PROEDIT: ONE-STEP PROTEIN OPTIMIZATION WITH DIFFUSION PROTEIN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Protein design often begins downstream optimization from a complete candidate sequence, yet reward-guided optimization of diffusion-pretrained protein models typically retains the native iterative denoising process. We ask whether this generative trajectory is necessary once a candidate already exists. We introduce PROEDIT, which repurposes a diffusion-pretrained protein language model as a target-specific protein sequence optimizer. Rather than optimizing through the native denoising process, the editor conditions on the complete parent sequence and samples a complete edited sequence as a single parallel action. This policy admits explicit action likelihoods and an exact conditional KL to a frozen reference policy, enabling direct reward learning without reconstructing diffusion trajectories. We train the editor from black-box structural rewards using parent-wise group-relative reinforcement learning, where candidates sampled from the same parent provide the relative optimization signal. On protein-binder optimization against PD-L1 and IFNA2, PROEDIT substantially outperforms native diffusion-rollout GRPO in reward and iPTM while reducing training GPU-hours by a factor of approximately 27 under matched reward-evaluation and optimizer-update budgets, and also outperforms reward-weighted supervised training. Most of the improvement is already present in raw policy proposals before reward-based selection, and the gains persist under independent structure predictors. These results show that diffusion pretraining can provide a strong initialization for downstream protein optimization without prescribing the decision process used to optimize complete candidate sequences.

1 INTRODUCTION

Modern protein-design methods can generate increasingly strong structures and sequences under structural and functional constraints (Watson et al., 2023; Dauparas et al., 2022). In many design workflows, however, generation is only the beginning. Once a candidate protein has been obtained, the downstream problem changes: rather than generating a sequence from scratch, the goal is to improve an existing candidate with respect to a task-specific objective while exploiting useful information already present in the sequence. This setting raises a natural optimization problem: given a complete parent sequence, how can we efficiently propose higher-quality variants?

Diffusion protein language models provide powerful generative priors for protein sequences (Wang et al., 2024), and recent work has extended diffusion models from generation toward goal-directed design in several ways. Reward-guided fine-tuning can adapt diffusion models to downstream objectives (Wang et al., 2025a; Su et al., 2026); inference-time methods can repeatedly refine samples using external reward information (Uehara et al., 2025); and DPLM-Evo explicitly models substitution, insertion, and deletion operations for protein evolution and post-editing (Wang et al., 2026a). Together, these advances demonstrate that pretrained diffusion models can serve not only as generative models, but also as powerful substrates for protein optimization.

Despite their different objectives and optimization algorithms, these approaches largely retain a sequential generative process as the downstream decision mechanism. This choice is natural when generation begins from noise or masking, but it is not obviously necessary when optimization begins from a complete candidate sequence. In this setting, the parent already provides global sequence context, while the quantity ultimately being optimized is the quality of the resulting edited sequence

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rather than the trajectory used to produce it. This raises a more basic question: *must the policy used for downstream protein optimization inherit the generative process used during pretraining?*

To address this question, we introduce PROEDIT, a full-context editing policy initialized from diffusion pretraining. Rather than executing the pretrained model's native iterative generation process, PROEDIT observes the complete parent sequence and samples an edited sequence in a single parallel step. We train this policy directly from black-box structural feedback using parent-wise group-relative reinforcement learning (Shao et al., 2024): multiple candidates sampled from the same parent are evaluated by a structural oracle, and their relative outcomes provide the learning signal. This formulation learns the downstream editing policy itself instead of repeatedly performing optimization through the pretrained generative process. On held-out protein-binder design, reward training substantially improves both the optimization objective and predicted interaction quality over the corresponding pretrained editing policy. Our evaluation further separates improvements encoded in the learned proposal policy from those obtained through additional inference-time selection and refinement. These results support a broader principle: the generative process used for pretraining can provide a powerful initialization without prescribing the decision process used for downstream protein optimization.

Our contributions are:

- **A direct, full-context policy for downstream protein optimization.** We decouple downstream reward optimization from the native denoising process of diffusion pretraining. Given a complete parent sequence, PROEDIT samples a complete edited sequence as a single parallel action, turning candidate optimization into learning an amortized editing policy.
- **Direct reward learning with tractable policy quantities.** We train the editor from black-box structural rewards using parent-wise group-relative reinforcement learning. The factorized policy provides explicit action likelihoods and an exact conditional sequence KL to a frozen reference policy without reconstructing denoising trajectories. We characterize its reference-regularized objective, distributional restriction, and local connection between residuewise updates and sequence-level optimization.
- **Effective and compute-efficient learned protein optimization.** Across two binder-design tasks, PROEDIT improves reward and iPTM over pretrained editing and matched-budget reward-trained baselines while substantially reducing training cost relative to native diffusion rollouts. Controlled analyses further distinguish improvements learned by the policy from gains due to inference-time search.

2 RELATED WORK

Protein generative modeling and editing. Generative protein models have increasingly moved beyond autoregressive sequence generation toward diffusion-based modeling of protein sequences and structures. EvoDiff introduced discrete diffusion for protein sequence generation (Alamdari et al., 2023), while DPLM and DPLM-2 developed diffusion protein language models that combine generative modeling with learned protein representations (Wang et al., 2024; 2025b). More recently, DPLM-Evo reformulated protein diffusion around explicit substitution, insertion, and deletion operations, enabling variable-length generation and evolutionary-style post-editing (Wang et al., 2026a). These works establish diffusion models as flexible priors for both protein generation and editing. Protein Editor builds on this direction but focuses on learning a task-specific editing policy for existing candidates using downstream reward.

Reward-guided optimization of generative models. A growing body of work adapts pretrained generative models toward task-specific biological or molecular objectives. DRAKES optimizes discrete diffusion models by differentiating through relaxed trajectories (Wang et al., 2025a), RL-DIF applies reinforcement learning to categorical diffusion for protein design (Ektefaie et al., 2024), and autoregressive protein language models have similarly been fine-tuned against external reward signals (Stocco et al., 2024). VIDD formulates reward-guided diffusion fine-tuning as iterative policy distillation using value-guided target policies (Su et al., 2026). Related work on reinforcement learning for diffusion language models has explored masked-state policy gradients, reconstructed denoising trajectories, and alternative policy parameterizations (Zhao et al., 2026; Wang et al., 2026b; Ni et al., 2026). A complementary line of work performs optimization at inference time rather

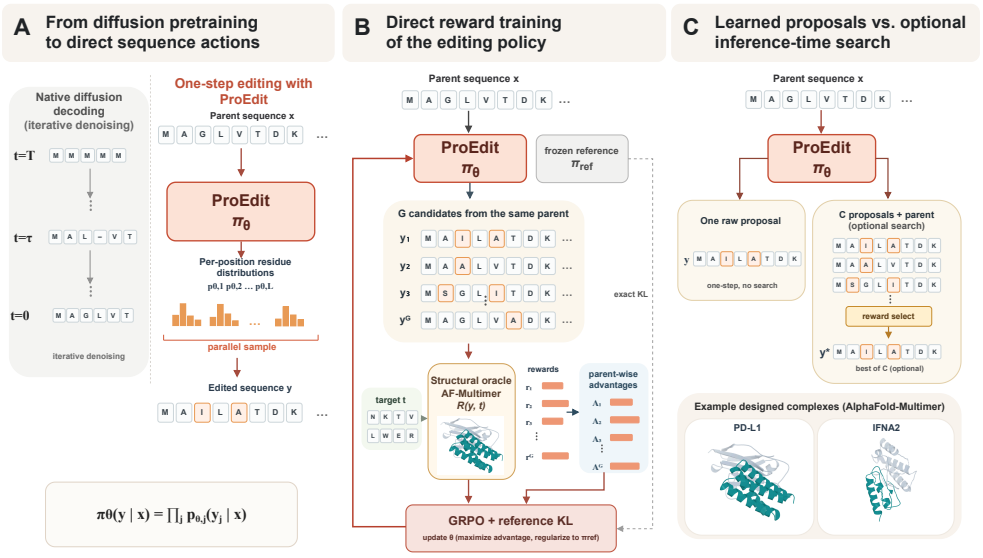


Figure 1: **Overview of PROEDIT. A**, DPLM-2 diffusion pretraining is repurposed as a full-context one-step editing policy that directly samples a complete edited sequence from the parent. **B**, the editing policy is trained from structural rewards using parent-wise GRPO with a frozen reference policy. **C**, the trained policy directly produces one-step proposals, with optional best-of-C reward-based selection at inference time.

than changing model parameters; RERD/ProDifEvo, for example, repeatedly applies reward-guided noising and denoising to improve generated samples (Uehara et al., 2025). Protein Editor is closest to the reward-learning line of work, but studies a different downstream policy: a parent-conditioned one-step editor whose optimization is separated from the pretrained generative trajectory.

3 PROTEIN EDITOR

3.1 DIRECT SEQUENCE ACTIONS

Diffusion pretraining specifies a generative process, whereas downstream optimization evaluates the quality of the resulting candidate sequence. We therefore decouple the policy used for reward optimization from the pretrained denoising process and directly parameterize a distribution over complete sequence actions.

For a fixed design target, let $x \sim \mathcal{D}$ be a parent sequence of length L_x over the amino-acid alphabet $\mathcal{A}$, and let $\mathcal{R}(y)$ be the structural reward of a candidate sequence y. The target enters through the reward, and a separate editing policy is trained for each target. Training therefore requires a pool of parent sequences and a target-specific reward function. The overall setting is candidate optimization over existing parents supplied by an upstream generative pipeline.

The editor is parameterized by a diffusion-pretrained protein language model with weights θ. It takes the complete, unmasked parent and computes all residue distributions $p_{\theta,j}(\cdot \mid x)$ in one forward pass. Sampling from these distributions independently gives the sequence policy

$$\pi_\theta(y \mid x) = \prod_{j=1}^{L_x} p_{\theta,j}(y_j \mid x), \qquad y \in \mathcal{A}^{L_x}. \tag{1}$$

The output preserves length, and each position can retain or replace its parent residue. Its exact log-probability under this sampling policy is

$$\log \pi_\theta(y \mid x) = \sum_{j=1}^{L_x} \log p_{\theta,j}(y_j \mid x). \tag{2}$$

Treating the parent as context and the complete candidate sequence as one action gives the contextual-bandit objective over a fixed parent distribution:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[\mathcal{R}(y)]. \tag{3}$$

Thus, both reward optimization and proposal generation operate on the same direct sequence policy, rather than on the terminal marginal of the pretrained model's native iterative decoder.

3.2 Reference-Regularized Optimization

Having defined optimization directly over complete sequence actions, we next ask what output distribution an ideal reward-optimized policy should represent. For a fixed parent, this question can be posed entirely over terminal sequences, without specifying a denoising trajectory.

Let π_{ref} be the editing policy at initialization, held fixed during training, and let $q(\cdot \mid x)$ denote an arbitrary distribution over candidate sequences in $\mathcal{A}^{L_x}$. To favor high-reward sequences while remaining close to the pretrained initialization, define $K_x(q) = D_{\text{KL}}(q(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x))$, and consider the idealized objective

$$F_{x,\lambda}(q) = \mathbb{E}_{y\sim q}[\mathcal{R}(y)] - \lambda K_x(q), \qquad \lambda > 0. \tag{4}$$

Proposition 1 (Regularized target)**.** *For finite rewards and a strictly positive reference on $\mathcal{A}^{L_x}$, the maximizer over all output distributions is*

$$q^*_{x,\lambda}(y) = \frac{\pi_{\text{ref}}(y \mid x)\exp(\mathcal{R}(y)/\lambda)}{Z_{x,\lambda}}, \tag{5}$$

where $Z_{x,\lambda}$ normalizes the distribution. For any q, $F_{x,\lambda}(q^*_{x,\lambda}) - F_{x,\lambda}(q) = \lambda D_{\text{KL}}(q\|q^*_{x,\lambda})$.

This characterization (Wang et al., 2025a; Su et al., 2026) shows that the ideal regularized solution is itself a distribution over complete output sequences: reward exponentially reweights the reference policy, without requiring a particular denoising trajectory. This supports optimizing a direct terminal-sequence policy rather than treating the pretrained generative process as part of the downstream objective. Our one-step policy further restricts the output distribution to the factorized form

$$q(y \mid x) = \prod_{j=1}^{L_x} q_j(y_j \mid x). \tag{6}$$

This restriction can prevent exact representation of correlations or multiple coordinated modes in $q^*_{x,\lambda}$. Among such factorized distributions, the corresponding gap in the ideal regularized objective is characterized by

$$\lambda \inf_q D_{\text{KL}}\left(q \,\|\, q^*_{x,\lambda}\right),$$

where the infimum is taken over distributions of the form in Eq. 6. Importantly, this restriction limits distribution modeling rather than point optimization: a factorized distribution can still concentrate on any individual high-reward sequence. Appendix A.1 gives the derivation and discusses the distinction between distributional expressivity and neural learnability.

The product structure also makes the conditional sequence KL exact:

$$K_x(\pi_\theta) = \sum_{j=1}^{L_x} D_{\text{KL}}(p_{\theta,j}(\cdot \mid x) \,\|\, p_{\text{ref},j}(\cdot \mid x)). \tag{7}$$

Each categorical KL is evaluated by summing over the 20 amino acids.

3.3 Learning from Sequence Rewards

We now turn from the distribution-level objective to the practical reward-learning update. Using the log-derivative identity, the gradient of the expected sequence reward is

$$\nabla_\theta J(\theta) = \mathbb{E}_{x,y}\left[\mathcal{R}(y)\sum_{j=1}^{L_x}\nabla_\theta \log p_{\theta,j}(y_j \mid x)\right]. \tag{8}$$

Each residue is updated through its own log-probability gradient, while sharing the reward of the complete sequence; no reward derivative or additive decomposition across residues is required.

In practice, we estimate this sequence-level learning signal using group-relative rewards from multiple candidates generated for the same parent. For each parent x, a behavior policy π_{old} samples G candidates $y_{1:G}$. Following GRPO (Shao et al., 2024), their rewards define the advantages

$$A_g = \frac{\mathcal{R}(y_g) - \bar{r}_x}{\max(s_x, \epsilon)}, \tag{9}$$

where $\bar{r}_x$ and s_x are the group mean and standard deviation computed with divisor G. Grouping by parent compares alternative actions in the same context; A_g weights all residues of candidate g. For a sampled residue, the current-to-behavior ratio is

$$\rho_{g,j}(\theta) = \frac{p_{\theta,j}(y_{g,j} \mid x)}{p_{\text{old},j}(y_{g,j} \mid x)}. \tag{10}$$

Define the GRPO clipped term $c_\delta(\rho, A) = \min\{\rho A, \text{clip}(\rho, 1-\delta, 1+\delta)A\}$ (Schulman et al., 2017; Shao et al., 2024). With $\widehat{\mathbb{E}}$ denoting the arithmetic mean over collected parent groups, training minimizes

$$\mathcal{L}(\theta) = \widehat{\mathbb{E}}\left[-\frac{1}{GL_x} \sum_{g=1}^{G} \sum_{j=1}^{L_x} c_\delta(\rho_{g,j}(\theta), A_g) + \frac{\beta}{L_x} K_x(\pi_\theta) \right]. \tag{11}$$

Here β controls the strength of reference regularization. Equation 4 provides a distribution-level interpretation of the reward–reference trade-off, whereas Eq. 11 is the standardized and clipped objective used for practical optimization.

Although the practical objective clips residuewise probability ratios, its local update is consistent with the complete-sequence policy defined in Eq. 1.

Proposition 2 (Local sequence-gradient consistency)**.** *For a fixed candidate y and advantage A, let $\ell_x(\theta) = -L_x^{-1} \sum_j c_\delta(\rho_j(\theta), A)$. For $\delta > 0$, at the behavior parameters,*

$$\nabla_\theta \ell_x(\theta)|_{\theta_{\text{old}}} = -\frac{A}{L_x} \nabla_\theta \log \pi_\theta(y \mid x)|_{\theta_{\text{old}}}. \tag{12}$$

The identity is local and conditional on the sampled advantages; the tokenwise and joint sequence-ratio surrogates differ under finite updates (Appendix A.3).

Training alternates between collecting candidate groups and updating the policy on each collection for E passes. Candidates, advantages, and behavior probabilities remain fixed during these passes; current probabilities and reference KL are evaluated under the original parent contexts. Algorithm 1 gives the procedure.

Inference. The trained policy proposes $y \sim \pi_\theta(\cdot \mid x)$ in one forward pass. Optional reward-assisted selection returns the highest-reward sequence among C independent proposals and the parent. We evaluate unfiltered proposals separately from selected outputs.

4 Experiments

4.1 Experimental Setup

Tasks and datasets. We evaluate ProEdit for protein-binder optimization against two targets, PD-L1 and IFNA2. For each target, we construct parent binders for optimization using the RFdiffusion–ProteinMPNN design pipeline (Watson et al., 2023; Dauparas et al., 2022): RFdiffusion first generates target-conditioned binder backbones, followed by ProteinMPNN sequence design. For each target, we use 5,000 sequences for training, 50 sequences for validation, and 200 sequences for testing. Detailed target definitions, dataset construction, and dataset information are provided in Appendix D.1.

Reward and evaluation metrics. Following the protein-binder setting of VIDD (Su et al., 2026), we use AlphaFold-Multimer (AF-M) (Evans et al., 2021) to provide the structural reward during training. For a candidate binder y, we use the composite reward

$$\mathcal{R}(y) = \text{iPTM}(y) + 0.1\ \text{pLDDT}_B(y) + 0.02\,\mathcal{R}_{\text{gyr}}(y), \tag{13}$$

Table 1: **Main binder-editing results on PD-L1 and IFNA2.** Budget denotes the number of training reward evaluations / optimizer updates. All methods are evaluated with the same AlphaFold-Multimer reward.

Method	Budget	PD-L1			IFNA2		
		Reward ↑	iPTM ↑	pLDDT ↑	Reward ↑	iPTM ↑	pLDDT ↑
Original RFdiffusion + ProteinMPNN	–	0.1796	0.1876	0.6426	0.2114	0.2308	0.6460
Frozen DPLM2	0 / 0	0.2597	0.2375	0.5021	0.2532	0.2420	0.5255
FullRollout-GRPO	204.8K / 3.2K	0.5107	0.4941	0.4982	0.2992	0.2742	0.4994
VIDD (Su et al., 2026)	640K / 10K	0.4324	0.3973	0.5841	0.5964	0.5498	0.5045
BindCraft (Pacesa et al., 2025)	–	0.2914	0.2857	0.6795	0.3271	0.3305	**0.6736**
BoltzGen (Stark et al., 2025)	–	0.2873	0.2446	0.5823	0.3054	0.2690	<u>0.6600</u>
PROEDIT (RW-SFT)	409.6K / 6.4K	0.9079	0.8393	0.6480	0.8509	0.7894	0.5358
PROEDIT (GRPO, 3.2K)	204.8K / 3.2K	<u>0.9332</u>	<u>0.8519</u>	<u>0.7794</u>	<u>0.8724</u>	<u>0.8076</u>	0.5418
PROEDIT (GRPO, 6.4K)	409.6K / 6.4K	**0.9787**	**0.8827**	**0.9225**	**0.9434**	**0.8707**	0.5944

where $\mathcal{R}_{\text{gyr}}$ is a radius-of-gyration reward related to structural compactness; its full definition is provided in Appendix C. We use the composite reward and iPTM as the primary evaluation metrics, with binder pLDDT as a complementary measure of structural confidence. To test whether the observed improvements depend on the AF-M model used for training, we further perform independent structure prediction on the fixed final outputs using Protenix-v1 (Team et al., 2026) and Boltz-2 (Passaro et al., 2025).

Sequence diversity. Because binder lengths vary, we use containment-aware sequence families rather than positional Hamming distance as the primary diversity representation. We report exact uniqueness, the effective number of 90%-identity families $N_{\text{eff},90}$, and the largest-family fraction; complete definitions and additional diagnostics are provided in Appendix E.1.

Baselines. PROEDIT and its primary DPLM-2 comparisons are based on the 650M-parameter DPLM-2 model (Wang et al., 2025b). **Frozen DPLM2** uses the same one-step parent-conditioned policy without reward training. **Editor-RW-SFT** uses the same one-step policy, candidate sampling, structural reward, and reference regularization, and is trained with reward-weighted maximum likelihood. **FullRollout-GRPO** applies GRPO through DPLM-2's native 500-step diffusion decoder. We further compare against the official implementations of VIDD (Su et al., 2026), BindCraft (Pacesa et al., 2025), and BoltzGen (Stark et al., 2025). Detailed implementation details for these baselines are provided in Appendix D.2.

Evaluation protocol. Unless otherwise specified, for each test parent, PROEDIT independently generates $C = 8$ candidates and includes the original parent in the candidate set, selecting the sequence with the highest composite reward as the final output. We also report the mean metrics over all generated candidates (raw candidate mean) to characterize the average quality of model-generated candidates. Training budgets, sampling parameters, and other implementation details are provided in Appendices B and D.1.

4.2 DIRECT REWARD TRAINING FOR ONE-STEP BINDER EDITING

We first ask whether direct reward training can teach a diffusion-pretrained protein model to perform effective parent-conditioned one-step editing. Table 1 summarizes the main results on PD-L1 and IFNA2, including controlled comparisons sharing the DPLM-2 backbone and external protein-design methods evaluated under a common AF-M evaluator. Frozen DPLM2 exhibits only limited editing ability on either task, whereas reward-trained PROEDIT achieves substantially stronger reward and iPTM. More importantly, at the same 204.8K / 3.2K training budget, PROEDIT reaches iPTM values of 0.852 and 0.808 on PD-L1 and IFNA2, compared with 0.494 and 0.274 for FullRollout-GRPO trained through the native diffusion trajectory. This result indicates that defining downstream reward optimization directly over the parent-conditioned sequence action is substantially more effective than retaining the multi-step diffusion generation process used during pretraining.

At the final training budget, GRPO also achieves higher reward and iPTM than the otherwise matched RW-SFT objective on both targets, showing that the difference cannot be explained by reward-weighted sequence fitting alone. VIDD, BindCraft, and BoltzGen provide additional external

reference points under the common AF-M evaluator; because they use different models and generation procedures, we do not treat them as controlled comparisons.

To test whether these results depend on the AF-M model used for training, we fix the final selected sequences and independently rescore them with Protenix-v1 (Team et al., 2026) and Boltz-2 (Passaro et al., 2025), without reselection under either predictor. As shown in Table 2, PROEDIT obtains the highest iPTM within this common cohort under both independent predictors and on both targets. The advantage observed under AF-M therefore persists under independent structure predictors rather than appearing only as a reward-model-specific score.

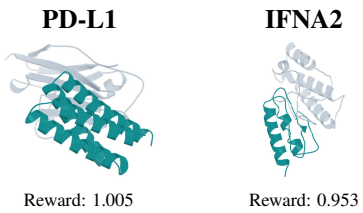


Figure 2: **Exemplar PROEDIT binding complexes.**

Table 2: **iPTM under independent structure predictors.** Fixed AF-M-selected outputs are rescored with Protenix-v1 (Prot-1) and Boltz-2 without reselection.

	PD-L1		IFNA2	
Method	Prot-1	Boltz-2	Prot-1	Boltz-2
Original designs	0.576	0.801	0.526	0.709
Frozen DPLM2	0.512	0.818	0.447	0.678
FullRollout-GRPO	0.815	0.901	0.460	0.657
PROEDIT (RW-SFT)	0.888	0.941	0.528	0.666
PROEDIT (GRPO)	**0.944**	**0.952**	**0.805**	**0.739**

4.3 CHARACTERIZING ONE-STEP EDITING

We next examine what is learned by the one-step policy itself, including proposal quality, dependence on parent context, sequence-family diversity, and training efficiency. Figure 3 separates raw proposal quality from best-of-8 selection and parent fallback. For the final PROEDIT, best-of-8 adds only 0.013 reward on PD-L1 and 0.011 on IFNA2 over the raw candidate mean, with no parent fallback on either target. Frozen DPLM2 benefits substantially more from selection. Thus, reward training improves the proposal distribution itself rather than relying primarily on inference-time search.

Replacing the natural parent with either a length-matched random sequence or an all-mask input substantially reduces reward and iPTM on both targets. The context controls in Table 3 rule out the degenerate explanation that the target-specific editor simply ignores its input and behaves as an unconditional target-specific generator.

Table 3: **Parent-context controls.**

	PD-L1		IFNA2	
Input	Reward	iPTM	Reward	iPTM
Natural parent	**0.9787**	**0.8827**	**0.9434**	**0.8707**
Random sequence	0.2789	0.2491	0.4095	0.3754
All-mask	0.4342	0.3957	0.2809	0.2522

Table 4 further shows a quality–diversity trade-off: PROEDIT-GRPO-3.2K retains full sequence-family coverage on both targets while achieving the highest reward at this level of diversity; further optimization to 6.4K increases reward at the cost of reduced diversity.

Figure 4 compares raw proposal reward as a function of training GPU-hours. At the matched 204.8K reward-evaluation / 3.2K optimizer-update budget, PROEDIT uses approximately 27.4 H200 GPU-hours averaged across the two targets, compared with 738.3 GPU-hours for FullRollout-GRPO—a roughly 27× difference. The direct formulation therefore reaches substantially stronger proposals at far lower training cost.

4.4 EDITING AND REFINING DE NOVO GENERATED DESIGNS

Finally, we ask whether PROEDIT can further edit designs produced by other *de novo* generation methods. We apply both one-step and repeated editing to outputs from VIDD, FullRollout, BindCraft, and BoltzGen.

Because outputs from *de novo* generators can be highly concentrated within a small number of sequence families, averaging all generated slots equally can be influenced by source repetition. We therefore report both slot-weighted and family-balanced reward changes, where the latter weights the observed 90%-containment-identity families equally; the full definition is given in Appendix F. Table 5 shows positive family-balanced gains for all source–target combinations. The two estimates

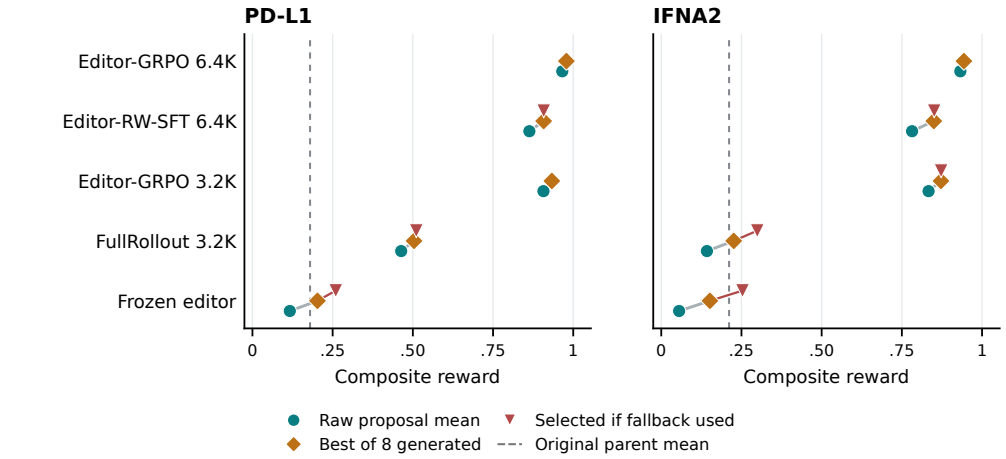


Figure 3: **One-step proposal quality before and after reward-based selection.** Raw is the mean reward over all generated candidates, Best-of-8 is the highest-reward generated candidate for each parent, and final selection additionally allows the original parent to be retained.

Table 4: **Sequence-family diversity of final outputs in the main benchmark.** Distinct is the number of exact distinct sequences among 200 selected outputs; $N_{\text{eff},90}$ is the effective family count at 90% containment identity; Max fam. is the largest-family fraction.

	PD-L1			IFNA2		
Method	Distinct	$N_{\text{eff},90}$	Max fam.	Distinct	$N_{\text{eff},90}$	Max fam.
Original parents	200	200.00	0.5%	200	200.00	0.5%
FullRollout-GRPO	41	1.14	93.5%	123	4.46	44.0%
VIDD (Su et al., 2026)	148	5.77	32.5%	193	38.99	13.5%
ProEdit (RW-SFT)	200	200.00	0.5%	200	200.00	0.5%
ProEdit (GRPO, 3.2K)	200	200.00	0.5%	200	200.00	0.5%
ProEdit (GRPO, 6.4K)	199	73.26	7.0%	173	20.86	14.0%

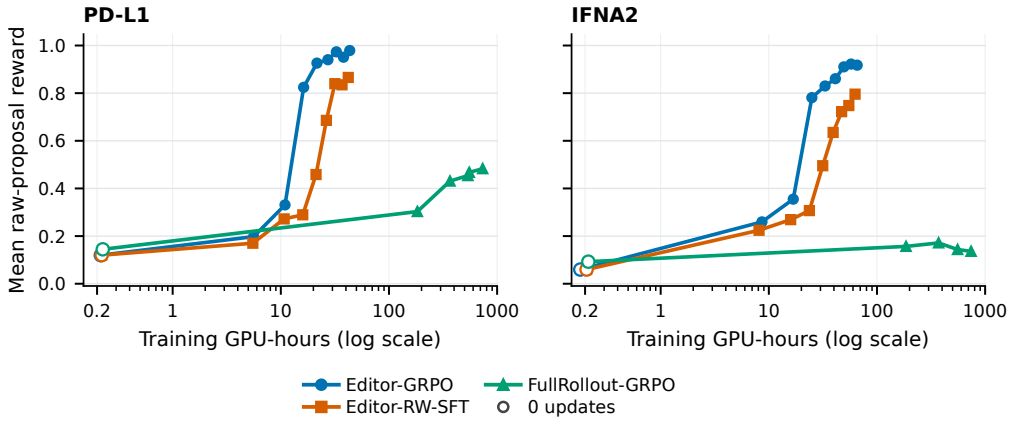


Figure 4: **Raw proposal quality as a function of training compute.** Mean raw-proposal reward on the fixed heldout50 monitoring cohort is plotted against checkpoint-ready H200 GPU-hours on a logarithmic scale.

are similar for sources with broad family coverage, whereas they can differ substantially for highly concentrated source distributions.

We further apply the same one-step policy using reward-accepted repeated editing. Figure 5 shows the resulting family-balanced reward across editing rounds. For most source distributions, the largest gains occur within the first few rounds and then gradually saturate, while the magnitude and rate of refinement remain strongly source-dependent.

Table 5: **Source-family concentration and family-balanced one-step editing.** $N_{\text{eff},90}$ is the effective family count of the source inputs at 90% containment identity. Slot ΔR weights all 200 source slots equally; Family ΔR first averages within each source family and then weights the observed families equally.

	PD-L1			IFNA2		
Source	$N_{\text{eff},90}$	Slot ΔR	Family ΔR	$N_{\text{eff},90}$	Slot ΔR	Family ΔR
VIDD	5.61	+0.102	+0.195	37.81	+0.317	+0.368
FullRollout	1.00	+0.006	+0.005	2.66	+0.030	+0.352
BindCraft	200.00	+0.339	+0.339	200.00	+0.334	+0.334
BoltzGen	84.03	+0.244	+0.279	116.28	+0.555	+0.597

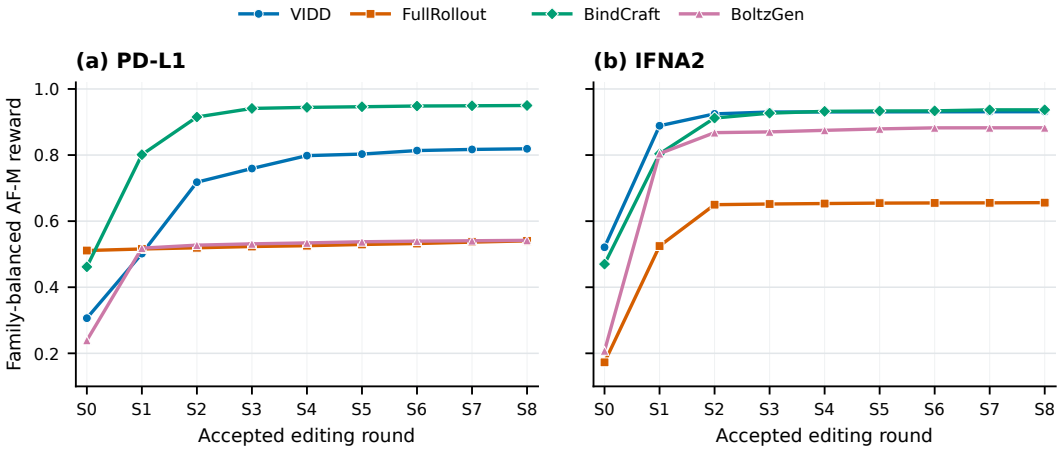


Figure 5: **Family-balanced reward across repeated editing rounds.** S0 denotes the frozen de novo source and S1–S8 the accepted sequences after successive editing rounds. Rewards are averaged with equal weight over the source sequence families.

Together, these results show that PROEDIT transfers beyond its training parent distribution to outputs from multiple other *de novo* generation methods, with repeated editing providing additional refinement for several source distributions. More details are reported in Appendix F.

5 CONCLUSION

We introduced PROEDIT, a parent-conditioned one-step protein editing policy that decouples downstream optimization from the iterative generative process used during diffusion pretraining. By training directly on complete sequence actions with structural rewards, PROEDIT learns high-quality edits that are already present in its direct proposals, substantially outperforming the pretrained editor and reward optimization through the native diffusion trajectory while requiring much less training compute. The learned policy remains strongly conditioned on the parent sequence, preserves substantial sequence-family diversity at strong reward levels, and transfers to designs produced by multiple *de novo* generation methods. Together, these results suggest that the generative process used for pretraining need not prescribe the decision process used for downstream protein optimization; instead, diffusion pretraining can serve as an initialization for a directly learned editing policy.

LIMITATION

A current limitation is that the target enters only through the reward, requiring a separate editor to be optimized for each target. An important next step is to condition the editor explicitly on target sequence or structure context, enabling a single policy to generalize across targets and support broader protein optimization tasks.

AI USE STATEMENT

We used generative AI tools to improve the readability and presentation of the manuscript and to assist with LaTeX formatting. All AI-assisted edits were reviewed by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work is based entirely on computational modeling and model-generated data. It does not involve human subjects, personal or sensitive data, or human-subject experimentation. We do not identify specific ethical concerns arising from the computational experiments conducted in this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed algorithm, training procedure, and evaluation methodology in the main paper and appendices. The appendices additionally document the implementation and experimental settings necessary to reproduce the reported results.

REFERENCES

Sarah Alamdari, Nitya Thakkar, Rianne Van Den Berg, Neil Tenenholtz, Robert Strome, Alan M Moses, Alex X Lu, Nicolò Fusi, Ava P Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *BioRxiv*, pp. 2023–09, 2023.

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

Yasha Ektefaie, Olivia Viessmann, Siddharth Narayanan, Drew Dresser, J Mark Kim, and Armen Mkrtchyan. Reinforcement learning on structure-conditioned categorical diffusion for protein inverse folding. *arXiv preprint arXiv:2410.17173*, 2024.

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

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, et al. One-shot design of functional protein binders with bindcraft. *Nature*, 646(8084):483–492, 2025.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, et al. Boltz-2: Towards accurate and efficient binding affinity prediction. *BioRxiv*, 2025.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Hannes Stark, Felix Faltings, MinGyu Choi, Yuxin Xie, Eunsu Hur, Timothy O'Donnell, Anton Bushuiev, Talip Uçar, Saro Passaro, Weian Mao, et al. Boltzgen: Toward universal binder design. *BioRxiv*, 2025.

Filippo Stocco, Maria Artigues-Lleixa, Andrea Hunklinger, Talal Widatalla, Marc Guell, and Noelia Ferruz. Guiding generative protein language models with reinforcement learning. *arXiv preprint arXiv:2412.12979*, 2024.

Xingyu Su, Xiner Li, Masatoshi Uehara, Sunwoo Kim, Yulai Zhao, Gabriele Scalia, Ehsan Hajiramezanali, Tommaso Biancalani, Degui Zhi, and Shuiwang Ji. Iterative distillation for reward-guided fine-tuning of diffusion models in biomolecular design. In *International Conference on Learning Representations*, volume 2026, pp. 139443–139469, 2026.

Protenix Team, Yuxuan Zhang, Chengyue Gong, Hanyu Zhang, Wenzhi Ma, Zhenyu Liu, Xinshi Chen, Jiaqi Guan, Lan Wang, Yanping Yang, et al. Protenix-v1: Toward high-accuracy open-source biomolecular structure prediction. *bioRxiv*, pp. 2026–02, 2026.

Masatoshi Uehara, Xingyu Su, Yulai Zhao, Xiner Li, Aviv Regev, Shuiwang Ji, Sergey Levine, and Tommaso Biancalani. Reward-guided iterative refinement in diffusion models at test-time with applications to protein and dna design. *arXiv preprint arXiv:2502.14944*, 2025.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Hanchen Wang, and Tommaso Biancalani. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. In *International Conference on Learning Representations*, volume 2025, pp. 47871–47899, 2025a.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Dplm-2: A multimodal diffusion protein language model. In *International Conference on Learning Representations*, volume 2025, pp. 35463–35490, 2025b.

Xinyou Wang, Liang Hong, Jiasheng Ye, Zaixiang Zheng, Yu Li, Shujian Huang, and Quanquan Gu. Towards a generative protein evolution machine with dplm-evo. *arXiv preprint arXiv:2605.00182*, 2026a.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *International Conference on Learning Representations*, volume 2026, pp. 101476–101501, 2026b.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:56729–56762, 2026.

A Analysis of the Sequence Editing Policy

This section collects the distributional and gradient properties used to interpret the direct sequence-action policy. We analyze the conditional policy for a fixed parent x and omit conditioning on x where unambiguous. We assume a finite sequence space, finite rewards, and strictly positive reference-policy probabilities.

A.1 Regularized target and policy restriction

Recall the reference-regularized objective

$$F_{x,\lambda}(q) = \mathbb{E}_{y\sim q}[\mathcal{R}(y)] - \lambda D_{\mathrm{KL}}(q \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x)) \,.$$

Proof of Proposition 1. Write $p_0 = \pi_{\mathrm{ref}}(\cdot \mid x)$, $Z = \sum_y p_0(y) e^{\mathcal{R}(y)/\lambda}$, and $q^*(y) = p_0(y)e^{\mathcal{R}(y)/\lambda}/Z$. Then

$$\lambda D_{\mathrm{KL}}(q\|q^*) = \lambda D_{\mathrm{KL}}(q\|p_0) - \mathbb{E}_q[\mathcal{R}] + \lambda \log Z = \lambda \log Z - F_{x,\lambda}(q). \tag{14}$$

Since the KL divergence is non-negative and vanishes only at $q = q^*$, q^* is the unique maximizer. □

Restricting q to the family $\Pi_{\mathrm{prod}}(x)$ of product distributions yields

$$F_{x,\lambda}(q^*) - \sup_{q\in\Pi_{\mathrm{prod}}(x)} F_{x,\lambda}(q) = \lambda \inf_{q\in\Pi_{\mathrm{prod}}(x)} D_{\mathrm{KL}}(q\|q^*). \tag{15}$$

Thus, the one-step factorization can limit how accurately the full reward-tilted distribution is represented. This is a distributional expressivity restriction rather than a restriction on assigning high probability to an individual sequence: a product distribution can concentrate on any single sequence. It also does not by itself guarantee learnability under a shared neural parameterization.

A.2 Exact likelihood and reference KL

For the product policy in Eq. 1,

$$\log \pi_\theta(y \mid x) = \sum_j \log p_{\theta,j}(y_j \mid x).$$

The corresponding conditional sequence KL decomposes exactly:

$$\begin{aligned} D_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}}) &= \sum_y \pi_\theta(y) \sum_j \log \frac{p_{\theta,j}(y_j)}{p_{\mathrm{ref},j}(y_j)} \\ &= \sum_j \sum_{a\in\mathcal{A}} p_{\theta,j}(a) \log \frac{p_{\theta,j}(a)}{p_{\mathrm{ref},j}(a)}. \end{aligned} \tag{16}$$

The reference penalty can therefore be evaluated exactly from the categorical distributions at each residue position.

A.3 Sequence reward and residuewise updates

For

$$J_x(\theta) = \sum_y \pi_\theta(y \mid x)\mathcal{R}(y),$$

the score-function identity gives

$$\nabla_\theta J_x = \mathbb{E}_{\pi_\theta}\left[\mathcal{R}(Y) \sum_j \nabla_\theta \log p_{\theta,j}(Y_j \mid x)\right]. \tag{17}$$

Thus, each residue receives a likelihood-gradient contribution weighted by the reward of the complete sampled sequence; the reward itself need not decompose across residues.

Proof of Proposition 2. At θ_{old}, each probability ratio equals one and lies inside the clipping interval. Locally,

$$\ell_x(\theta) = -\frac{A}{L_x} \sum_j \rho_j(\theta).$$

Holding the sampled advantage and behavior probabilities fixed,

$$\nabla_\theta \rho_j(\theta)|_{\theta_{\text{old}}} = \nabla_\theta \log p_{\theta,j}(y_j \mid x)|_{\theta_{\text{old}}}.$$

Summing over residues yields Eq. 12. □

The tokenwise and joint sequence-ratio surrogates therefore share the same first-order gradient at the behavior policy, although they differ under finite parameter updates. Tokenwise clipping should not be interpreted as a hard constraint on the joint sequence probability ratio.

B Training and Inference Procedures

B.1 Reward training

Algorithm 1 Reward training of the one-step editing policy

Require: Parent distribution $\mathcal{D}$, reward $\mathcal{R}$, pretrained parameters θ_0, group size G, update passes E

Initialize $\theta \leftarrow \theta_0$ and freeze $\pi_{\text{ref}} = \pi_{\theta_0}$
while training budget remains **do**
  Set $\theta_{\text{old}} \leftarrow \theta$ and sample a parent batch
  Draw G complete sequence actions from $\pi_{\text{old}}(\cdot \mid x)$ for each parent
  Evaluate rewards and compute parent-wise advantages
  for $e = 1, \ldots, E$ **do**
    Recompute current-policy probabilities and reference KL
    Update θ using Eq. 11
  end for
end while
return π_θ

All actual residue positions are treated as action positions over the 20 canonical amino acids; retaining the parent residue is a valid action. Advantages are computed over the complete group of G candidates from the same parent. The token loss is normalized by candidate length and then averaged over candidates, matching Eq. 11. The reference policy remains fixed throughout training.

B.2 Proposal selection and repeated editing

A direct proposal is a single draw from $\pi_\theta(\cdot \mid x)$. In the standard evaluation, C independent proposals are generated from the same parent and the original parent is included as a fallback:

$$y^* \in \operatorname*{arg\,max}_{z \in \{x, y_1, \ldots, y_C\}} \mathcal{R}(z), \qquad y_c \overset{\text{iid}}{\sim} \pi_\theta(\cdot \mid x). \tag{18}$$

The main benchmark uses one editing round. For the repeated-editing analysis in Section 4.4, the accepted sequence becomes the parent of the next round:

$$x^{(r)} \in \operatorname*{arg\,max}_{z \in \{x^{(r-1)}, y_1^{(r)}, \ldots, y_C^{(r)}\}} \mathcal{R}(z), \qquad y_c^{(r)} \overset{\text{iid}}{\sim} \pi_\theta(\cdot \mid x^{(r-1)}). \tag{19}$$

This reward-accepted transition is used only for the repeated-editing experiment and is not part of one-step policy training.

C Structural Reward Details

For the composite reward in Eq. 13, the compactness term is

$$\mathcal{R}_{\mathrm{gyr}}(y) = -\,\mathrm{ELU}\big(R_g(y) - 2.38L_y^{0.365}\big)\,, \tag{20}$$

$$R_g(y) = \sqrt{\frac{1}{L_y}\sum_{j=1}^{L_y}\|\mathbf{r}_j - \bar{\mathbf{r}}\|_2^2 + 10^{-8}}, \qquad \bar{\mathbf{r}} = \frac{1}{L_y}\sum_j \mathbf{r}_j. \tag{21}$$

Here $\mathbf{r}_j$ denotes the predicted binder Cα coordinate and L_y is the number of binder residues. The term penalizes radii above the length-dependent reference value $2.38L_y^{0.365}$ while providing a bounded positive contribution below it.

D Experimental Protocol and Controlled Baselines

D.1 Experimental settings

Following the backbone-to-sequence design workflow of RFdiffusion (Watson et al., 2023), we generate target-conditioned binder backbones with prescribed surface hotspots and lengths of 70–100 residues. ProteinMPNN (Dauparas et al., 2022) then samples eight binder sequences per backbone at temperature 0.1. Generation covers four hotspot sets for PD-L1 and six for IFNA2. After exact-sequence deduplication, the training pools contain 5,000 PD-L1 and 5,000 IFNA2 candidates. For each target, the fixed 50-sequence validation and 200-sequence test cohorts are disjoint subsets of an independently generated 500-sequence pool.

Table 6: **Dataset and target definitions.**

Target	Target residues	Train	Validation	Test
PD-L1	115	5,000	50	200
IFNA2	165	5,000	50	200

Training budgets are reported as reward evaluations / optimizer updates, consistent with the main text.

Table 7: **Training recipes used in the controlled comparisons.**

Recipe	GPUs	Parents/collection	G	Update passes	Reward evals.	Updates
ProEdit GRPO, 3.2K	8	16	8	2	204.8K	3.2K
ProEdit GRPO, 6.4K	8	16	8	2	409.6K	6.4K
ProEdit RW-SFT	8	16	8	2	409.6K	6.4K
FullRollout-GRPO	8	8	8	1	204.8K	3.2K

Table 8: **Inference protocols for the controlled DPLM-2 comparisons.**

Setting	ProEdit	FullRollout
Backbone	DPLM-2 650M	DPLM-2 650M
Input	Complete parent sequence	All-mask sequence with matched binder length
Sequence generation	One parallel sequence action	Native 500-step diffusion decoding
Evaluation	$C = 8$ proposals + parent fallback	$C = 8$ terminal sequences + parent fallback
Selection	Composite AF-M reward	Composite AF-M reward

D.2 Controlled baseline objectives

ProEdit (RW-SFT). RW-SFT uses the same DPLM-2 initialization, complete-parent conditioning, candidate collection, structural reward, and reference regularization as the one-step GRPO editor.

Table 9: **Optimization hyperparameters for the controlled DPLM-2 training recipes.**

Parameter	PROEDIT GRPO	PROEDIT RW-SFT	FullRollout-GRPO
Backbone	DPLM-2 650M	DPLM-2 650M	DPLM-2 650M
Optimizer	AdamW	AdamW	AdamW
Learning rate	10^{-6}	10^{-6}	10^{-6}
Weight decay	0.01	0.01	0.01
Gradient norm limit	1.0	1.0	1.0
Reference-KL coefficient	0.02	0.02	0.02
GRPO clipping δ	0.2	–	0.2
Group size G	8	8	8
Update passes / collection	2	2	1
Training sampling	temperature 1.0	temperature 1.0	native $2.0 \to 0.1$ schedule

Table 10: **Structural reward and evaluation settings.**

Setting	Value
AF-M protocol	Binder, `model_1_multimer_v3`
AF-M precision / dropout	bfloat16 / disabled
Additional recycles	0
Binder tensor length	100; padded residues masked
Reward weights (iPTM, pLDDT, compactness)	$(1, 0.1, 0.02)$
Editor evaluation temperature	0.8
Candidates per parent	8
Main selection	Composite reward with original-parent fallback
FullRollout decoding	500 steps, native annealing $2.0 \to 0.1$

It replaces the clipped group-relative objective with reward-weighted maximum likelihood. For parent-wise standardized advantages $A_{1:G}$, detached candidate weights are

$$w_g = G \frac{\exp(A_g/\alpha)}{\sum_{h=1}^{G} \exp(A_h/\alpha)}, \qquad \alpha = 1. \tag{22}$$

The training loss is

$$\mathcal{L}_{\text{RW-SFT}} = -\frac{1}{G} \sum_{g=1}^{G} w_g \left[\frac{1}{L_x} \sum_{j=1}^{L_x} \log p_{\theta,j}(y_{g,j} \mid x) \right] + \frac{\beta}{L_x} K_x(\pi_\theta). \tag{23}$$

RW-SFT therefore isolates the effect of the policy-optimization objective while retaining the same one-step policy class and reward signal.

FullRollout-GRPO. FullRollout-GRPO uses the same pretrained DPLM-2 backbone and structural reward, with the native 500-step diffusion decoder. For each parent x, we sample $G = 8$ trajectories from an all-mask sequence of length L_x.

Let $\mathcal{C}_g$ contain the accepted token actions along trajectory g, including actions followed by remasking, and let $N_g = |\mathcal{C}_g|$. The terminal reward defines the group-relative advantage A_g through Eq. 9, which weights all recorded actions in that trajectory. The training loss is

$$\mathcal{L}_{\text{FullRollout}} = \frac{1}{G} \sum_{g=1}^{G} \frac{1}{N_g} \sum_{(t,j) \in \mathcal{C}_g} \left[-c_\delta\left(\rho_g^{t,j}, A_g\right) + \beta\, \kappa_g^{t,j} \right]. \tag{24}$$

Here $\rho_g^{t,j}$ is the current-to-behavior token probability ratio under the native sampler, c_δ is the clipped term from Section 3.3, and $\kappa_g^{t,j}$ is the categorical reference KL over the 20 amino acids before nucleus filtering. We use $\delta = 0.2$, $\beta = 0.02$, and one optimization pass per collection, holding the recorded states, actions, advantages, and behavior probabilities fixed.

E Sequence Diversity and Computational Analysis

E.1 Containment-aware sequence diversity

For sequences of unequal length, positional Hamming distance can confound terminal truncation with genuine sequence change. We therefore define a containment-aware edit dissimilarity by aligning the complete shorter sequence to the best region of the longer sequence. Terminal overhangs of the longer sequence have zero cost, whereas substitutions and internal insertions or deletions contribute edit cost. The distance is normalized by the length of the shorter sequence.

We cluster exact distinct sequences by complete linkage at containment distance at most 0.1 and refer to the resulting groups as 90%-containment sequence families. After restoring output multiplicities, let n_k be the number of output slots assigned to family k and $N = \sum_k n_k$. We define the effective number of sequence families as

$$N_{\mathrm{eff},90} = \left[\sum_{k=1}^{K_{90}} \left(\frac{n_k}{N} \right)^2 \right]^{-1}. \tag{25}$$

$N_{\mathrm{eff},90}$ equals the observed family count when families are equally represented and decreases as the output distribution becomes concentrated in a smaller number of dominant families.

Table 11: **Containment-aware diversity of main-benchmark outputs.** Distinct counts distinct sequences; D_{cont} is mean pairwise containment dissimilarity; K_{90} is the number of observed 90%-containment families; and $N_{\mathrm{eff},90}$ is their effective number after accounting for family frequencies.

Method	Distinct	D_{cont}	K_{90}	$N_{\mathrm{eff},90}$
PD-L1				
Original parents	200	0.582	200	200.00
Frozen DPLM2	200	0.725	200	200.00
FullRollout-GRPO	41	0.101	14	1.14
VIDD	148	0.333	79	5.77
BindCraft	200	0.662	200	200.00
BoltzGen	198	0.700	185	109.89
ProEdit (RW-SFT)	200	0.516	200	200.00
ProEdit (GRPO, 3.2K)	200	0.403	200	200.00
ProEdit (GRPO, 6.4K)	199	0.247	136	73.26
IFNA2				
Original parents	200	0.635	200	200.00
Frozen DPLM2	200	0.713	200	200.00
FullRollout-GRPO	123	0.418	68	4.46
VIDD	193	0.303	136	38.99
BindCraft	200	0.675	200	200.00
BoltzGen	199	0.677	193	156.25
ProEdit (RW-SFT)	200	0.555	200	200.00
ProEdit (GRPO, 3.2K)	200	0.449	200	200.00
ProEdit (GRPO, 6.4K)	173	0.267	59	20.86

E.2 Training trajectories and computational cost

The fixed 50-parent validation cohort is used to monitor learning trajectories. Raw-proposal reward averages all eight proposals per parent without best-of-eight selection or parent fallback. Figure 4 plots these measurements against checkpoint-ready H200 GPU-hours.

E.3 Representative predicted target–binder complexes

Figures 6 and 7 show representative high-gain held-out examples for the matched DPLM-2 comparisons.

Table 12: **PD-L1 heldout50 monitoring.** Each entry reports mean raw-proposal reward / checkpoint-ready H200 GPU-hours.

Updates	PROEDIT GRPO	RW-SFT	FullRollout-GRPO
0	0.120 / 0.2	0.120 / 0.2	0.145 / 0.2
800	0.198 / 5.6	0.170 / 5.5	0.303 / 183.0
1,600	0.331 / 10.9	0.272 / 10.7	0.431 / 366.5
2,400	0.825 / 16.3	0.289 / 16.0	0.467 / 552.7
3,200	0.926 / 21.6	0.458 / 21.2	0.483 / 735.1
4,000	0.940 / 27.3	0.686 / 26.4	–
4,800	0.973 / 32.6	0.840 / 31.6	–
5,600	0.951 / 38.0	0.835 / 36.8	–
6,400	0.979 / 43.4	0.866 / 42.0	–

Table 13: **IFNA2 heldout50 monitoring.** Each entry reports mean raw-proposal reward / checkpoint-ready H200 GPU-hours.

Updates	PROEDIT GRPO	RW-SFT	FullRollout-GRPO
0	0.060 / 0.2	0.060 / 0.2	0.093 / 0.2
800	0.260 / 8.6	0.224 / 8.1	0.157 / 185.6
1,600	0.355 / 16.8	0.269 / 15.9	0.171 / 371.1
2,400	0.781 / 25.0	0.307 / 23.7	0.143 / 556.4
3,200	0.830 / 33.1	0.495 / 31.5	0.136 / 741.3
4,000	0.861 / 41.2	0.635 / 39.3	–
4,800	0.910 / 49.4	0.722 / 47.1	–
5,600	0.922 / 57.6	0.748 / 54.9	–
6,400	0.918 / 65.6	0.796 / 62.7	–

Table 14: **Training cost at the reported endpoints.**

Target	Method	Reward evals.	Updates	Wall (h)	GPU-h
PD-L1	PROEDIT GRPO	204.8K	3.2K	2.70	21.63
PD-L1	FullRollout-GRPO	204.8K	3.2K	91.90	735.18
PD-L1	PROEDIT GRPO	409.6K	6.4K	5.43	43.42
PD-L1	PROEDIT RW-SFT	409.6K	6.4K	5.26	42.09
IFNA2	PROEDIT GRPO	409.6K	6.4K	8.20	65.63
IFNA2	PROEDIT RW-SFT	409.6K	6.4K	7.84	62.74
IFNA2	FullRollout-GRPO	204.8K	3.2K	92.67	741.34

Table 15: **Measured training-stage time on PD-L1.** Values are mean seconds per completed collection on eight H200 GPUs.

Method	Sampling	AF-M	Policy update	Total
PROEDIT GRPO	0.634	3.693	2.269	6.048
FullRollout-GRPO	21.985	1.829	81.520	103.137



Figure 6: **Representative PD-L1 editing example.** The one-step editor rows share the same parent.



Figure 7: **Representative IFNA2 editing example.**

F De Novo Source Editing

The transfer experiments use fixed cohorts of 200 generated source slots for each method and target. Because some source generators repeatedly produce the same or closely related sequences, we report both slot-weighted and family-balanced statistics.

For an exact source sequence u, let $I(u)$ denote the editing trials that start from that sequence. We first average its paired reward changes,

$$\bar{\Delta}(u) = \frac{1}{|I(u)|} \sum_{i \in I(u)} \left[\mathcal{R}(y_i) - \mathcal{R}(x_i) \right]. \tag{26}$$

If U_k is the set of distinct source sequences in the kth 90%-containment family, the family-balanced reward change is

$$\Delta R_{\mathrm{fam}} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|U_k|} \sum_{u \in U_k} \bar{\Delta}(u). \tag{27}$$

Thus, repeated trials of the same exact source are averaged before distinct sources within a family are averaged, and the observed families then receive equal weight. Slot-weighted means instead give each of the 200 source slots equal weight.

F.1 Source-family composition

Table 16: **Sequence-family composition of the frozen de novo source cohorts.**

Target	Source	Distinct	K_{90}	$N_{\mathrm{eff},90}$
PD-L1	VIDD	146	63	5.61
	FullRollout	28	1	1.00
	BindCraft	200	200	200.00
	BoltzGen	192	174	84.03
IFNA2	VIDD	193	129	37.81
	FullRollout	76	7	2.66
	BindCraft	200	200	200.00
	BoltzGen	199	186	116.28

F.2 Reward-accepted repeated editing

Sources are frozen generated sequences collected before fallback to the original RFdiffusion–ProteinMPNN parent. For BindCraft, we use one generated sequence from each of 200 trajectories, including accepted final designs, rejected MPNN designs, and intermediate trajectory sequences. Reported gains are relative to these generated sources.

For repeated editing, each round samples eight proposals from the current sequence and retains the proposal only if it improves the composite reward. We denote the frozen source by Source, the accepted sequence after one round by R1, and the accepted sequence after eight rounds by R8.

The family-balanced curves in Figure 5 therefore characterize improvement across observed source families, while the slot-weighted values above preserve the empirical frequency of the generated source cohort.

Table 17: **Slot-weighted and family-balanced reward during repeated editing.** Source, R1, and R8 denote the frozen source and the accepted sequences after one and eight reward-accepted rounds, respectively.

		Slot-weighted reward			Family-balanced reward		
Target	Source	Source	R1	R8	Source	R1	R8
PD-L1	VIDD	0.4176	0.5198	0.6818	0.3063	0.5017	0.8190
	FullRollout	0.5037	0.5097	0.5397	0.5111	0.5158	0.5403
	BindCraft	0.4617	0.8011	0.9500	0.4617	0.8011	0.9500
	BoltzGen	0.2384	0.4821	0.5042	0.2392	0.5182	0.5418
IFNA2	VIDD	0.5790	0.8958	0.9329	0.5210	0.8886	0.9311
	FullRollout	0.2259	0.2560	0.2792	0.1728	0.5246	0.6560
	BindCraft	0.4700	0.8036	0.9369	0.4700	0.8036	0.9369
	BoltzGen	0.2044	0.7599	0.8333	0.2077	0.8043	0.8825

F.3 Sequence diversity after repeated editing

Table 18: **Sequence-family diversity before and after R8 editing.** Input and R8 families are clustered from their corresponding sequence cohorts.

Source	Input Distinct	R8 Distinct	Input $N_{\mathrm{eff},90}$	R8 $N_{\mathrm{eff},90}$
PD-L1				
VIDD	146	148	5.61	6.39
FullRollout	28	145	1.00	6.13
BindCraft	200	200	200.00	70.92
BoltzGen	192	197	84.03	47.28
IFNA2				
VIDD	193	183	37.81	25.45
FullRollout	76	186	2.66	7.03
BindCraft	200	194	200.00	28.09
BoltzGen	199	189	116.28	19.80

F.4 Representative de novo source edits

Figures 8 and 9 show representative source-to-editor examples for the four de novo source methods. These examples are qualitative illustrations and do not estimate the frequency of improvement across the corresponding source cohorts.

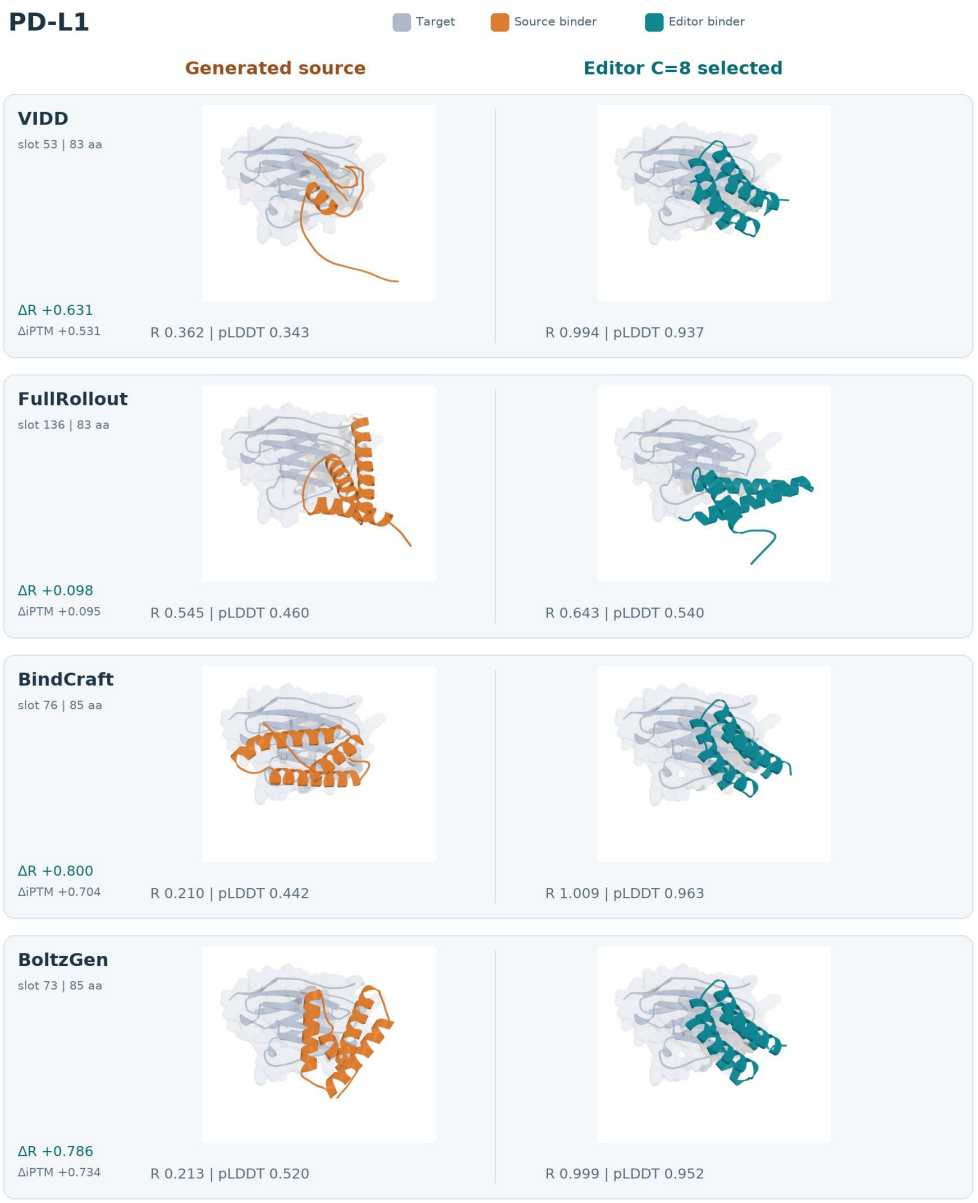


Figure 8: **Representative PD-L1 de novo source edits.** Orange denotes the frozen source binder and teal the selected edited binder.



Figure 9: **Representative IFNA2 de novo source edits.** Colors follow Figure 8.

G Additional Analysis

G.1 Across-parent metric variation

Table 19: **Mean ± standard deviation over the 200 test parents.** The standard deviation describes variation across test parents for each fixed checkpoint.

Method	Updates	Reward	iPTM	pLDDT
PD-L1				
Original designs	–	0.180 ± 0.180	0.188 ± 0.131	0.643 ± 0.130
Frozen DPLM2	0	0.260 ± 0.114	0.237 ± 0.108	0.502 ± 0.199
FullRollout-GRPO	3.2K	0.511 ± 0.051	0.494 ± 0.039	0.498 ± 0.089
ProEdit (GRPO, 3.2K)	3.2K	0.933 ± 0.038	0.852 ± 0.034	0.779 ± 0.077
ProEdit (RW-SFT)	6.4K	0.908 ± 0.039	0.839 ± 0.032	0.648 ± 0.077
ProEdit (GRPO, 6.4K)	6.4K	0.979 ± 0.025	0.883 ± 0.022	0.922 ± 0.055
IFNA2				
Original designs	–	0.211 ± 0.209	0.231 ± 0.186	0.646 ± 0.135
Frozen DPLM2	0	0.253 ± 0.179	0.242 ± 0.177	0.525 ± 0.203
FullRollout-GRPO	3.2K	0.299 ± 0.156	0.274 ± 0.161	0.499 ± 0.172
ProEdit (GRPO, 3.2K)	3.2K	0.872 ± 0.061	0.808 ± 0.059	0.542 ± 0.046
ProEdit (RW-SFT)	6.4K	0.851 ± 0.069	0.789 ± 0.066	0.536 ± 0.068
ProEdit (GRPO, 6.4K)	6.4K	0.943 ± 0.028	0.871 ± 0.024	0.594 ± 0.035

G.2 Parent-context controls

Table 20: **Complete parent-context control results.** The final target-specific ProEdit checkpoint is evaluated with the natural test parent, a length-matched random sequence, or an all-mask input.

	PD-L1			IFNA2		
Input	Reward	iPTM	pLDDT	Reward	iPTM	pLDDT
Natural parent	0.9787	0.8827	0.9225	0.9434	0.8707	0.5944
Random sequence	0.2789	0.2491	0.3853	0.4095	0.3754	0.4040
All-mask	0.4342	0.3957	0.4936	0.2809	0.2522	0.3855

G.3 Group-size sensitivity

We compare $G \in \{8, 64, 128\}$ on PD-L1 with 1,024 candidates per collection. The corresponding global parent batches are 128, 16, and 8, and each collection receives two optimization passes. Within each reported budget block, all runs share the candidate count and number of optimizer updates. At both training budgets, $G = 8$ achieves the highest reward.

Table 21: **PD-L1 group-size sensitivity under matched candidate budgets.**

G	Train candidates	Updates	Reward	iPTM	pLDDT
8	643,072	1,256	0.9272	0.8467	0.7494
64	643,072	1,256	0.9054	0.8297	0.7405
128	643,072	1,256	0.8839	0.8186	0.6504
8	1,048,576	2,048	0.9684	0.8749	0.8862
64	1,048,576	2,048	0.9628	0.8721	0.8727
128	1,048,576	2,048	0.9521	0.8641	0.8004