we randomly subsample 10%, 20%, and 50% of D_0, using the same subsample per seed across all methods to ensure identical proxy training data. All other experimental settings are held fixed. We conduct this targeted stress test on UBE2I, TEM, LGK, and Pab1, with the goal to assess performance under reduced initial labeled data in settings where training data for proxy might be limited. We report performance with both primary

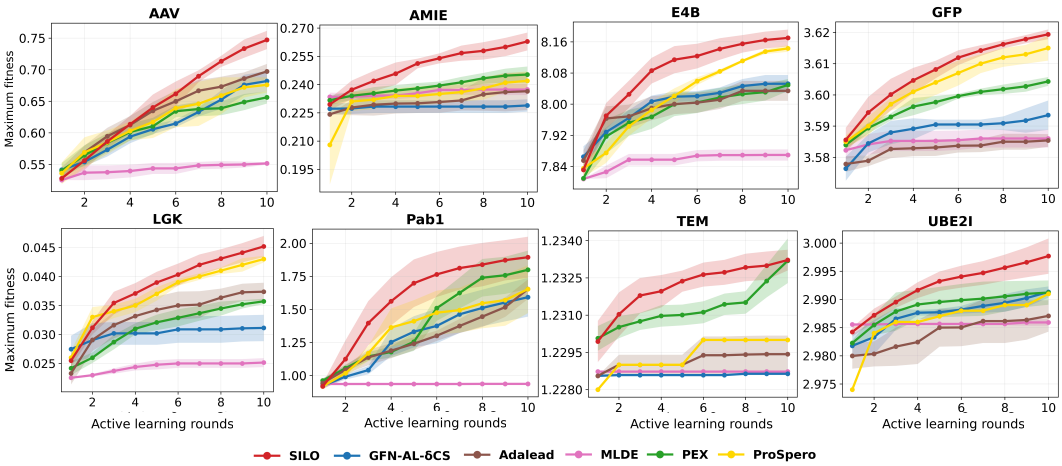

Figure 3: **Active learning performance across eight protein fitness landscapes.** Higher oracle fitness indicates better performance. All methods improve over wildtype sequence fitness $O(x_{\mathrm{WT}})$; however, SILO consistently achieves the highest maximum fitness compared to all baselines. High quality sequences are discovered in earlier rounds, indicating more efficient use of the limited oracle budgets. Shaded regions denote standard deviation over five runs.

and secondary metrics (Tables 12- 17), evaluating the ability of each method to identify high-quality sequences despite reduced training data.

Results Figure 4 compares maximum fitness across four protein fitness landscapes under varying fractions of the initial labeled dataset. SILO achieves the highest observed mean maximum fitness on LGK across all evaluated data fractions and performs competitively on UBE2I, where differences between the strongest methods remain small. On TEM, several methods achieve comparable fitness, particularly at higher data fractions, with SILO remaining among the strongest performers. Performance on Pab1 is more variable: SILO is competitive at lower data fractions and achieves the highest observed mean fitness with the full initial dataset, while other methods lead at intermediate fractions. Overall, these results indicate that SILO maintains competitive performance under the evaluated low-data conditions, although its relative advantage depends on the protein landscape and the amount of initial labeled data.

4.4 Experiment 3: Noisy proxy setting

Setup In this experiment, we investigate the performance of different methods under degraded proxy quality by deliberately corrupting the proxy model, while keeping other experimental details constant. We replace the proxy f_ϕ with an ensemble of noisy oracles f_ϵ, obtained by adding zero-mean Gaussian noise to ground-truth oracles, following the inference scheme of [40]. The noise magnitude is governed by the signal-to-noise ratio (SNR), with $\delta_{noise} = \sqrt{Var(D_0)} \times 10^{-SNR/10}$, where $Var(D_0)$ denotes the variance of fitness values in the initial dataset (D_0). We examine three levels of surrogate noise, namely -25, -15, and -5, on AMIE, E4B, GFP and AAV.

Results Figure 5 reports maximum fitness across four protein fitness landscapes under varying levels of proxy noise. Importantly, noise is introduced only into proxy predictions used for candidate ranking, while oracle evaluations remain noise-free. SILO achieves the highest observed mean fitness across all tested noise levels on AMIE, GFP, and AAV, and on three of the four noise levels on E4B, while remaining competitive with ProSpero. On AMIE and GFP, SILO's performance varies only modestly as noise increases, while on AAV and E4B, fitness decreases more noticeably under stronger noise but remains competitive with or above the strongest baselines. Several competing methods exhibit larger performance reductions under increased noise, although these trends vary across landscapes. Overall, these results suggest that SILO maintains strong performance under the evaluated synthetic proxy-noise conditions, with task-dependent sensitivity to increasing noise.

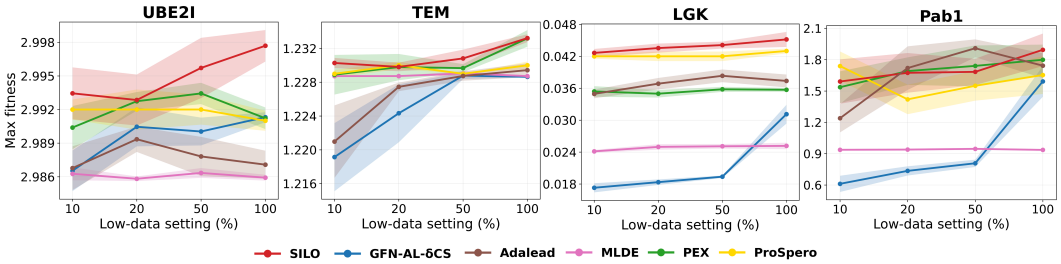

Figure 4: **Performance under limited initial training data.** Maximum fitness achieved across four protein fitness landscapes (UBE2I, TEM, LGK, and Pab1) with varying fractions of the initial training dataset (D_0) (10%, 20%, 50%, and 100%). SILO remains competitive across the evaluated data fractions, with relative performance varying across landscapes and data availability.

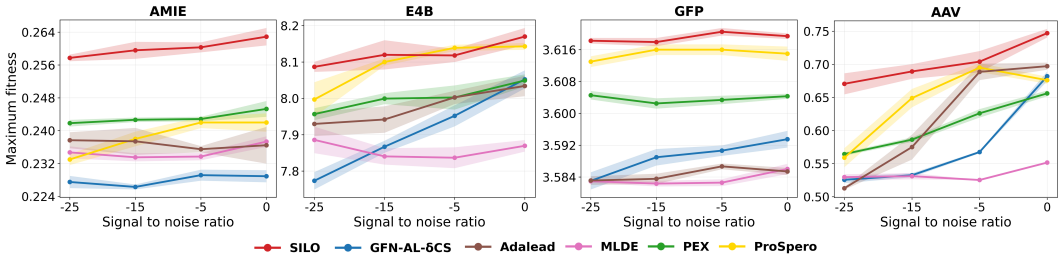

Figure 5: **Performance under noisy proxy setting.** Maximum fitness achieved across four protein fitness landscapes (AMIE, E4B, GFP, and AAV) under varying signal-to-noise ratios (-25, -15, -5, 0). SILO maintains competitive performance across the evaluated noise levels, achieving the highest observed mean fitness in most settings, with task-dependent differences relative to the strongest baselines.

However, these results do not establish robustness to noisy oracle labels, which is particularly important for SILO because policy updates are based on top oracle-evaluated trajectories.

4.5 Experiment 4: Ablation studies

Setup We perform ablations to investigate the contributions of (i) sequence generation strategy, (ii) sampling and selection strategy, and (iii) policy learning. We first compare the fitness-guided starting-sequence selection and policy-based mutation generation against a random mutation baseline that randomly selects starting sequences, mutation positions, and amino acid substitutions. Both variants use the same Equation 5 scoring function for candidate prioritization. Next, we compare sampling and selection strategies by replacing SBS with standard beam search (BS) and by removing the AFS term, retaining only the proxy UCB score in Equation 3 instead of the complete scoring function in Equation 5. Finally, to quantify the impact of learning, we evaluate a randomly initialized frozen-policy variant, where policy parameters are not updated across active learning rounds. Table 2 reports these ablations on all eight landscapes. Complete ablation results are provided in Tables 24-27.

Table 2: Results of ablation studies across all eight landscapes, reporting the maximum fitness achieved over five independent runs across 10 active learning rounds. The ablations analyze the contributions of sequence generation strategy, sampling and selection strategy, and policy learning. **Bold**: the best per each task.

Ablation	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
Random baseline	0.530±0.015	0.229±0.002	6.970±1.156	3.600±0.010	0.027±0.001	1.018±0.103	1.229±0.000	2.984±0.000
SIL + BS + AFS	0.587±0.028	0.232±0.007	7.706±0.173	3.602±0.012	0.026±0.001	1.022±0.146	1.228±0.000	2.987±0.001
SIL + SBS w/o AFS	0.694±0.019	0.258±0.003	8.140±0.062	3.615±0.001	0.042±0.001	1.558±0.134	1.232±0.000	**2.999±0.001**
Frozen + SBS + AFS	0.706±0.013	0.256±0.003	8.149±0.041	3.615±0.001	0.042±0.001	1.530±0.268	**1.233±0.002**	2.997±0.001
SILO	**0.747±0.014**	**0.262±0.004**	**8.169±0.052**	**3.619±0.002**	**0.045±0.003**	**1.894±0.348**	**1.233±0.003**	2.997±0.003

Results Table 2 summarizes the ablation results across all eight protein fitness landscapes. The random mutation baseline achieves a lower maximum fitness than SILO does across all eight landscapes, despite using the same candidate-ranking objective. This suggests that the choice of starting sequences and mutation proposals contributes to optimization performance beyond the ranking function alone. Among the individual components, SBS provides the strongest and most consistent performance gains. Replacing SBS with BS, while retaining AFS and policy learning, shows statistically significant drop in performance on all eight landscapes after Holm correction (Table 29). In contrast, AFS provides a smaller, complementary contribution: adding AFS to BS improves the observed mean maximum fitness on six landscapes and leaves it unchanged at the reported precision on the remaining two compared to the variant with BS and without AFS (Table 24). Iterative policy learning yields additional, task-dependent benefits. Compared with the frozen-policy SBS variant, full SILO achieves statistically significant improvements after Holm correction on one task, while the remaining landscapes do not reach the corrected significance threshold (Table 28). Overall, these results identify SBS as the primary driver of performance gains, with AFS and SIL based policy adaptation providing complementary benefits whose magnitude varies across tasks.

5 Conclusion

We presented **SILO**, a trajectory-level protein optimization approach that combines self-improvement imitation learning for policy updates, incremental stochastic beam search for diverse trajectory sampling, and proxy-guided candidate selection complemented with an alanine-scan-inspired fitness score. These components form an iterative optimization procedure that leverages top oracle-evaluated trajectories to improve the policy under limited oracle budgets.
Across eight protein benchmark tasks, SILO achieves a strong performance within the evaluated oracle budget, often exhibiting faster early-stage improvement than reproduced baselines. Paired statistical tests with Holm correction identify significant improvements on two landscapes for maximum fitness and on four landscapes for top-100 mean fitness, with positive observed paired differences across all eight landscapes for both metrics. Ablations highlight SBS as the core component for performance gains, while AFS and SIL-based policy adaptation provide complementary, task-dependent benefits. Additional experiments under low-data and noisy-proxy conditions demonstrate competitive performance across the evaluated settings.
Future work should investigate the use of SILO with larger mutation budgets, sensitivity to noisy oracle evaluations, and experimental validation beyond *in silico* fitness oracles.

Funding and acknowledgments

This work was supported by the Pythagoras Project (Grant No. 031B1407B) within the program "Climate-neutral products through biotechnology–CO_2 and C1 compounds as sustainable raw materials for the industrial bioeconomy (CO2BioTech)", which is funded by the Bundesministerium für Forschung, Technologie und Raumfahrt (BMFTR). The Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) further supported this work through Project No. 466387255 within the Priority Programme "SPP 2331: Machine Learning in Chemical Engineering". The authors gratefully acknowledge the Competence Center for Digital Agriculture (KoDA) at the University of Applied Sciences Weihenstephan-Triesdorf for providing computational resources.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] 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.

[3] 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.

[4] Shimon Bershtein, Michal Segal, Roy Bekerman, Nobuhiko Tokuriki, and Dan S Tawfik. Robustness–epistasis link shapes the fitness landscape of a randomly drifting protein. *Nature*, 444(7121):929–932, 2006.

[5] 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.

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

[7] Andrea Corsini, Angelo Porrello, Simone Calderara, and Mauro Dell'Amico. Self-labeling the job shop scheduling problem. *Advances in Neural Information Processing Systems*, 37:105528–105551, 2024.

[8] Brian C Cunningham and James A Wells. High-resolution epitope mapping of hgh-receptor interactions by alanine-scanning mutagenesis. *Science*, 244(4908):1081–1085, 1989.

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

[10] J. Arjan G.M. de Visser and Joachim Krug. Empirical fitness landscapes and the predictability of evolution. *Nature Reviews Genetics*, 15(7):480–490, 2014. ISSN 1471-0064. doi: 10.1038/nrg3744. URL `http://dx.doi.org/10.1038/nrg3744`.

[11] Elad Firnberg, Jason W Labonte, Jeffrey J Gray, and Marc Ostermeier. A comprehensive, high-resolution map of a gene's fitness landscape. *Molecular biology and evolution*, 31(6):1581–1592, 2014.

[12] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[13] J D Hermes, S C Blacklow, and J R Knowles. Searching sequence space by definably random mutagenesis: improving the catalytic potency of an enzyme. *Proceedings of the National Academy of Sciences*, 87(2): 696–700, January 1990. ISSN 1091-6490. doi: 10.1073/pnas.87.2.696. URL `http://dx.doi.org/10.1073/pnas.87.2.696`.

[14] Hervé Jacquier, André Birgy, Hervé Le Nagard, Yves Mechulam, Emmanuelle Schmitt, Jérémy Glodt, Beatrice Bercot, Emmanuelle Petit, Julie Poulain, Guilène Barnaud, et al. Capturing the mutational landscape of the beta-lactamase tem-1. *Proceedings of the National Academy of Sciences*, 110(32): 13067–13072, 2013.

[15] 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.

[16] Sanjeevaiah Kasturi, Ako Kihara, David FitzGerald, and Ira Pastan. Alanine scanning mutagenesis identifies surface amino acids on domain ii of pseudomonas exotoxin required for cytotoxicity, proper folding, and secretion into periplasm. *Journal of Biological Chemistry*, 267(32):23427–23433, 1992.

[17] 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. *arXiv preprint arXiv:2410.04461*, 2024.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Justin R Klesmith, John-Paul Bacik, Emily E Wrenbeck, Ryszard Michalczyk, and Timothy A Whitehead. Trade-offs between enzyme fitness and solubility illuminated by deep mutational scanning. *Proceedings of the National Academy of Sciences*, 114(9):2265–2270, 2017.

[20] Michal Kmicikiewicz, Vincent Fortuin, and Ewa Szczurek. Prospero: Active learning for robust protein design beyond wild-type neighborhoods. *arXiv preprint arXiv:2505.22494*, 2025.

[21] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[22] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International conference on machine learning*, pages 3499–3508. PMLR, 2019.

[23] HA Daniel Lagassé, Aikaterini Alexaki, Vijaya L Simhadri, Nobuko H Katagiri, Wojciech Jankowski, Zuben E Sauna, and Chava Kimchi-Sarfaty. Recent advances in (therapeutic protein) drug development. *F1000Research*, 6:113, 2017.

[24] 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. *arXiv preprint arXiv:2405.18986*, 2024.

[25] Huitian Lin, Cheng Zhu, Tianfeng Shang, Ning Zhu, Kang Lin, Chengyun Zhang, Xiang Shao, Xudong Wang, and Hongliang Duan. Highplay: Cyclic peptide sequence design based on reinforcement learning and protein structure prediction. *Journal of Medicinal Chemistry*, 68(11):12047–12057, 2025.

[26] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, 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*, 2022:500902, 2022.

[27] Fu Luo, Xi Lin, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Self-improved learning for scalable neural combinatorial optimization. *arXiv preprint arXiv:2403.19561*, 2024.

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

[29] Giles Obinna Ndochinwa, Qing-Yan Wang, Nkwachukwu Oziamara Okoro, Oyetugo Chioma Amadi, Tochukwu Nwamaka Nwagu, Chukwudi Innocent Nnamchi, Anene Nwabu Moneke, and Arome Solomon Odiba. New advances in protein engineering for industrial applications: Key takeaways. *Open life sciences*, 19(1):20220856, 2024.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[31] Jonathan Pirnay and Dominik G Grimm. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *Transactions on Machine Learning Research (TMLR)*, 2024. doi: 10.48550/arxiv.2403.15180. URL `https://openreview.net/forum?id=agT8ojoH0X`. Awarded with "Featured" certification.

[32] Jonathan Pirnay, Jan G Rittig, Alexander B Wolf, Martin Grohe, Jakob Burger, Alexander Mitsos, and Dominik G Grimm. Graphxform: graph transformer for computer-aided molecular design. *Digital Discovery*, 4(4):1052–1065, 2025.

[33] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

[34] Julia C. Reisenbauer, Kathleen M. Sicinski, and Frances H. Arnold. Catalyzing the future: recent advances in chemical synthesis using enzymes. *Current Opinion in Chemical Biology*, 83:102536, 2024. ISSN 1367-5931. doi: https://doi.org/10.1016/j.cbpa.2024.102536. URL `https://www.sciencedirect.com/science/article/pii/S1367593124001121`.

[35] 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.

[36] 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.

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

[38] Kajetan Schweighofer, Markus Hofmarcher, Marius-Constantin Dinu, Philipp Renz, Angela Bitto-Nemling, Vihang Prakash Patil, and Sepp Hochreiter. Understanding the effects of dataset characteristics on offline reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021. URL `https://openreview.net/forum?id=A4EWtf-TO3Y`.

[39] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[40] 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.

[41] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

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

[43] ESM Team et al. Esm cambrian: Revealing the mysteries of proteins with unsupervised learning. *EvolutionaryScale Website*, 2024.

[44] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021.

[45] Thanh VT Tran and Truong Son Hy. Protein design by directed evolution guided by large language models. *IEEE Transactions on Evolutionary Computation*, 29(2):418–428, 2024.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] 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.

[48] Jochen Weile, Song Sun, Atina G Cote, Jennifer Knapp, Marta Verby, Joseph C Mellor, Yingzhou Wu, Carles Pons, Cassandra Wong, Natascha van Lieshout, et al. A framework for exhaustively mapping functional missense variants. *Molecular systems biology*, 13(12):MSB177908, 2017.

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

Appendix

A Related work **15**

B Experiment setting and implementation details **15**

B.1 Benchmark datasets 15

B.2 Proxy architecture and training 16

B.3 Baselines 16

B.4 Protein sequence representations 17

B.5 Hyperparameters for SILO 17

B.6 Evaluation metrics 17

C Discussion **17**

C.1 Hardware and runtime comparisons 17

C.2 Limitations 18

C.3 Broader impact 18

D Algorithmic details **18**

D.1 Incremental stochastic beam search 18

E Complete results **19**

E.1 Sequence optimization task 19

E.2 Statistical significance tests 19

E.3 Early-round protein design 20

E.4 Low-labeled data regime 21

E.5 Noisy proxy setting 22

E.6 Ablation studies 24

E.7 Hyperparameter sensitivity analysis 25

E.8 Results reported from ProSpero 27

A Related work

Search and surrogate-guided generative methods Adapt-with-the-Leader **(AdaLead)** is a model-guided method that iteratively refines sequences by greedily applying mutations informed by a learned proxy, following the evolutionary and follow the perturbed leader approaches [40]. Proximal Exploration (**PEX**), introduced by Ren et al., emphasizes conservative exploration by prioritizing lower-order mutants and close to the wildtype sequence, framing the problem of local search as proximal optimization [35]. Within Machine Learning-Driven Evolution **(MLDE)** framework, Tran and Hy leveraged ESM-2 protein language model to generate mutants through random and importance-based masking of the wild-type sequence [45]. Lastly, **ProSpero** integrates proxy guidance with inference-time control of pretrained generative model, EvoDiff, using targeted masking and biologically constrained sampling to generate candidates without updating any learned parameters of EvoDiff across rounds [20].

Reinforcement learning **DyNA-PPO** [2] applies model-based RL to generate sequences using PPO [37] to guide the policy and using a learned proxy for generating high-fitness candidates. **LatProtRL** extends this paradigm by performing optimization in the latent space of ESM-2 model. A PPO-based policy is trained to navigate this latent space, enabling smoother exploration [24]. **EvoPlay** [47] adopts a self-play RL framework inspired by AlphaZero [39], combining a policy-value network with Monte Carlo Tree Search [21] to iteratively refine candidate sequences.

Generative and distribution-based models Methods like GFlowNets [3], learn a stochastic policy by combining offline datasets with on-policy samples during training to generate high fitness candidates [15]. While they improve stability over methods like DyNA-PPO, they can be sensitive to proxy misspecification [17, 44]. Building on this, Kim et al. introduced GFN-AL-δCS, which utilizes conservativeness parameter δ to balance novelty and robustness [17]. Authors of DbAS iteratively refine a generative model, i.e. variational autoencoder, to high fitness sequences under a surrogate model, effectively concentrating probability mass on promising regions [6]. CbAS extends DbAS by adding distributional regularization to mitigate surrogate bias and improve stability [5].

Difference between SILO and related methods SILO differs from above methods in several key ways. First, unlike DbAS/CbAS and GFlowNets/GFN-AL-δCS, SILO operates in action trajectory space rather than sequence space, decomposing the problem into finer-grained decisions, rather than modeling the sequence as a whole, enabling more structured and targeted exploration of the mutational space. Second, while both SILO and ProSpero use an alanine-scan inspired score to provide feedback on disruptive positions, ProSpero relies on a pretrained generative model that is never updated after deployment. In contrast, SILO learns a reusable editing policy from top oracle-evaluated trajectories, accumulating task-specific knowledge about which positions and substitutions lead to fitness gains across rounds. Finally, unlike EvoPlay and other PPO-based approaches that employ a critic, SILO requires neither a value function nor a critic network. Instead, the policy is trained via cross-entropy loss on action trajectories of the highest-fitness oracle-evaluated mutants from each round, without directly relying on proxy-predicted fitness as a training target.

SILO vs current applications of SIL The SIL paradigm from Pirnay and Grimm [31], and its applications to routing problems [31] and molecular design [32], trains a policy via supervised cross-entropy on action trajectories that produced the best discovered solutions. Importantly, both settings assume cheap and unlimited evaluations from the objective function. Our setting, however, imposes a strict oracle budget per round, making effective candidate prioritization essential and motivating the combination of proxy-based acquisition with an auxiliary alanine-scan-inspired fitness signal in SILO.

B Experiment setting and implementation details

B.1 Benchmark datasets

SILO was evaluated on eight diverse benchmark protein fitness landscapes. As mentioned in [20], the AAV and GFP benchmarks were originally introduced by Kim et al. (Apache-2.0 license) [17], while the remaining datasets were compiled by Tran and Hy [45] (GPL-3.0 license). Oracles are taken from FLEXS [40] for AAV and GFP (Apache-2.0 license), and from Ren et al. [35] for the other tasks (Apache-2.0 license).

1. **Adeno-associated Virus (AAV):** The task focuses on improving binding affinity of an amino acid segment (position 450-540) of the VP1 protein located in the capsid of the Adeno-associated virus. The dataset D_0 containing 15,307 sequences, was generated by Kim et al. [17] by randomly mutating the wild-type sequence while filtering out sequences that have higher scores than the wild-type. Here, the sequence length $L = 90$, corresponding to search space of 20^{90}. The starting fitness is the wildtype sequence is $O(x_{WT}) = 0.500$ and its average Hamming distance to sequences in D_0 is Novelty (D_0, x_{WT}) = 5.05.

2. **Aliphatic Amide Hydrolase (AMIE):** The task aims to optimize amidase sequences for increased enzymatic activity [49]. Initial dataset D_0 contains 6417 sequences with single mutations to model the fitness landscape. The length of sequence $L = 341$. The starting fitness is $O(x_{WT}) = 0.224$, and Novelty (D_0, x_{WT}) = 2.
3. **Ubiquitination Factor Ube4b (E4B):** The goal is generate sequences that enhance E4B ubiquitination enzyme. Starita et al. [42] measured the rates of ubiquitination of the mutants to the target protein. The full dataset includes 91,032 sequences of length $L = 102$, from which [20] randomly selected 10,000 sequences to form D_0. The starting sequence has $O(x_{WT}) = 7.743$ and Novelty (D_0, x_{WTt}) = 5.42.
4. **Green Fluorescent Protein (GFP):** The task seeks to generate sequences with high log-fluorescence intensity [36]. The D_0 dataset contains 10,200 sequences generated by [17], similarly to AAV. The sequence length is L =238, with $O(x_{WT}) = 3.572$ and Novelty (D_0, x_{WT}) = 42.87.
5. **Levoglucosan Kinase (LGK):** This task aims to improve enzymatic activity of *levoglucosan kinase*, which converts LG to the glycolytic intermediate glucose-6-phosphate [19]. The initial dataset D_0 includes 7,633 sequences of length $L = 439$, with $O(x_{WT}) = 0.020$ and Novelty (D_0, x_{WT}) = 2.0.
6. **Poly(A)-binding Protein (Pab1):** The poly(A)-binding protein Pab1 binds to polyadenosine (poly-A) sequences via its RNA recognition motif (RRM). Melamed et al. [28] conducted a high-throughput screening assay to measure binding fitness for approximately 36,000 double mutants of Pab1 within the RRM region. This task focuses on improving binding affinity within the RRM. As with E4B, [17] subsampled 10,000 sequences to construct D_0. The length of the sequence is $L = 75$, the starting sequence has $O(x_{WT}) = 0.843$ and Novelty (D_0, x_{WT}) = 3.95.
7. **TEM-1 β-Lactamase (TEM):** TEM-1 β-Lactamase resistance to penicillin antibiotics in *E.coli* is widely studied to understand mutational effect and fitness landscape [14], [4]. The aim of this task is to generate TEM variants with improved thermodynamic stability. The dataset D_0 derived [11] from contains 5,199 sequences of length L =286. The starting sequence has fitness $O(x_{WT}) = 1.229$ and Novelty (D_0, x_{WT}) = 2.0.
8. **SUMO E2 Conjugase (UBE2I):** Variants of the disease-relevant protein, human SUMO E2 conjugase were generated by Weile et al. [48] and the goal of this task is to optimize these variants for functional mapping applications. D_0 comprise of 3,022 sequences of length L =159. The starting sequence has $O(x_{WT})$ =2.978 and Novelty (D_0, x_{WT}) = 2.0.

B.2 Proxy architecture and training

For the proxy model, we use an ensemble of one-dimensional convolutional neural network [20, 40]. The ensemble size is assigned to 3, and the model is trained using the Adam optimizer [18] with learning rate and L2 penalty set to 0.0001. We do a random 90%-10% train-validation data split. The batch size is set to 256. We train the ensemble for 3000 epoches. To mitigate overfitting, we use early stopping based on a validation set comprising 10% of the data, stopping training if the validation loss does not improve for ten consecutive evaluations.

B.3 Baselines

We reproduce the top five comparison partners from the "Fitness optimization" experiment in ProSpero by rerunning them under the same experimental conditions, including identical hyperparameter configurations and seeds as specified in the original work [20]. All methods are evaluated under a consistent experimental protocol, including the same datasets, oracle implementation, query budget, and proxy architecture. Despite these efforts, some reproduced results differ from those originally reported. We therefore use the reproduced results for our primary comparisons and report the originally published results separately for reference (Tables 33, 34). All comparison partners are operated under active learning rounds $N = 10$, number of queries to the oracle per round $K = 128$, and the same proxy architecture.

1. **AdaLead [40]:** We employed the open-source implementations provided by Sinai et al. [40] available at `https://github.com/samsinai/FLEXS/tree/master` under the Apache-2.0 license. We used the default hyperparameters of the model, with a recombination rate of 0.2, a mutation rate of 1/L, where L is the sequence length, and a threshold $\tau = 0.05$. The numbers of model predictions is set to 2000.
2. **GFN-AL-δCS [17]:** For the conservative strategy GFN-AL-δCS proposed by Kim et al. [17], we employ an adaptive δ with a maximum masking radius of 0.05 and rank-based proxy training with a reweighting factor k=0.01. We set the scaling factor λ =0.1 for AAV, E4B, and Pab1, and λ =1 for GFP, AMIE, TEM, UBE2I, and LGK. We use the publically released codebase from `https://github.com/hyeonahkimm/delta_cs.git` under the Apache-2.0 license.
3. **MLDE [45]:** We use the MLDE implementation from Tran and Hy [45], from their publicly available codebase `https://github.com/HySonLab/Directed_Evolution` under the GPL-3.0 license. Adapting their setup to perform active learning, we perform 10 rounds of surrogate-guided optimization

with a population size of 128 and a beam size of 4. The masking strategy employs a random-to-importance ratio of 0.6:0.4, and unmasking is performed using ESM-2 [26] with 35M parameters.

4. **PEX [35]:** We implement PEX from Ren et al. [35] using the official codebase `https://github.com/HeliXonProtein/proximal-exploration/tree/main` under the Apache-2.0 license. We use the default configuration, including 2 random mutations and a frontier neighborhood size of 5.
5. **ProSpero [20]:** We use ProSpero under the official codebase `https://github.com/szczurek-lab/ProSpero.git` released by Kmicikiewicz, Fortuin, and Szczurek [20] under the GPL-3.0 license. We use the default hyperparameters of the method, keeping the range of number of corruptions introduced within the sequences between 3-10 for shorter sequences (E4B, Pab1, AAV) and 5-15 for longer sequences (GFP, LGK, AMIE, TEM, UBE2I). Number of scans (S) for targeted masking is set to 16 and sequential monte carlo (SMC) batch size is 256. The UCB exploitation-exploration hyperparamaters for targeted masking and biologically-constrained SMC are $k = 1.0$ and 0.1, respectively. Additionally, we run experiments under fully deterministic setting to reinforce reproducibility.

B.4 Protein sequence representations

To compute sequence embeddings, we use a frozen ESM Cambrian [43], a pretrained protein language model with 300M parameters, available under Cambrian Open License Agreement.

B.5 Hyperparameters for SILO

At each round, we instantiate $M = 5$ mutation-constrained search instance with the budget of 1–3 mutations and apply SBS to sample diverse trajectories. The beam size β is 32 and the total sequences generated for proxy evaluation per round are 480. We use $P = 50$ top oracle-evaluated candidates per round for training the policy. The batch size for training is 16. The number of transformer blocks for the policy are 2, number of heads are 16, and latent dimension R_D is 512. We train the policy using Adam optimizer [18] with a learning rate of 0.0001.

B.6 Evaluation metrics

Considering $\mathcal{D}_{\text{top}} = \{(x^{(i)}, O(x^{(i)}))\}_{i=1}^{100}$ contains the set of the top-100 oracle evaluated sequences identified across N active learning rounds. We evaluate all methods using the following metrics:

1. **Maximum fitness:** The highest oracle fitness attained among the evaluated sequences, measuring the ability to identify high-performing candidates:

$$\text{MaxFitness}(\mathcal{D}_{\text{top}}) = \max_{x \in \mathcal{D}_{\text{top}}} O(x) \tag{6}$$

2. **Mean fitness:** The average oracle fitness of the top-100 evaluated sequences, characterizing the quality of the highest-scoring candidate set:

$$\text{MeanFitness}(\mathcal{D}_{\text{top}}) = \frac{1}{|\mathcal{D}_{\text{top}}|} \sum_{x \in \mathcal{D}_{\text{top}}} O(x) \tag{7}$$

3. **Novelty:** Quantifies the average Hamming distance between the top sequences and the wildtype sequence:

$$\text{Novelty}(\mathcal{D}_{\text{top}}, x_{WT}) = \frac{1}{|\mathcal{D}_{\text{top}}|} \sum_{x \in \mathcal{D}_{\text{top}}} d(x, x_{WT}) \tag{8}$$

4. **Diversity:** The average pairwise Hamming distance among the top-100 sequences, characterizing sequence variation within the selected set:

$$\text{Diversity}(\mathcal{D}_{\text{top}}) = \frac{1}{|\mathcal{D}_{\text{top}}|(|\mathcal{D}_{\text{top}}| - 1)} \sum_{x, x' \in \mathcal{D}_{\text{top}}, x \neq x'} d(x, x') \tag{9}$$

C Discussion

C.1 Hardware and runtime comparisons

Our code is developed in PyTorch [30] v.2.8.0. Experiments were conducted on an NVIDIA A40 GPU using CUDA 12.2. We measured the wall-clock runtime for one complete experiment consisting of ten active learning rounds across all eight landscapes for runtime comparison. Under this setup, SILO requires 4.5 hours. It is faster than ProSpero (20.86 hours) and GFN-AL-δCS (5.66 hours), but slower than MLDE (1.96 hours), PEX (0.70 hours), and AdaLead (0.67 hours). SILO uses additional sampling and policy-update computation in exchange

for stronger oracle-budgeted optimization, not to minimize local computation. Thus, SILO has intermediate computational cost rather than a uniform runtime advantage.

Table 3: Compute time required for one complete experiment for each method. A complete experiment consists of 10 active learning rounds performed on all eight benchmark landscapes.

Method	Runtime per experiment (hours)
PEX	0.70
AdaLead	0.67
GFN-AL-δCS	5.66
MLDE	1.96
ProSpero (enforcing reproducibility setting)	20.86
SILO	4.5

C.2 Limitations

Local search constraints. Because improvements are made incrementally within a budget of 1-3 edits per round, SILO may struggle to discover distant high-fitness regions that are not reachable through a sequence of local improvements. Evaluating larger mutational budgets remains an important direction for future work.
Unexamined sensitivity to oracle noise. Our noisy-proxy experiments perturb the proxy predictions used for candidate ranking while retaining noise-free oracle evaluations. These experiments do not establish robustness to noisy oracle labels, which could affect the selection of high-performing trajectories used for imitation-based policy updates and therefore should be examined further.
***In silico* oracles.** All experiments use *in silico* oracles rather than wet-lab measurements. Performance under real experimental conditions may differ due to batch effects, assay noise, and domain shift between the oracle model and actual biological fitness. Evaluating SILO in a wet-lab-in-the-loop setting would be an important next step toward validating practical applicability.

C.3 Broader impact

Our work improves methods for protein sequence optimization, with applications in areas such as enzyme design, biotechnology, and drug discovery. By enabling efficient search for high fitness variants under limited and noisy proxy settings, it may help accelerate the development of beneficial biological systems by reducing the burden of extensive wet lab experimentation. However, protein design methods can carry dual-use risks if misapplied. Notably, one of our benchmark tasks involves AAV caspid optimization, a protein family directly relevant to gene therapy vectors. More broadly, such optimization methods could be applied to optimize proteins with pathogenic, toxic, or otherwise harmful phenotypes. Therefore, appropriate safeguards for responsible use of these methods in downstream applications should be enforced. Furthermore, we will not release any task-specific fine-tuned policy weights for sequences flagged by biosecurity screening tools.

D Algorithmic details

D.1 Incremental stochastic beam search

We employ SBS, originally introduced by Kool et al. [22], as the core sampling procedure, which is a key component within SIL. SBS is a stochastic variant of beam search, which enables sampling trajectories **without replacement** by perturbing policy log-probabilities with Gumbel noise. Concretely, given a partial trajectory $a_{0:t}$, each candidate extension $a_{0:t+1}$ is scored using a perturbed log-probability and top-β candidates are retained in the beam.

$$\tilde{G}(a_{0:t+1}) = \log \pi_\theta(a_{0:t+1}) + \epsilon, \quad \epsilon \sim \mathrm{Gumbel}(0, 1) \tag{10}$$

In its incremental formulation, SBS allows sampling multiple complete trajectories sequentially while dynamically updating the underlying distribution. After a trajectory is sampled, its probability mass is removed and the distribution is renormalized, ensuring that subsequent samples are drawn from the remaining probability space. In our framework, SBS operates over the action space, where each node in the search tree represents a partial mutation trajectory. Combined with the SIL loop, this results in a set of high-quality and diverse action trajectories used for training.

E Complete results

E.1 Sequence optimization task

Table 4: Mean fitness of top-100 sequences attained by all methods on 8 benchmark datasets across 10 active learning rounds. Reported values are the mean ± standard deviation over 5 independent runs. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task. Underline: second-best

Method	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
PEX	0.603±0.010	0.234 ± 0.001	7.877± 0.047	3.597 ± 0.001	0.033 ± 0.001	1.476 ± 0.224	1.228 ± 0.000	2.987 ± 0.002
AdaLead	0.661 ± 0.013	0.228 ± 0.010	7.874 ± 0.064	3.571 ±0.002	0.034 ± 0.003	1.455 ± 0.154	1.209±0.000	2.974 ± 0.002
GFN-AL-δCS	0.638±0.007	0.184±0.010	7.917±0.054	3.576±0.001	0.020±0.002	1.485±0.324	1.207±0.003	2.981±0.002
MLDE	0.533±0.005	0.230±0.002	7.769±0.010	3.579±0.001	0.022±0.001	0.902±0.019	1.227±0.000	2.981±0.000
ProSpero	0.632 ± 0.023	0.232 ± 0.005	8.061 ± 0.010	3.611 ± 0.003	0.041 ± 0.001	1.512 ± 0.425	1.193 ± 0.011	2.985 ± 0.002
SILO	**0.727 ± 0.014**	**0.259 ± 0.005**	**8.148 ± 0.058**	**3.617 ± 0.001**	**0.044 ± 0.002**	**1.856 ± 0.355**	**1.232 ± 0.001**	**2.996 ± 0.002**

Table 5: Average novelty of top-100 sequences generated by all methods on 8 benchmark datasets across 10 active learning rounds. Reported values are the mean ± standard deviation over 5 independent runs. **Bold**: the best per each task. Underline: second-best

Method	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
PEX	6.694 ± 0.758	4.640 ± 0.799	3.584 ± 0.992	9.300 ± 1.308	9.302 ± 1.588	6.351 ± 1.376	2.206 ± 0.264	5.176 ± 0.818
AdaLead	10.510 ± 1.409	6.866 ± 1.883	4.886 ± 1.221	24.584 ± 5.407	32.806 ± 3.168	9.557 ± 1.650	2.642 ± 0.115	8.132 ± 3.066
GFN-AL-δCS	9.831 ± 0.325	5.414 ± 0.843	6.364 ± 0.616	23.630 ± 4.945	24.415 ± 3.535	**11.372 ± 2.705**	2.948 ± 0.048	7.685 ± 1.071
MLDE	2.896 ± 0.479	4.056 ± 0.710	1.280 ± 0.916	3.168 ± 0.420	4.218 ± 1.267	1.924 ± 0.638	1.080 ± 0.104	2.444 ± 0.699
ProSpero	**13.118 ± 1.647**	**16.666 ± 3.931**	**9.236 ± 1.931**	**42.328 ± 5.149**	**64.146 ± 4.232**	10.416 ± 2.879	3.224 ± 0.872	**16.256 ± 3.193**
SILO	12.320 ± 1.411	15.892 ± 1.610	8.516 ± 1.395	20.040 ± 0.825	32.432 ± 20.070	10.879 ± 2.013	**6.803 ± 1.872**	14.815 ± 2.996

Table 6: Average diversity between top-100 generated sequences generated by all methods on 8 benchmark datasets across 10 active learning rounds. Reported values are the mean ± standard deviation over 5 independent runs. **Bold**: the best per each task. Underline: second-best.

Method	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
PEX	7.069 ± 1.250	6.996 ± 0.905	5.164 ± 1.022	10.045 ± 1.117	8.042 ± 2.237	5.253 ± 0.659	4.016 ± 0.439	7.002 ± 0.629
AdaLead	6.766 ± 1.026	7.921 ± 3.262	6.651 ± 0.749	27.435 ± 9.213	19.442 ± 3.454	3.736 ± 1.473	3.853 ± 0.130	**11.461 ± 3.550**
GFN-AL-δCS	**10.743 ± 0.965**	9.287 ± 1.706	**7.368 ± 1.735**	**36.140 ± 6.136**	**43.118 ± 6.043**	**6.262 ± 1.382**	4.214 ± 0.096	11.015 ± 0.513
MLDE	2.395 ± 0.381	4.950 ± 0.580	1.757 ± 1.055	4.805 ± 0.690	6.885 ± 1.781	2.331 ± 0.688	1.893 ± 0.205	4.064 ± 0.999
ProSpero	5.565 ± 0.317	**12.404 ± 0.658**	3.815 ± 0.184	10.298 ± 1.520	12.519 ± 1.205	3.683 ± 0.369	**4.629 ± 0.828**	9.294 ± 1.592
SILO	2.624 ± 0.362	3.288 ± 0.484	2.619 ± 0.339	3.427 ± 1.140	6.497 ± 6.769	2.784 ± 0.219	2.523 ± 0.491	3.496 ± 0.537

Table 7: Median fitness of top-50 generated sequences generated all methods on 8 benchmark datasets across 10 active learning rounds. Reported values are the mean ± standard deviation over 5 independent runs. **Bold**: the best per each task. Underline: second-best.

Method	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
PEX	0.598±0.010	0.233±0.001	7.869±0.051	3.596±0.001	0.032±0.001	1.473±0.226	1.228±0.000	2.986±0.002
AdaLead	0.650±0.013	0.227±0.010	7.863±0.062	3.570±0.002	0.033±0.003	1.445±0.159	1.205±0.002	2.974±0.002
GFN-AL-δCS	0.634±0.007	0.181±0.013	7.911±0.052	3.575±0.002	0.019±0.002	1.480±0.325	1.203±0.005	2.981±0.002
MLDE	0.531±0.005	0.229±0.002	7.745±0.006	3.578±0.001	0.022±0.001	0.894±0.030	1.227±0.000	2.981±0.000
ProSpero	0.627±0.025	0.232±0.005	8.056±0.011	3.611±0.003	0.041±0.001	1.503±0.425	1.208±0.012	2.985±0.002
SILO	**0.726±0.014**	**0.259±0.005**	**8.146±0.058**	**3.617±0.001**	**0.044±0.002**	**1.854±0.356**	**1.232±0.001**	**2.996±0.002**

E.2 Statistical significance tests

We conducted two-sided paired *t*-test over the five aligned seeds against the strongest reproduced baseline, selected separately for each task and metric. We additionally report paired mean differences, 95% confidence intervals, Hedges' g_z, and Holm-adjusted *p*-values to account for multiple hypothesis testing (Tables 8) and 9).

Table 8: Paired comparison of maximum fitness between SILO and the strongest reproduced baseline for each benchmark using five aligned seeds. We report the paired mean difference (Δ), two-sided 95% confidence interval (CI), Hedges' g_z, the raw two-sided paired t-test p-value, and the Holm-adjusted p-value across the eight benchmarks. An asterisk indicates raw or Holm-adjusted $p < 0.05$.

Protein	**Baseline**	Δ	**95% CI**	**Hedges'** g_z	**Raw** p	**Holm** p
AAV	AdaLead	0.049	[0.023, 0.076]	1.897	0.006*	0.042*
AMIE	PEX	0.017	[0.014, 0.020]	5.221	0.0001*	8×10^{-4}*
E4B	ProSpero	0.027	[-0.046, 0.100]	0.368	0.361	1.000
GFP	ProSpero	0.004	[-0.002, 0.010]	0.643	0.146	0.730
LGK	ProSpero	0.002	[-0.001, 0.006]	0.559	0.192	0.768
Pab1	PEX	0.094	[-0.621, 0.811]	0.131	0.731	1.000
TEM	PEX	2.72×10^{-5}	[-0.003, 0.003]	0.008	0.982	1.000
UBE2I	GFN-AL-δCS	0.006	[0.002, 0.010]	1.468	0.014*	0.084

Table 9: Paired statistical comparison of top-100 mean fitness between SILO and the strongest reproduced baseline for each benchmark using five aligned random seeds. We report the paired mean difference Δ, two-sided 95% confidence interval (CI), Hedges' g_z, the raw two-sided paired t-test p-value, and the Holm-adjusted p-value across the eight benchmarks. An asterisk indicates raw or Holm-adjusted $p < 0.05$.

Protein	**Baseline**	Δ	**95% CI**	**Hedges'** g_z	**Raw** p	**Holm** p
AAV	AdaLead	0.066	[0.041, 0.090]	2.691	0.001*	0.007*
AMIE	PEX	0.024	[0.018, 0.031]	3.810	4.40×10^{-4}*	0.004*
E4B	ProSpero	0.087	[0.019, 0.154]	1.287	0.022*	0.084
GFP	ProSpero	0.006	[0.001, 0.011]	1.318	0.021*	0.084
LGK	ProSpero	0.003	[-4.83×10^{-4}, 0.007]	0.865	0.072	0.144
Pab1	ProSpero	0.344	[-0.255, 0.943]	0.570	0.186	0.186
TEM	PEX	0.004	[0.002, 0.005]	3.106	9.68×10^{-4}*	0.007*
UBE2I	PEX	0.009	[0.005, 0.013]	2.334	0.002*	0.010*

E.3 Early-round protein design

Table 10: Maximum fitness achieved by all methods on 8 benchmark datasets for the first 5 active learning rounds. Reported values are the mean $\pm$ standard deviation over 5 independent runs. **Bold**: the best per each task. Underline: second-best

Method	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
PEX	0.610±0.013	0.237 ± 0.002	7.999± 0.074	3.597 ± 0.001	0.032 ± 0.002	1.254 ± 0.151	1.231 ± 0.001	2.989 ± 0.002
AdaLead	0.634± 0.013	0.230 ± 0.005	8.000± 0.063	3.583 ±0.002	0.034 ± 0.004	1.240 ± 0.107	1.228±0.000	2.985 ± 0.003
GFN-AL-δCS	0.605±0.007	0.227±0.004	8.019±0.032	3.590±0.001	0.030±0.003	1.330±0.263	1.228±0.000	2.987±0.000
MLDE	0.543±0.008	0.235±0.002	7.857±0.036	3.585±0.003	0.024±0.001	0.935±0.000	1.228±0.000	2.985±0.000
ProSpero	0.617 ± 0.025	0.234 ± 0.005	8.021 ± 0.048	3.604 ± 0.004	**0.037 ± 0.002**	1.411± 0.356	1.229± 0.001	2.987 ± 0.002
SILO	**0.640 ± 0.020**	**0.251 ± 0.005**	**8.114 ± 0.058**	**3.608 ± 0.002**	0.038± 0.003	**1.698 ± 0.402**	**1.232 ± 0.001**	**2.993 ± 0.001**

Table 11: Mean fitness of top-100 sequences generated by all methods on 8 benchmark datasets for first 5 active learning rounds. Reported values are the mean $\pm$ standard deviation over 5 independent runs. **Bold**: the best per each task. Underline: second-best

Method	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
PEX	0.558±0.010	0.228± 0.002	7.738± 0.053	3.588 ± 0.001	0.027 ± 0.002	1.022 ± 0.008	1.220 ± 0.001	2.983± 0.001
AdaLead	0.574± 0.012	0.218 ± 0.006	7.772 ± 0.063	3.563 ±0.003	0.029 ± 0.005	1.115 ± 0.115	1.204±0.000	2.967± 0.003
GFN-AL-δCS	0.564±0.003	0.176±0.012	7.785±0.013	3.568±0.002	0.017±0.003	1.077±0.106	1.204±0.002	2.977±0.002
MLDE	0.518±0.008	0.227±0.001	7.744±0.004	3.576±0.001	0.021±0.000	0.860±0.004	1.226±0.001	2.980±0.000
ProSpero	0.566 ± 0.019	0.225± 0.005	7.841 ± 0.045	3.599 ± 0.004	0.034 ± 0.002	1.239 ± 0.246	1.087 ± 0.038	2.978 ± 0.004
SILO	**0.606 ± 0.011**	**0.244 ± 0.005**	**8.044 ± 0.087**	**3.605 ± 0.003**	**0.037 ± 0.003**	**1.504 ± 0.384**	**1.231 ± 0.001**	**2.991 ± 0.001**

E.4 Low-labeled data regime

Table 12: Maximum fitness and mean fitness of top-100 generated sequences by all methods under the low-data setting using 10% of the initial D_0 dataset. Results are reported as mean $\pm$ standard deviation over 5 independent runs. **Bold**: best result for each task and metric. Underline: second-best result.

	Maximum fitness				Mean fitness			
Method	UBE2I	TEM	LGK	Pab1	UBE2I	TEM	LGK	Pab1
PEX	2.990 ± 0.004	1.228 ± 0.005	0.035 ± 0.002	1.536 ± 0.340	2.984 ± 0.003	1.203 ± 0.012	0.032 ± 0.001	1.330 ± 0.304
AdaLead	2.986 ± 0.004	1.220 ± 0.009	0.034 ± 0.001	1.239 ± 0.301	2.976 ± 0.006	1.196 ± 0.003	0.031 ± 0.002	1.136 ± 0.246
GFN-AL-δCS	2.986 ± 0.004	1.219 ± 0.009	0.017 ± 0.0019	0.611 ± 0.172	2.975 ± 0.009	1.200 ± 0.013	-0.024 ± 0.000	0.161 ± 0.007
MLDE	2.986 ± 0.001	1.228 ± 0.000	0.024 ± 0.000	0.937 ± 0.003	2.982 ± 0.000	1.222 ± 0.005	0.022 ± 0.000	0.928 ± 0.004
ProSpero	2.992 ± 0.002	**1.229 ± 0.001**	**0.042 ± 0.001**	**1.739 ± 0.312**	2.986 ± 0.001	1.161 ± 0.046	0.040 ± 0.001	**1.611 ± 0.272**
SILO	**2.993 ± 0.005**	**1.229 ± 0.001**	**0.042 ± 0.001**	1.591 ± 0.474	**2.992 ± 0.005**	**1.228 ± 0.000**	**0.041 ± 0.001**	1.450 ± 0.455

Table 13: Average novelty and diversity of the top-100 generated sequences under the low-data setting using 10% of the initial D_0 dataset. Results are reported as mean $\pm$ standard deviation over 5 independent runs. **Bold**: best result per task and metric. Underline: second-best result.

	Novelty				Diversity			
Method	UBE2I	TEM	LGK	Pab1	UBE2I	TEM	LGK	Pab1
PEX	5.440±1.798	1.160±0.272	10.406±2.155	6.715±1.940	7.807±1.644	3.313±0.296	7.894±1.603	5.970±1.227
AdaLead	4.660±2.292	0.910±0.082	52.266±10.176	11.418±1.657	7.498±2.207	3.638±0.136	26.002±3.422	4.826±1.896
GFN-AL-δCS	5.376±1.782	1.564±0.749	**425.002±0.406**	**71.226±0.251**	5.812±1.382	4.045±0.672	**424.433±0.080**	**71.123±0.024**
MLDE	5.192±1.800	**3.979±0.493**	4.892±0.826	2.056±0.286	5.165±0.718	1.872±0.179	7.520±1.427	1.825±0.650
ProSpero	**17.584±3.784**	2.608±0.407	60.100±6.931	11.440±0.829	**10.371±0.842**	**4.133±0.411**	16.152±3.274	3.626±0.150
SILO	10.580±3.255	1.456±0.564	21.844±3.006	10.680±1.346	2.587±0.580	1.704±0.120	4.081±0.701	2.984±0.135

Table 14: Maximum fitness and mean fitness under the low-data setting using 20% of the initial D_0 dataset. Results are reported over 5 independent runs. **Bold**: best result for each task and metric. Underline: second-best result.

	Maximum fitness				Mean fitness			
Method	UBE2I	TEM	LGK	Pab1	UBE2I	TEM	LGK	Pab1
PEX	**2.992 ± 0.001**	1.229 ± 0.003	0.034 ± 0.001	1.690 ± 0.294	2.987 ± 0.000	1.215 ± 0.008	0.031 ± 0.001	1.449 ± 0.366
AdaLead	2.989 ± 0.002	1.227 ± 0.001	0.036 ± 0.002	**1.719 ± 0.465**	2.979 ± 0.002	1.201 ± 0.000	0.033 ± 0.003	**1.625 ± 0.428**
GFN-AL-δCS	2.990 ± 0.003	1.224 ± 0.007	0.018 ± 0.001	0.735 ± 0.098	2.981 ± 0.008	1.213 ± 0.008	-0.016 ± 0.000	0.185 ± 0.025
MLDE	2.985 ± 0.000	1.228 ± 0.000	0.024 ± 0.001	0.939 ± 0.002	2.981 ± 0.000	1.224 ± 0.003	0.022 ± 0.000	0.930 ± 0.001
ProSpero	**2.992 ± 0.004**	**1.230 ± 0.001**	0.042 ± 0.002	1.420 ± 0.316	2.985 ± 0.003	1.152 ± 0.036	0.041 ± 0.003	1.336 ± 0.261
SILO	**2.992 ± 0.005**	1.229 ± 0.001	**0.043 ± 0.001**	1.674 ± 0.424	**2.992 ± 0.004**	**1.228 ± 0.001**	**0.042 ± 0.001**	1.600 ± 0.424

Table 15: Average novelty and diversity of the top-100 generated sequences under the low-data setting using 20% of the initial D_0 dataset across five independent runs. Values represent mean $\pm$ standard deviation. **Bold**: best result per task and metric. Underline: second-best result.

	Novelty				Diversity			
Method	UBE2I	TEM	LGK	Pab1	UBE2I	TEM	LGK	Pab1
PEX	4.873±0.631	1.507±0.698	8.546±0.913	7.514±0.459	7.905±1.175	3.464±0.411	7.838±0.965	5.865±0.675
AdaLead	3.596±0.493	0.621±0.131	45.868±10.567	10.158±0.880	7.092±1.089	3.102±0.235	24.923±7.244	4.406±1.106
GFN-AL-δCS	6.846±1.777	2.507±0.667	**424.669±0.596**	**71.160±0.112**	8.908±2.035	4.755±0.583	**424.510±0.087**	**71.129±0.036**
MLDE	5.320±1.071	**4.562±0.439**	4.792±0.742	1.982±0.137	5.529±0.691	2.227±0.233	6.272±1.397	1.702±0.200
ProSpero	**16.210±4.653**	3.358±0.922	59.084±9.199	11.604±1.622	**11.095±1.276**	**5.006±1.087**	13.506±1.878	3.578±0.186
SILO	9.864±2.990	3.239±2.407	20.696±3.424	12.132±2.062	2.796±0.479	2.006±0.487	4.434±0.283	3.227±0.307

Table 16: Maximum fitness and mean fitness under the low-data setting using 50% of the initial D_0 dataset. Results are reported over 5 independent runs. **Bold**: best result for each task and metric. Underline: second-best result.

	Maximum fitness				Mean fitness			
Method	UBE2I	TEM	LGK	Pab1	UBE2I	TEM	LGK	Pab1
PEX	2.993 ± 0.002	1.229 ± 0.001	0.035 ± 0.001	1.740 ± 0.437	2.987 ± 0.000	1.220 ± 0.004	0.032 ± 0.002	1.510 ± 0.305
AdaLead	2.987 ± 0.003	1.228 ± 0.001	0.038 ± 0.002	**1.909 ± 0.190**	2.975 ± 0.003	1.204 ± 0.001	0.035 ± 0.002	**1.726 ± 0.223**
GFN-AL-δCS	2.990 ± 0.002	1.228 ± 0.000	0.019 ± 0.000	0.808 ± 0.077	2.981 ± 0.005	1.206 ± 0.003	-0.003 ± 0.001	0.239 ± 0.026
MLDE	2.986 ± 0.000	1.229 ± 0.000	0.025 ± 0.000	0.946 ± 0.011	2.982 ± 0.000	1.223 ± 0.004	0.023 ± 0.001	0.923 ± 0.005
ProSpero	2.992 ± 0.003	1.229 ± 0.000	0.042 ± 0.002	1.550 ± 0.317	2.986 ± 0.002	1.101 ± 0.042	0.041 ± 0.002	1.453 ± 0.316
SILO	**2.995 ± 0.006**	**1.230 ± 0.002**	**0.044 ± 0.001**	1.683 ± 0.279	**2.994 ± 0.005**	**1.229 ± 0.001**	**0.043 ± 0.001**	1.572 ± 0.293

Table 17: Average novelty and diversity of the top-100 generated sequences under the low-data setting using 50% of the initial D_0 dataset across five independent runs. Values represent mean ± standard deviation. **Bold**: best result per task and metric. Underline: second-best result.

	Novelty				Diversity			
Method	UBE2I	TEM	LGK	Pab1	UBE2I	TEM	LGK	Pab1
PEX	6.157±0.879	1.525±0.233	9.768±0.977	7.312±3.337	8.350±0.374	3.515±0.262	7.903±1.375	5.133±0.629
AdaLead	4.309±0.717	0.578±0.075	38.750±11.402	10.318±4.507	8.692±1.009	2.993±0.123	20.172±7.390	4.785±2.180
GFN-AL-δCS	6.006±1.454	1.223±0.800	**424.602±0.194**	**71.174±0.274**	8.164±2.749	3.898±1.040	**424.528±0.056**	**71.128±0.0210**
MLDE	5.056±0.013	**5.050±0.933**	4.904±0.645	2.528±0.198	5.461±0.839	2.275±0.296	6.658±1.621	2.705±0.579
ProSpero	**18.938±4.909**	2.864±0.818	64.126±5.210	11.188±1.895	**11.420±0.936**	**4.380±0.550**	12.877±2.704	3.911±0.239
SILO	11.908±3.113	2.752±2.999	22.291±2.457	10.358±1.146	2.871±0.616	1.848±0.431	3.799±0.527	3.086±0.280

E.5 Noisy proxy setting

Table 18: Results under noisy proxy setting. Maximum fitness and mean fitness of top-100 sequences by all methods on four benchmark datasets across 10 active learning rounds under the -5 signal-to-noise setting. Reported values represent mean ± standard deviation across 5 independent runs. **Bold**: best result for each task and metric. Underline: second-best result.

	Maximum fitness				Mean fitness			
Method	AMIE	E4B	GFP	AAV	AMIE	E4B	GFP	AAV
PEX	0.242 ± 0.001	8.002 ± 0.074	3.603 ± 0.002	0.626 ± 0.014	0.234 ± 0.001	7.826 ± 0.037	3.595 ± 0.002	0.573 ± 0.015
AdaLead	0.235 ± 0.001	8.002 ± 0.040	3.586 ± 0.001	0.688 ± 0.027	0.227 ± 0.001	7.805 ± 0.034	3.570 ± 0.001	0.641 ± 0.027
GFN-AL-δCS	0.229 ± 0.002	7.952 ± 0.064	3.590 ± 0.002	0.567 ± 0.003	0.212 ± 0.003	7.700 ± 0.031	3.580 ± 0.001	0.516 ± 0.005
MLDE	0.233 ± 0.002	7.836 ± 0.063	3.582 ± 0.001	0.525 ± 0.000	0.227 ± 0.001	7.738 ± 0.021	3.576 ± 0.000	0.506 ± 0.003
ProSpero	0.242 ± 0.003	**8.139 ± 0.019**	3.616 ± 0.003	0.695 ± 0.015	0.233 ± 0.003	8.074 ± 0.023	3.611 ± 0.003	0.655 ± 0.018
SILO	**0.260 ± 0.002**	8.118 ± 0.040	**3.620 ± 0.002**	**0.704 ± 0.035**	**0.257 ± 0.003**	**8.088 ± 0.044**	**3.618 ± 0.002**	**0.673 ± 0.026**

Table 19: Average novelty and diversity of the top-100 sequences by all methods on four benchmark datasets across 10 active learning rounds under the -5 signal-to-noise setting. Reported values represent mean ± standard deviation across 5 independent runs. **Bold**: best result per task and metric. Underline: second-best result.

	Novelty				Diversity			
Method	AMIE	E4B	GFP	AAV	AMIE	E4B	GFP	AAV
PEX	3.934±0.242	2.932±0.561	7.914±1.998	5.414±0.974	6.442±0.738	5.161±0.823	11.277±2.310	7.024±0.726
AdaLead	4.135±0.421	3.175±0.509	22.606±2.789	10.558±1.527	8.264±0.700	5.742±0.579	**34.636±2.832**	6.137±1.434
GFN-AL-δCS	1.985±0.210	3.796±0.509	17.610±5.068	4.696±0.384	4.835±0.341	**6.512±0.539**	24.559±5.178	**8.072±0.401**
MLDE	2.985±0.074	1.690±0.139	2.268±0.197	1.682±0.116	3.343±0.429	2.586±0.260	4.003±0.264	2.673±0.160
ProSpero	**17.916±1.868**	**9.970±1.982**	**48.762±3.168**	**14.472±0.740**	**12.485±0.732**	4.329±0.510	15.842±0.420	6.229±0.748
SILO	14.172±2.557	8.532±1.336	18.253±1.623	9.936±1.744	2.917±0.355	2.709±0.217	3.836±0.813	3.164±0.206

Table 20: Results under noisy proxy setting. Maximum fitness and mean fitness of top-100 sequences by all methods on four benchmark datasets across 10 active learning rounds under the -15 signal-to-noise setting. Reported values represent mean $\pm$ standard deviation across 5 independent runs. **Bold**: best result for each task and metric. Underline: second-best result.

	Maximum fitness				Mean fitness			
Method	AMIE	E4B	GFP	AAV	AMIE	E4B	GFP	AAV
PEX	0.242 ± 0.001	7.999 ± 0.033	3.602 ± 0.002	0.586 ± 0.007	0.233 ± 0.001	7.783 ± 0.026	3.595 ± 0.002	0.531 ± 0.006
AdaLead	0.237 ± 0.007	7.941 ± 0.081	3.583 ± 0.002	0.574 ± 0.041	0.228 ± 0.003	7.717 ± 0.045	3.562 ± 0.002	0.529 ± 0.037
GFN-AL-δCS	0.226 ± 0.001	7.866 ± 0.036	3.588 ± 0.004	0.532 ± 0.006	0.205 ± 0.002	7.277 ± 0.128	3.576 ± 0.001	0.482 ± 0.003
MLDE	0.233 ± 0.002	7.840 ± 0.050	3.582 ± 0.001	0.531 ± 0.010	0.227 ± 0.001	7.728 ± 0.020	3.575 ± 0.000	0.495 ± 0.006
ProSpero	0.238 ± 0.004	8.100 ± 0.022	3.616 ± 0.003	0.649 ± 0.031	0.226 ± 0.004	7.953 ± 0.076	3.609 ± 0.002	0.598 ± 0.030
SILO	$\mathbf{0.259 \pm 0.004}$	$\mathbf{8.119 \pm 0.089}$	$\mathbf{3.617 \pm 0.002}$	$\mathbf{0.689 \pm 0.024}$	$\mathbf{0.256 \pm 0.004}$	$\mathbf{8.090 \pm 0.093}$	$\mathbf{3.615 \pm 0.002}$	$\mathbf{0.644 \pm 0.011}$

Table 21: Average novelty and diversity of the top-100 sequences by all methods on four benchmark datasets across 10 active learning rounds under the -15 signal-to-noise setting. Reported values represent mean $\pm$ standard deviation across 5 independent runs. **Bold**: best result per task and metric. Underline: second-best result.

	Novelty				Diversity			
Method	AMIE	E4B	GFP	AAV	AMIE	E4B	GFP	AAV
PEX	3.621±0.416	2.582±0.214	7.620±1.146	3.412±0.292	5.961±0.735	4.699±0.315	11.368±1.185	5.483±0.443
AdaLead	4.662±1.262	3.243±0.518	26.125±3.414	4.948±1.917	7.881±0.464	5.930±0.778	**42.165±5.277**	5.849±0.922
GFN-AL-δCS	1.843±0.268	3.543±0.262	16.782±5.940	3.520±0.150	4.803±0.403	**5.960±0.153**	24.311±7.52	**6.645±0.283**
MLDE	2.879±0.133	1.644±0.260	1.908±0.197	1.319±0.060	3.089±0.259	2.538±0.504	3.472±0.344	2.086±0.471
ProSpero	**16.274±3.813**	7.028±1.661	**49.168±4.558**	**13.546±1.954**	**14.019±1.878**	5.081±0.223	17.787±2.054	6.276±0.425
SILO	13.880±1.225	**8.492±2.085**	17.966±2.793	8.950±1.576	3.145±0.954	2.441±0.286	3.700±0.379	3.150±0.170

Table 22: Results under noisy proxy setting. Maximum fitness and mean fitness of top-100 sequences by all methods on four benchmark datasets across 10 active learning rounds under the -25 signal-to-noise setting. Reported values represent mean $\pm$ standard deviation across 5 independent runs. **Bold**: best result for each task and metric. Underline: second-best result.

	Maximum fitness				Mean fitness			
Method	AMIE	E4B	GFP	AAV	AMIE	E4B	GFP	AAV
PEX	0.241 ± 0.001	7.956 ± 0.036	3.604 ± 0.002	0.564 ± 0.005	0.232 ± 0.002	7.744 ± 0.018	3.597 ± 0.002	0.514 ± 0.005
AdaLead	0.237 ± 0.004	7.929 ± 0.074	3.583 ± 0.002	0.512 ± 0.004	0.225 ± 0.001	7.602 ± 0.092	3.557 ± 0.003	0.438 ± 0.011
GFN-AL-δCS	0.227 ± 0.003	7.773 ± 0.053	3.583 ± 0.004	0.525 ± 0.011	0.201 ± 0.001	5.674 ± 0.127	3.566 ± 0.003	0.455 ± 0.005
MLDE	0.234 ± 0.003	7.885 ± 0.081	3.582 ± 0.000	0.529 ± 0.009	0.226 ± 0.002	7.725 ± 0.023	3.575 ± 0.000	0.490 ± 0.008
ProSpero	0.233 ± 0.003	7.997 ± 0.106	3.613 ± 0.003	0.559 ± 0.032	0.217 ± 0.003	7.785 ± 0.141	3.607 ± 0.002	0.493 ± 0.022
SILO	$\mathbf{0.257 \pm 0.001}$	$\mathbf{8.086 \pm 0.030}$	$\mathbf{3.618 \pm 0.001}$	$\mathbf{0.670 \pm 0.035}$	$\mathbf{0.254 \pm 0.000}$	$\mathbf{8.056 \pm 0.034}$	$\mathbf{3.616 \pm 0.001}$	$\mathbf{0.635 \pm 0.033}$

Table 23: Average novelty and diversity of the top-100 sequences by all methods on four benchmark datasets across 10 active learning rounds under the -25 signal-to-noise setting. Reported values represent mean $\pm$ standard deviation across 5 independent runs. **Bold**: best result per task and metric. Underline: second-best result.

	Novelty				Diversity			
Method	AMIE	E4B	GFP	AAV	AMIE	E4B	GFP	AAV
PEX	3.262±0.646	2.126±0.246	8.456±1.077	2.338±0.248	5.605±0.701	3.912±0.362	11.754±1.755	4.268±0.509
AdaLead	4.482±0.300	3.623±0.608	29.833±4.268	3.392±0.061	8.638±0.448	6.261±0.807	**48.738±6.171**	6.418±0.092
GFN-AL-δCS	1.553±0.145	1.684±0.188	21.454±5.738	3.572±0.250	4.339±0.175	**7.308±0.339**	33.207±6.993	6.817±0.468
MLDE	3.076±0.153	1.666±0.200	1.932±0.168	1.180±0.136	3.153±0.299	2.515±0.625	3.458±0.305	1.955±0.430
ProSpero	12.078±3.181	6.324±1.752	**53.108±1.373**	7.372±1.412	**14.366±0.581**	5.752±0.193	19.187±2.451	**7.507±0.452**
SILO	**14.315±1.927**	**7.828±1.053**	19.990±2.111	**9.238±1.602**	3.465±0.373	2.632±0.236	3.870±0.936	3.196±0.233

E.6 Ablation studies

Table 24: Ablation results across all 8 protein fitness landscapes, reporting the maximum fitness achieved over 10 active-learning rounds. Reported values represent the mean ± standard deviation across 5 independent runs. The ablations examine the contributions of sequence generation strategy, search and candidate selection strategies, and iterative policy learning. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task.

Landscape	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
Sequence generation strategy								
Random baseline	0.530±0.015	0.229 ± 0.002	6.970 ±1.156	3.600±0.010	0.027 ±0.001	1.018± 0.103	1.229 ± 0.000	2.984 ± 0.000
Sampling and selection								
SIL + BS + w/o AFS score	0.549±0.026	0.230±0.006	7.642±0.289	3.593±0.006	0.026±0.002	0.942±0.090	1.228±0.000	2.986± 0.001
SIL + BS + with AFS score	0.587±0.028	0.232 ± 0.007	7.706 ±0.173	3.602±0.012	0.026 ±0.001	1.022± 0.146	1.228 ± 0.000	2.987 ± 0.001
SIL + SBS + w/o AFS score	0.694±0.019	0.258 ±0.003	8.140 ±0.062	3.615±0.001	0.042±0.001	1.558 ± 0.134	1.232 ± 0.000	**2.999 ± 0.001**
Impact of Learning								
Frozen policy + BS + AFS score	0.551±0.024	0.227±0.005	7.604±0.269	3.595±0.010	0.027±0.003	0.901±0.045	1.228±0.000	2.987±0.001
Frozen policy + SBS + AFS score	0.706±0.013	0.256 ± 0.003	8.149 ± 0.041	3.615±0.001	0.042 ±0.001	1.530±0.268	1.233 ±0.002	2.997 ± 0.001
SILO (SBS + AFS score + SIL)	**0.747 ± 0.014**	**0.262 ± 0.004**	**8.169 ± 0.052**	**3.619 ± 0.002**	**0.045 ± 0.003**	**1.894 ± 0.348**	**1.233 ± 0.003**	2.997 ± 0.003

Table 25: Ablation results across all 8 protein fitness landscapes, reporting the mean fitness of top-100 sequences achieved over 10 active-learning rounds. Reported values represent the mean ± standard deviation across 5 independent runs. The ablations examine the contributions of sequence generation strategy, search and candidate selection strategies, and iterative policy learning. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task.

Landscape	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
Sequence generation strategy								
Random baseline	0.526±0.014	0.225 ± 0.001	6.722 ±1.138	3.599±0.010	0.024 ±0.001	0.997± 0.010	1.228 ± 0.000	2.982 ± 0.000
Sampling and selection								
SIL + BS + w/o AFS score	0.548±0.025	0.230±0.006	7.601±0.319	3.592±0.006	0.025±0.000	0.929±0.093	1.217±0.024	2.985± 0.001
SIL + BS + with AFS score	0.582±0.028	0.231 ± 0.007	7.641 ±0.248	3.600±0.011	0.025 ±0.001	1.002± 0.143	1.222 ± 0.011	2.986 ± 0.001
SIL + SBS + w/o AFS score	0.673±0.014	0.253 ±0.004	8.107 ±0.067	3.614±0.000	0.041±0.001	1.467 ± 0.111	1.231 ± 0.000	**2.998 ± 0.001**
Impact of Learning								
Frozen policy + BS + AFS score	0.549±0.022	0.227±0.004	7.589±0.271	3.594±0.010	0.027±0.003	0.898±0.042	1.215±0.020	2.986±0.001
Frozen policy + SBS + AFS score	0.669±0.024	0.251 ± 0.003	8.106 ± 0.060	3.613±0.001	0.041 ±0.001	1.383±0.207	1.231 ±0.001	2.995 ± 0.001
SILO (SBS + AFS score + SIL)	**0.727 ± 0.014**	**0.259 ± 0.005**	**8.148 ± 0.058**	**3.617 ± 0.001**	**0.044 ± 0.002**	**1.856 ± 0.355**	**1.232 ± 0.001**	2.996 ± 0.002

Table 26: Ablation results across all 8 protein fitness landscapes, reporting the novelty achieved over 10 active-learning rounds. Reported values represent the mean ± standard deviation across 5 independent runs. The ablations examine the contributions of sequence generation strategy, search and candidate selection strategies, and iterative policy learning. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task.

Landscape	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
Sequence generation strategy								
Random baseline	3.268±1.436	1.866 ± 0.043	1.230 ±0.252	13.052±6.317	6.452 ± 1.919	4.928± 2.663	1.478 ± 0.521	2.700 ± 0.071
Sampling and selection								
SIL + BS + w/o AFS score	3.524±1.648	4.802±4.159	1.473±1.049	9.064±2.930	6.994±3.277	2.916±2.847	1.354±0.229	4.950± 1.067
SIL + BS + with AFS score	5.556±1.327	5.216 ± 4.248	1.723 ±0.945	10.099±3.812	6.614 ±3.116	4.360± 3.408	1.372 ± 0.248	6.078 ± 1.461
SIL + SBS + w/o AFS score	9.584±0.685	14.406 ±2.543	8.354 ±1.870	19.360±2.542	22.386±2.356	10.512± 1.742	**6.830 ± 1.916**	13.434 ± 1.524
Impact of Learning								
Frozen policy + BS + AFS score	3.736±1.398	3.636±2.210	1.030±0.375	12.356±4.095	8.408±2.931	1.564±0.507	1.230±0.213	5.808±1.465
Frozen policy + SBS + AFS score	10.388 ±0.973	14.218 ± 2.179	7.916 ± 1.114	18.612 ± 3.984	19.360 ±2.179	10.384± 1.911	6.102 ±1.515	13.366 ± 2.197
SILO (SBS + AFS score + SIL)	**12.320 ± 1.411**	**15.892 ± 1.610**	**8.516 ± 1.395**	**20.040 ± 0.825**	**32.432 ± 20.070**	**10.798 ± 2.013**	6.803 ± 1.872	**14.815 ± 2.992**

Table 27: Ablation results across all 8 protein fitness landscapes, reporting the diversity achieved over 10 active-learning rounds. Values represent the mean ± standard deviation across 5 independent runs. The ablations examine the contributions of sequence generation strategy, search and candidate selection strategies, and iterative policy learning. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task.

Landscape	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
Sequence generation strategy								
Random baseline	1.518±1.147	3.292 ± 0.669	2.227 ±0.500	2.139± 0.382	**7.740 ± 2.957**	1.531± 0.888	2.055± 0.074	**4.237 ± 0.927**
Sampling and selection								
SIL + BS + w/o AFS score	0.665± 0.419	1.432± 0.779	1.044±0.434	1.745± 0.708	2.250±0.895	0.600±0.543	0.798±0.172	1.593± 0.420
SIL + BS + with AFS score	1.701±1.156	1.614 ± 0.938	1.083 ±0.334	1.912± 0.821	2.139 ±0.786	1.076± 0.844	0.825 ± 0.220	1.983 ± 0.323
SIL + SBS + w/o AFS score	2.477±0.167	**3.642± 0.885**	**2.927 ± 0.194**	2.861± 0.191	3.716± 0.588	2.665± 0.029	3.081 ± 0.266	3.197 ± 0.560
Impact of Learning								
Frozen policy + BS + AFS score	0.656± 0.425	1.115± 0.651	0.728±0.254	2.159± 0.435	2.335±0.581	0.081±0.094	0.564 ±0.089	1.594±0.507
Frozen policy + SBS + AFS score	2.616 ± 0.225	3.474 ± 0.291	2.875 ± 0.351	2.778 ± 0.220	3.896 ± 0.494	2.669± 0.386	**3.123 ± 0.409**	3.013 ± 0.292
SILO (SBS + AFS score + SIL)	**2.624 ± 0.362**	3.288 ± 0.484	2.619 ± 0.339	**3.427 ± 1.140**	6.497 ± 6.769	**2.784 ± 0.219**	2.523 ± 0.491	3.496 ± 0.537

Table 28: Paired statistical comparison of max-fitness values attained by full SILO and frozen-policy SBS for each benchmark using the five shared seeds. We report the paired mean difference (Δ), 95% confidence interval (CI), Hedges' g_z, the raw paired t-test p-value, and the Holm-adjusted p-value. An asterisk indicates raw or Holm-adjusted $p < 0.05$.

Protein	Δ	**95% CI**	**Hedges' g_z**	**Raw p**	**Holm p**
AAV	0.040	[0.023, 0.057]	2.365	0.002*	0.016*
AMIE	0.006	[0.001, 0.010]	1.321	0.020*	0.114
E4B	0.020	[-0.066, 0.108]	0.233	0.548	1.000
GFP	0.004	[0.001, 0.006]	1.721	0.008*	0.056
LGK	0.002	[-0.001, 0.006]	0.549	0.199	0.796
Pab1	0.363	[0.096, 0.630]	1.352	0.019*	0.114
TEM	-0.000	[-0.002, 0.002]	-0.045	0.905	1.000
UBE2I	0.000	[-0.002, 0.004]	0.193	0.618	1.000

Table 29: Paired statistical comparison of max fitness values attained by SILO against the SIL-beam search-AFS variant for each benchmark using the five shared random seeds. We report the paired mean difference (Δ), 95% confidence interval (CI), Hedges' g_z, the raw paired t-test p-value, and the Holm-adjusted p-value. An asterisk indicates raw or Holm-adjusted $p < 0.05$.

Protein	Δ	**95% CI**	**Hedges' g_z**	**Raw p**	**Holm p**
AAV	0.159	[0.133, 0.185]	6.066	$7.11 \times 10^{-5*}$	$5.6 \times 10^{-4*}$
AMIE	0.030	[0.016, 0.045]	2.146	0.003*	0.012*
E4B	0.463	[0.271, 0.654]	2.403	0.002*	0.010*
GFP	0.017	[0.001, 0.032]	1.115	0.035*	0.035*
LGK	0.019	[0.012, 0.025]	3.111	$9.6 \times 10^{-4*}$	0.007*
Pab1	0.871	[0.465, 1.270]	2.134	0.003*	0.012*
TEM	0.004	[0.002, 0.006]	2.204	0.003*	0.012*
UBE2I	0.010	[0.007, 0.013]	3.105	$9.6 \times 10^{-4*}$	0.007*

E.7 Hyperparameter sensitivity analysis

Influence of varying number of top trajectories P for policy training For the number of top oracle evaluated trajectories used for training policy, $P = 50$ achieves the highest mean maximum fitness on six landscapes, ties at the displayed precision on TEM, and is within 0.001 of the best setting on UBE2I (Table 30). Relative to $P = 25$, it improves the mean on six landscapes and ties on two. Increasing P further to 75 does not provide a consistent additional benefit, suggesting that $P = 50$ provides a reasonable balance between the number and quality of imitation trajectories.

Influence of varying β beam size within stochastic beam search Increasing the beam width from $\beta = 16$ to $\beta = 32$ improves the reported mean on seven landscapes and ties on UBE2I (Table 31). Increasing it

further β =64 to produces only small, task-dependent changes without a consistent improvement across the suite. We therefore retain as a practical balance between optimization performance and sampling cost.

Influence of varying γ_1 and γ_2 UCB coefficients Performance is generally stable across the evaluated uncertainty coefficients. The default $(\gamma_1, \gamma_2) = (0.1, 1.0)$ is best or tied at the displayed precision on seven of eight landscapes and is within 0.002 of the best result on UBE2I (Table 32). These results show that the selected setting is locally stable within the tested range.

Table 30: Effect of number of top oracle evaluated trajectories (P) for policy training on SILO's performance on 8 benchmark datasets across 10 active learning rounds (maximum fitness). Reported values are the mean ± standard deviation over 5 independent runs. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task. Underline: second-best

Parameter	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
P =25	0.725 ± 0.027	0.260 ± 0.006	8.119 ± 0.047	3.618 ± 0.002	0.042 ± 0.001	1.481 ± 0.436	1.233 ± 0.001	2.997 ± 0.002
P =75	0.709 ± 0.031	0.258 ± 0.004	8.107 ± 0.043	3.616 ± 0.001	0.043 ± 0.001	1.665 ± 0.337	1.233 ± 0.002	**2.998 ± 0.002**
P =50 (SILO)	**0.747 ± 0.014**	**0.262 ± 0.004**	**8.169 ± 0.052**	**3.619 ± 0.002**	**0.045 ± 0.003**	**1.894 ± 0.348**	1.233 ± 0.003	2.997 ± 0.003

Table 31: Effect of varying length of beam search (β) within stochastic beam search on SILO's performance on 8 benchmark datasets across 10 active learning rounds (maximum fitness). Reported values are the mean ± standard deviation over 5 independent runs. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task. Underline: second-best

Parameter	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
β=16	0.676 ± 0.027	0.258±0.004	8.143 ± 0.028	3.617 ± 0.002	0.042 ± 0.002	1.508 ± 0.283	1.231 ± 0.001	2.997 ± 0.003
β=64	0.728 ± 0.021	**0.262 ± 0.005**	**8.171 ± 0.027**	3.618 ± 0.002	0.044 ± 0.002	**1.898 ± 0.180**	**1.234 ± 0.001**	**2.999 ± 0.002**
β=32 (SILO)	**0.747 ± 0.014**	**0.262 ± 0.004**	8.169 ± 0.052	**3.619 ± 0.002**	**0.045 ± 0.003**	1.894 ± 0.348	1.233 ± 0.003	2.997 ± 0.003

Table 32: Sensitivity analysis over all four combinations of the UCB uncertainty coefficients γ_1 and γ_2 on SILO's performance on 8 benchmark datasets across 10 active learning rounds (maximum fitness). Reported values are the mean ± standard deviation over 5 independent runs. Values in brackets denote wildtype (x_{WT}) fitness. **Bold**: the best per each task. Underline: second-best

Parameters	AAV (0.500)	AMIE (0.224)	E4B (7.743)	GFP (3.572)	LGK (0.020)	Pab1 (0.84)	TEM (1.229)	UBE2I (2.978)
$\gamma_1 = 0.1, \gamma_2 = 0.1$	0.718 ± 0.028	0.260±0.005	8.139 ± 0.028	3.615 ± 0.001	0.044 ± 0.001	1.738 ± 0.298	**1.233 ± 0.001**	2.998 ± 0.002
$\gamma_1 = 1.0, \gamma_2 = 0.1$	0.725 ± 0.034	0.259 ± 0.002	8.168 ± 0.032	3.618 ± 0.002	0.042 ± 0.000	1.678 ± 0.293	**1.233 ± 0.002**	**2.999 ± 0.002**
$\gamma_1 = 1.0, \gamma_2 = 1.0$	0.732 ± 0.021	**0.262 ± 0.005**	8.123 ± 0.040	**3.619 ± 0.001**	0.042 ± 0.002	1.623 ± 0.358	1.232 ± 0.001	2.998 ± 0.002
$\gamma_1 = 0.1, \gamma_2 = 1.0$ (SILO)	**0.747 ± 0.014**	**0.262 ± 0.004**	**8.169 ± 0.052**	**3.619 ± 0.002**	**0.045 ± 0.003**	**1.894 ± 0.348**	**1.233 ± 0.003**	2.997 ± 0.003

E.8 Results reported from ProSpero

Table 33: Maximum fitness values obtained by all methods on 8 benchmark protein design tasks. Results reported for comparison partners are taken from [20]. Green indicates an improvement in fitness relative to the wild-type (x_{WT}) sequence. **Bold**: the best per each task. Underline: second-best.

Algorithm	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
CMA-ES	0.000±0.000	-6.857±0.257	-0.429±0.252	1.972±0.135	-1.337±0.021	0.553±0.038	0.037±0.01	0.135±0.178
DyNaPPO	0.009±0.018	-3.683±0.575	3.924±0.883	3.550±0.012	-0.007±0.015	0.783±0.036	0.067±0.008	2.796±0.059
BO	0.667±0.024	0.168±0.056	7.442±0.242	3.584± 0.007	0.026±0.003	0.814±0.081	0.682±0.369	2.883±0.069
PEX	0.665±0.022	0.248±0.007	8.099±0.017	3.603± 0.003	0.037±0.001	1.499±.343	1.232±0.000	2.991±0.001
AdaLead	0.683±0.037	0.235±0.002	8.034±0.036	3.581±0.003	0.038±0.001	**1.978±0.188**	1.228±0.002	2.985±0.002
CbAS	0.000±0.000	-8.202±0.032	-0.569±0.092	1.858±0.067	-1.492±0.035	0.351±0.043	0.019±0.002	-0.056±0.003
GFN-AL	0.000±0.000	-7.853±0.270	0.160±0.228	2.004±0.022	-1.164±0.118	0.507±0.025	0.027±0.020	0.271±0.443
GFN-AL-δCS	0.686±0.021	0.203±0.005	7.930±0.055	3.589 ±0.006	0.033±0.001	1.297± 0.337	0.701±0.148	2.984±0.002
LatProtRL	0.593±0.018	0.224±0.000	7.902±0.086	3.590±0.003	0.020±0.000	1.122±0.152	1.229 ±0.000	2.983±0.000
MLDE	0.555±0.000	0.241±0.003	7.934±0.077	3.596±0.003	0.038±0.002	0.896±0.015	1.229±0.000	2.984±0.003
ProSpero	0.720±0.027	0.246±0.006	8.114±0.037	3.617±0.002	0.043±0.002	1.527±0.254	1.231±0.002	2.993±0.003
SILO	**0.747 ± 0.014**	**0.262 ± 0.004**	**8.169 ± 0.052**	**3.619 ± 0.002**	**0.045 ± 0.003**	1.894 ± 0.348	**1.233 ± 0.003**	**2.997 ± 0.003**

Table 34: Mean fitness of top-100 sequences generated by each method. Results reported for comparison partners are taken from [20]. Reported values are the mean and standard deviation over 5 runs. **Bold**: the best per each task. Underline: second-best.

Method	AAV	AMIE	E4B	GFP	LGK	Pab1	TEM	UBE2I
CMA-ES	0.000 ± 0.000	-8.317 ± 0.029	-1.009 ± 0.029	1.593 ± 0.008	-1.538 ± 0.008	0.232 ± 0.012	0.013 ± 0.000	-0.072 ± 0.004
DynaPPO	0.000 ± 0.000	-6.493 ± 0.155	0.574 ± 0.148	2.064 ± 0.068	-1.020 ± 0.045	0.481 ± 0.013	0.027 ± 0.002	1.600 ± 0.101
BO	0.618 ± 0.010	-0.849 ± 0.474	5.909 ± 0.785	3.538 ± 0.036	-0.017 ± 0.020	0.510 ± 0.047	0.606 ± 0.352	2.695 ± 0.148
PEX	0.620 ± 0.017	0.238 ± 0.004	7.948 ± 0.046	3.597 ± 0.003	0.033 ± 0.001	1.307 ± 0.258	1.227 ± 0.002	2.987 ± 0.001
AdaLead	0.644 ± 0.031	0.229 ± 0.001	7.846 ± 0.040	3.563 ± 0.007	0.037 ± 0.001	1.836 ± 0.266	1.201 ± 0.002	2.976 ± 0.003
CbAS	0.000 ± 0.000	-8.361 ± 0.025	-0.820 ± 0.068	1.666 ± 0.021	-1.659 ± 0.023	0.162 ± 0.082	0.010 ± 0.001	-0.072 ± 0.003
GFN-AL	0.000 ± 0.000	-8.268 ± 0.010	-0.415 ± 0.091	1.776 ± 0.009	-1.345 ± 0.037	0.276 ± 0.036	0.015 ± 0.001	0.172 ± 0.396
GFN-AL-δCS	0.648 ± 0.020	-0.244 ± 0.137	7.653 ± 0.136	3.569 ± 0.009	0.024 ± 0.004	1.070 ± 0.113	0.192 ± 0.027	2.968 ± 0.006
LatProtRL	0.563 ± 0.009	0.217 ± 0.001	7.562 ± 0.060	3.582 ± 0.003	0.019 ± 0.000	0.888 ± 0.072	1.222 ± 0.000	2.975 ± 0.001
MLDE	0.555 ± 0.000	0.231 ± 0.004	7.843 ± 0.122	3.591 ± 0.003	0.036 ± 0.002	0.877 ± 0.024	1.131 ± 0.021	2.975 ± 0.005
ProSpero	0.679 ± 0.025	0.236 ± 0.007	8.017 ± 0.054	3.613 ± 0.002	0.040 ± 0.002	1.401 ± 0.202	1.176 ± 0.029	2.987 ± 0.003
SILO	**0.727 ± 0.014**	**0.259 ± 0.005**	**8.148 ± 0.058**	**3.617 ± 0.001**	**0.044 ± 0.002**	**1.856 ± 0.355**	**1.232 ± 0.001**	**2.996 ± 0.002**

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: The abstract and introduction accurately reflect SILO's methodological contribution as a trajectory-level protein sequence optimization approach combining stochastic beam search without replacement (SBS), proxy-guided candidate selection complemented by an auxiliary alanine-scan-inspired fitness score (AFS), and iterative policy adaptation through self-improvement imitation learning (SIL). The empirical claims are supported by evaluations across eight protein fitness landscapes in subsection 4.2, including paired statistical comparisons that identify significant improvements on two landscapes for maximum fitness and four for top-100 mean fitness after Holm correction. Additional low-data and noisy-proxy experiments in subsections 4.3 4.4 and assess performance under selected challenging conditions, without claiming general robustness or convergence beyond the evaluated oracle budgets.
   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: The limitations of our work are mentioned in subsection C.2.
   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: We do not include any theoretical results or proofs in this work.
   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: We provide a detailed explanation regarding our action space, policy architecture, and learning algorithm in section 3. Additionally, we also provide code to reproduce the results of experiments in subsections 4.2, 4.3, and 4.4 under the github repository: `https://github.com/grimmlab/SILO.git`.

   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 code to reproduce the results of experiments in subsections 4.2, 4.3, and 4.4 and the datasets used to train the proxies under the github repository: `https://github.com/grimmlab/SILO.git`. Moreover, details regarding datasets and oracles used, proxy training, and implementation of baselines is discussed.

   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 setup of our experiments is discussed in subsections 4.2, 4.3, and 4.4. We further provide information about the datasets and oracles used, proxy training, and implementation of baselines in section B.
   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 mean and standard deviation over five independent runs for all experiments in section 4. Results are presented as mean $\pm$ standard deviation in tables, and as shaded regions in figures to show variability across 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 discuss the details about compute workers and execution time in subsection C.1.
   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: Our work conforms 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: We discuss the broader impacts of our work in subsection C.3.

    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: This work does not involve the release of models or datasets that pose a high risk of misuse.

    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: All asset owners are properly credited, and licenses explicitly mentioned and respected.

    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: [Yes]
    Justification: We will release documented code implementing our method under the GPL-3.0 license.
    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 work does not involve crowdsourcing or 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 work does not involve 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: This work does not involve LLMs as an important, original, or non-standard component of the core methodology.
    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.