# LEARNING NOISE OPTIMIZATION FOR REWARD ALIGNMENT IN FEW-STEP GENERATIVE MODELS

**Anonymous authors**
Paper under double-blind review

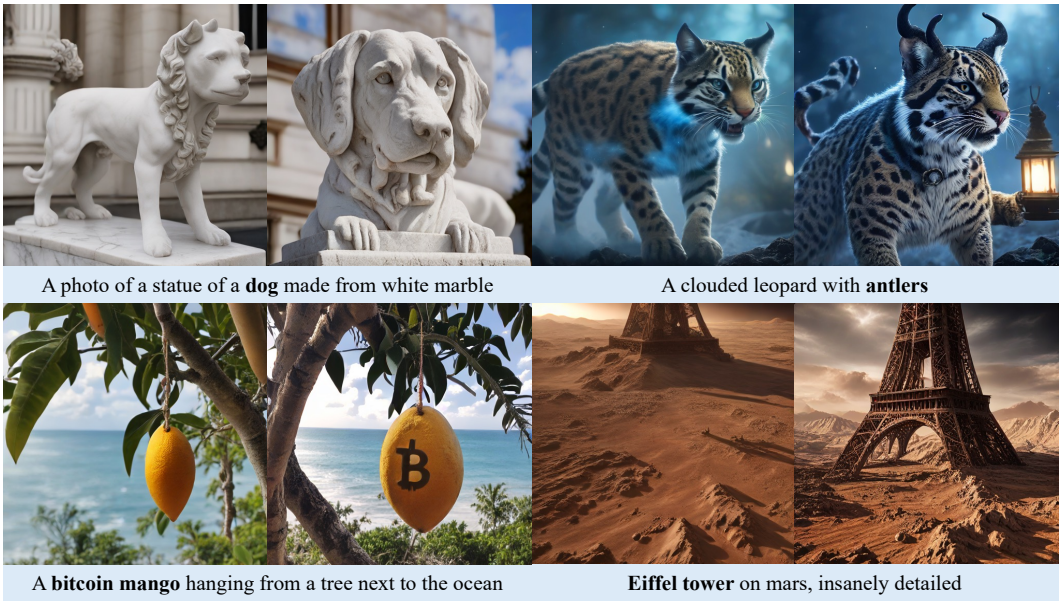



Figure 1: LeNO learns reward-improving corrections directly in initial noise space. Each pair shows generations before (left) and after (right) applying the learned noise correction, with the pretrained generator kept frozen. The corrected noises produce images that better capture the desired visual attributes across diverse prompts.

## ABSTRACT

Reward alignment for diffusion- and flow-based generative models is often formulated over stochastic trajectories, making it less suitable for few-step deterministic generators. Initial noise optimization provides a trajectory-free alternative, but existing gradient-free methods require costly per-sample search at inference. We introduce **LeNO** (**Le**arning **N**oise **O**ptimization), which amortizes reward-guided initial noise optimization into a learned residual transport while keeping the pretrained generator frozen. LeNO learns an anchor-conditioned distribution over noise corrections using Local GRPO and scalar rewards, eliminating iterative test-time optimization. For Gaussian initial noise with norm-preserving projection, we derive a tractable upper bound on the KL divergence between the corrected noise distribution and the original prior, providing a principled regularization objective. We evaluate LeNO on few-step text-to-image and protein structure generation, including expensive black-box designability rewards. Across both domains, LeNO improves target rewards with a single noise correction at inference while largely preserving generation quality and diversity.

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.

# 1 INTRODUCTION

Reward-based post-training has emerged as a powerful paradigm for adapting pretrained generative models toward objectives such as human preference, visual quality, and task-specific utility. In diffusion and flow models, many recent approaches formulate generation as a sequential decision process and optimize the generative trajectory using reinforcement learning or stochastic control (Black et al., 2024; Liu et al., 2026; Lee & Ye, 2025). Recent generative models, however, increasingly rely on few-step or even one-step deterministic sampling (Song et al., 2023; Yin et al., 2024; Sauer et al., 2024; Geng et al., 2025). While highly efficient, these models provide only few step deterministic generation trajectory, making conventional trajectory-level policy optimization less natural.

A promising alternative is to move reward optimization from the generative trajectory to the *initial noise space*, as the initial noise strongly influences the generated output. This has motivated recent inference time methods that search for reward-favorable noises while keeping the pretrained generator frozen (Eyring et al., 2024; Yoon et al., 2026; Kim et al., 2026). While these methods enable deterministic generators to be aligned with a given reward, including non-differentiable functions, they require an iterative noise-update process involving repeated generator executions and reward evaluations for each test sample. This becomes particularly costly when either generation or reward evaluation requires an expensive external pipeline.

This motivates *amortizing* reward-guided noise optimization into a learned mapping. Existing approaches, however, typically require either pre-collected offline data or differentiable reward functions. For example, GoldenNoise (Zhou et al., 2025) learns from a pre-constructed dataset of favorable noise–sample pairs, while HyperNoise (Eyring et al., 2026) learns to modify the initial noise through differentiable reward feedback. In contrast, directly learning sample-dependent noise corrections from online scalar rewards, without offline data or reward gradients, remains relatively underexplored.

To address these limitations, we introduce **LeNO** (**Le**arning **N**oise **O**ptimization), a framework that amortizes per-sample initial noise optimization for deterministic generators under differentiable or non-differentiable rewards. Rather than predicting an optimized noise sample from scratch, LeNO parameterizes the mapping in residual form and learns an anchor-dependent correction to the initial noise. To optimize this residual mapping using only scalar rewards, we formulate it as a stochastic transport that samples Gaussian residual actions around the predicted correction mean. These actions are trained with Local GRPO, where group-relative reward comparisons among corrections sampled from the same anchor provide the learning signal. We further provide a distributional interpretation of this residual parameterization. For Gaussian initial noise with anchor-norm-preserving projection, we show that a scaled squared norm of the predicted residual mean upper-bounds the KL divergence between the corrected noise distribution and the prior. This provides a tractable lower bound on the KL-regularized reward objective and directly motivates the residual regularization in Local GRPO. At inference, the learned transport generates a corrected initial noise with a single residual sample, eliminating per-sample optimization and additional reward evaluations.

We evaluate LeNO on few-step text-to-image generation and protein structure generation. In text-to-image generation, we consider direct optimization of multiple preference rewards. For proteins, we optimize a black-box designability objective whose evaluation requires sequence design and structure prediction, a setting where gradient-based amortization is inapplicable and per-sample search is costly. Across these settings, LeNO improves target rewards with only a lightweight residual correction at inference time, while retaining favorable held-out compositional performance and structural diversity. These results demonstrate that learned initial noise correction provides an efficient and broadly applicable alternative to repeated per-sample reward optimization.

# 2 BACKGROUND

**Reward-Tilted Noise Distribution.** For a deterministic generator $\boldsymbol{x} = G_\phi(\boldsymbol{z}, c)$ with initial noise $\boldsymbol{z} \sim p_0(\boldsymbol{z})$, the generated sample is fully determined by $z$ once the condition $c$ is fixed. Let $r(\cdot)$ denote a scalar, potentially black-box reward function defined on the generated sample. Reward-based alignment can therefore be formulated directly in the initial noise space by tilting the noise prior toward high-reward regions:

$$p^\star(\boldsymbol{z} \mid c) \propto p_0(\boldsymbol{z}) \exp\left(\frac{r(G_\phi(\boldsymbol{z}, c))}{\lambda}\right), \tag{1}$$

where $\lambda > 0$ controls the trade-off between reward maximization and preservation of the original noise prior. This formulation provides a direct basis for initial noise optimization, where the pretrained generator is kept fixed and reward alignment is performed by searching for higher-reward noise samples.

**Zeroth-Order Initial Noise Optimization.** Reward optimization can be performed directly in the initial noise space without differentiating through either the generator or the reward function. Let

$$\mathcal{R}(\boldsymbol{z}; c) := r(G_\phi(\boldsymbol{z}, c)), \qquad \mathcal{R}_\sigma(\boldsymbol{z}; c) := \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}\left[\mathcal{R}(\boldsymbol{z} + \sigma\epsilon; c)\right]. \tag{2}$$

The gradient $g(\boldsymbol{z}, c)$ of the Gaussian-smoothed reward can be estimated from centered reward-weighted perturbations as

$$g(\boldsymbol{z}, c) := \nabla_{\boldsymbol{z}} \mathcal{R}_\sigma(\boldsymbol{z}; c) \approx \frac{1}{(K-1)\sigma} \sum_{i=1}^{K} \left(R_i - \bar{R}_{\boldsymbol{z}}\right) \epsilon_i, \tag{3}$$

where $R_i = \mathcal{R}(\boldsymbol{z} + \sigma\epsilon_i; c)$ and $\bar{R}_{\boldsymbol{z}} = \frac{1}{K}\sum_{i=1}^{K} R_i$. This form underlies Natural Evolution Strategy style (Wierstra et al., 2014) gradient estimation and related gradient-free initial noise optimization methods, including approaches that derive reward-weighted perturbation updates from stochastic control perspectives (Kim et al., 2026). Such methods avoid reward backpropagation, but require estimating a new local direction for each noise state through repeated generator and reward evaluations. LeNO instead seeks to amortize these state-dependent noise corrections into a learned residual transport, avoiding repeated optimization at inference time.

**Policy Optimization for Deterministic Generative Models.** Recent approaches have explored how to enable policy optimization for deterministic generative models by recovering a suitable source of stochasticity. Flow-GRPO (Liu et al., 2026), for example, converts the deterministic probability-flow ODE into an equivalent stochastic differential equation, thereby introducing stochastic transitions on which policy-gradient optimization can be performed. An alternative perspective preserves the deterministic sampling dynamics and instead constructs stochasticity through local trajectory neighborhoods.

Neighbor GRPO (He et al., 2026) follows this local-neighborhood perspective. Given a shared reference noise $\epsilon^\star$, it samples neighboring initial conditions

$$\epsilon^{(i)} = \sqrt{1-\sigma^2}\epsilon^\star + \sigma\delta^{(i)}, \qquad \delta^{(i)} \sim \mathcal{N}(0, I), \tag{4}$$

and propagates them through the same deterministic ODE to construct a local trajectory neighborhood. Despite the similar shared-anchor perturbation geometry, Neighbor GRPO optimizes the generative model through a contrastive, distance-based surrogate policy defined over neighboring ODE trajectories. In contrast, LeNO keeps the generator frozen and learns a separate stochastic residual transport in the initial noise space, using local perturbations to obtain reward-based training signals. The two approaches are therefore complementary, as LeNO leaves the pretrained model unchanged and applies to any generator that exposes an initial noise input.

## 3 METHODS

### 3.1 NOISE-SPACE RESIDUAL TRANSPORT

We formulate amortized initial noise optimization as a stochastic residual transport for a frozen pretrained generator $G_\phi$. Given an initial noise $\boldsymbol{z} \sim p_0$ and condition $c$, we learn an anchor-dependent residual mean $\mu_\theta(\boldsymbol{z}, c)$ to improve the generated output using scalar reward feedback. The transport samples a residual action and applies it to the anchor:

$$\boldsymbol{a} = \eta\mu_\theta(\boldsymbol{z}, c) + \sigma\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{5}$$

$$\boldsymbol{z}' = F_\theta(\boldsymbol{z}, \epsilon, c) := \mathcal{T}(\boldsymbol{z}, \boldsymbol{a}), \tag{6}$$

where $\eta > 0$ controls the correction scale and $\sigma > 0$ is a fixed sampling scale. For Gaussian initial noise, we use the norm-preserving transition

$$\mathcal{T}(\boldsymbol{z}, \boldsymbol{a}) = \mathrm{NormProj}(\boldsymbol{z} + \boldsymbol{a}; \|\boldsymbol{z}\|_2), \qquad \mathrm{NormProj}(\boldsymbol{x}; r) := r\,\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}. \tag{7}$$

This updates the direction of the initial noise while preserving its norm. The stochastic transport induces the corrected noise distribution

$$\boldsymbol{z}' = F_\theta(\boldsymbol{z}, \epsilon, c) \sim q_\theta(\cdot \mid c), \tag{8}$$

where $\boldsymbol{z} \sim p_0$ and $\epsilon \sim \mathcal{N}(0, I)$ are independent.

At inference, we sample one corrected noise and generate $G_\phi(\boldsymbol{z}', c)$. During training, we draw multiple actions per anchor and compare their rewards to learn the residual mean. The same conditional Gaussian structure provides both a tractable distributional KL bound and an action likelihood for reward-based learning. We next derive the bound, followed by the Local GRPO objective.

### 3.2 Distributional Target and KL Regularization

We aim to align the corrected noise distribution $q_\theta(\cdot \mid c)$ with the reward-tilted target

$$p^\star(\boldsymbol{z} \mid c) = \frac{p_0(\boldsymbol{z}) \exp(\mathcal{R}(\boldsymbol{z}; c)/\lambda)}{Z(c)}, \qquad \lambda > 0, \tag{9}$$

where $p_0 = \mathcal{N}(0, I)$, $\mathcal{R}(\boldsymbol{z}; c) = r(G_\phi(\boldsymbol{z}, c))$ and $Z(c) := \mathbb{E}_{\boldsymbol{z}\sim p_0}[\exp(\mathcal{R}(\boldsymbol{z}; c)/\lambda)] < \infty$. This target favors high-reward noises while retaining the prior as a reference.

Whenever the relevant expectations and divergences are finite,

$$\begin{aligned} D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p^\star(\cdot \mid c)) = D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p_0) \\ - \frac{1}{\lambda} \mathbb{E}_{\boldsymbol{z}\sim q_\theta(\cdot|c)}[\mathcal{R}(\boldsymbol{z}; c)] + \log Z(c). \end{aligned} \tag{10}$$

Since $Z(c)$ is independent of $\theta$, minimizing this KL divergence is equivalent to maximizing

$$\mathcal{L}(q_\theta; c) := \mathbb{E}_{\boldsymbol{z}\sim q_\theta(\cdot|c)}[\mathcal{R}(\boldsymbol{z}; c)] - \lambda D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p_0)\,. \tag{11}$$

The reward term can be estimated from generated samples, but the prior-relative KL remains intractable because $q_\theta$ is implicitly defined by the transport. We therefore bound this KL term using the conditional Gaussian action distributions.

**Proposition 1.** *Let $p_0 = \mathcal{N}(0, I)$ and let $q_\theta$ be a corrected noise distribution induced by the norm-preserving transition $\mathcal{T}$ in Eq. (7). For $\sigma > 0$ and $\mathbb{E}_{\boldsymbol{z}\sim p_0}\|\mu_\theta(\boldsymbol{z}, c)\|_2^2 < \infty$, the following upper bound holds:*

$$\begin{aligned} D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p_0) &\le \mathbb{E}_{\boldsymbol{z}\sim p_0} D_{\mathrm{KL}}\big(\mathcal{N}(\eta\mu_\theta(\boldsymbol{z}, c), \sigma^2 I) \,\|\, \mathcal{N}(0, \sigma^2 I)\big) \\ &= \frac{\eta^2}{2\sigma^2} \mathbb{E}_{\boldsymbol{z}\sim p_0} \|\mu_\theta(\boldsymbol{z}, c)\|_2^2. \end{aligned} \tag{12}$$

The inequality follows from prior preservation under zero-mean residual actions, together with the data-processing inequality applied to the joint distributions of anchors and residual actions. The equality follows from the closed-form KL divergence between Gaussian distributions with identical covariance. A detailed proof is provided in Appendix B.

Under the conditions of Proposition 1, we obtain a tractable lower bound on the distributional objective:

$$\mathcal{L}_{\mathrm{aux}}(\theta; c) := \mathbb{E}_{\boldsymbol{z}'\sim q_\theta(\cdot|c)}[\mathcal{R}(\boldsymbol{z}'; c)] - \frac{\lambda\eta^2}{2\sigma^2} \mathbb{E}_{\boldsymbol{z}\sim p_0} \|\mu_\theta(\boldsymbol{z}, c)\|_2^2 \le \mathcal{L}(q_\theta; c). \tag{13}$$

The reward term can be estimated from sampled transport outputs, while the regularizer depends only on the predicted residual mean. Together with Eq. (10), this implies that maximizing $\mathcal{L}_{\mathrm{aux}}$ minimizes an upper bound on the KL divergence from $q_\theta$ to the reward-tilted target $p^\star$. Using the same stochastic transport and noise scale during training and inference ensures that the bound applies to the corrected noise distribution used for generation.

### 3.3 Local GRPO Objective

**Training Objective.** We next optimize the auxiliary reward objective $\mathcal{L}_{\mathrm{aux}}(\theta; c)$ through a tractable Local GRPO surrogate. Rather than maximizing the expected reward directly, Local GRPO uses group-relative comparisons among samples from the stochastic residual transport to update the residual mean from scalar rewards, while preserving the regularization implied by the distributional KL bound.

At the start of each training iteration, we snapshot the current transport parameters as $\theta_{\text{old}}$ and hold them fixed throughout the subsequent GRPO updates on the sampled particles. For each anchor $\boldsymbol{z}$ and condition $c$, we draw $K$ residual actions from the conditional Gaussian distribution in Eq. (5). For $i = 1, \ldots, K$,

$$\boldsymbol{a}^{(i)} = \eta \mu_{\theta_{\text{old}}}(\boldsymbol{z}, c) + \sigma \epsilon_i, \tag{14}$$

where $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$. Each action produces a corrected noise $\boldsymbol{z}^{(i)} = \mathcal{T}(\boldsymbol{z}, \boldsymbol{a}^{(i)})$ with reward $R_i = \mathcal{R}(\boldsymbol{z}^{(i)}; c)$. These particles are samples of the same stochastic transport used at inference, sharing the anchor and predicted residual mean while differing in their Gaussian perturbations. We compare their rewards within each group to construct the group-relative advantages:

$$A_i = \frac{R_i - \bar{R}}{s_R + \varepsilon}, \qquad \bar{R} = \frac{1}{K} \sum_{j=1}^{K} R_j, \tag{15}$$

**Algorithm 1** Training LeNO with Local GRPO

**Require:** Frozen generator $G_\phi$, reward $r$, residual transport $F_\theta$, group size $K$, residual scale $\eta$, exploration scale $\sigma$, regularization coefficient $\beta$

**for** each training iteration **do**
  $\theta_{\text{old}} \leftarrow \theta$
  Sample condition $c$ and anchor $\boldsymbol{z} \sim p_0(\boldsymbol{z})$
  **for** $i = 1, \ldots, K$ **do**
    Sample $\epsilon_i \sim \mathcal{N}(0, I)$
    $\boldsymbol{a}^{(i)} \leftarrow \eta \mu_{\theta_{\text{old}}}(\boldsymbol{z}, c) + \sigma \epsilon_i$
    $\boldsymbol{z}^{(i)} \leftarrow \mathcal{T}(\boldsymbol{z}, \boldsymbol{a}^{(i)})$
    $R_i \leftarrow r\big(G_\phi(\boldsymbol{z}^{(i)}, c)\big)$
  **end for**
  Compute group-relative advantages $A_i$
  Compute exploration likelihood ratios $\rho^{(i)}(\theta)$
  Update $\theta$ using Eq. (17)
**end for**

where $s_R$ is the within-group reward standard deviation and $\varepsilon > 0$ ensures numerical stability.

Let $\pi_\theta(\boldsymbol{a} \mid \boldsymbol{z}, c)$ denote the Gaussian action density specified by Eq. (5). For the sampled actions, we define the likelihood ratio

$$\rho^{(i)}(\theta) := \frac{\pi_\theta(\boldsymbol{a}^{(i)} \mid \boldsymbol{z}, c)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}^{(i)} \mid \boldsymbol{z}, c)}. \tag{16}$$

This ratio is evaluated on the residual actions before projection and does not require the density of the corrected noise marginal $q_\theta$. Holding the sampled actions and advantages fixed, our penalized local GRPO maximizes the following:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{K} \sum_{i=1}^{K} \min\Big(\rho^{(i)}(\theta) A_i,\ \text{clip}\Big(\rho^{(i)}(\theta), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}\Big) A_i\Big) - \beta \frac{\eta^2}{2\sigma^2} \|\mu_\theta(\boldsymbol{z}, c)\|_2^2\right], \tag{17}$$

where the expectation is over training conditions, anchors $\boldsymbol{z} \sim p_0$, and particles sampled under $\theta_{\text{old}}$. Here, $\epsilon_{\text{clip}} > 0$ is the PPO clipping parameter and $\beta > 0$ controls regularization strength. In Eq. (17), the first term promotes residual actions with higher group-relative rewards, while the second regularizes the policy toward the zero-mean Gaussian reference through the conditional KL divergence, thereby linking to the distributional KL term in Eq. (13). In practice, we combine group-normalized rewards with PPO-style clipping to obtain a stable Local GRPO surrogate, where $\beta$ controls the strength of KL regularization relative to the normalized reward scale.

The local sampling procedure also admits a zeroth-order interpretation. Before norm projection, reward-weighted Gaussian perturbations provide an estimator of the gradient of the Gaussian-smoothed reward evaluated at the mean-shifted anchor, closely related to zeroth-order noise optimization methods such as ZeNO (Kim et al., 2026). Unlike such methods, which optimize the initial noise independently for each test-time sample, LeNO uses these local reward signals to learn an anchor-conditioned transport and thereby amortizes noise optimization across samples. See Appendix A for more details of the derivation.

**Why State-Dependent Noise Corrections?** LeNO conditions the residual mean $\mu_\theta(\boldsymbol{z}, c)$ on the anchor $\boldsymbol{z}$ itself, rather than solely on the condition $c$. This design is motivated by the zeroth-order interpretation above: because local sampling estimates the reward-improvement direction in a neighborhood of a specific anchor, the resulting direction is generally anchor-dependent. In contrast, a correction conditioned only on $c$ cannot adapt to such local variation and instead captures only an average direction across anchors. Appendix C formalizes this intuition using a simplified first-order surrogate of $\mathcal{L}_{\text{aux}}$. Under this surrogate, the optimal anchor-dependent correction follows the local reward-improvement direction at each anchor, whereas the optimal anchor-independent correction follows their population average. The resulting objective gap is proportional to the total variance

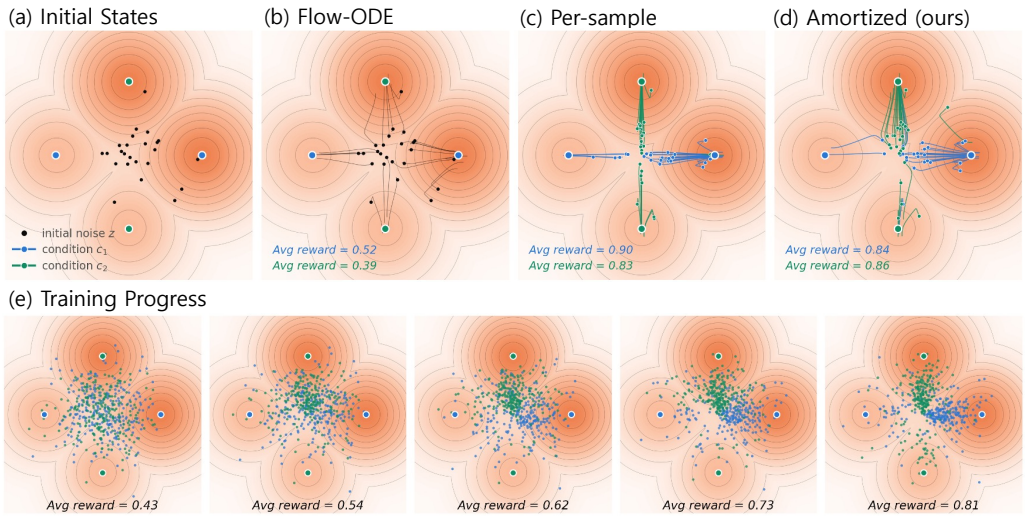



Figure 2: Toy Experiment. (a) Initial noise samples and the reward landscape of a four-component Gaussian mixture. (b) Samples obtained by directly passing the initial noise through the frozen probability-flow ODE. (c) Per-sample initial noise optimization with 30 step optimization. (d) Our amortized noise space correction with the residual transport. The learned correction successfully redirects samples toward the condition-specific high-reward modes with substantially lower inference time computation. (e) Corrected noise distribution over training.

of the local directions across anchors and vanishes only when these directions are identical for all anchors (Lemma 1). We empirically confirm the benefit of anchor-dependent conditioning in Fig. 4.

# 4 Experiments

## 4.1 Toy Experiment

We consider a four-component Gaussian mixture model where two different conditions, $c_1$ and $c_2$, select disjoint pairs of target modes with distinct reward values. The deterministic generator $G_\phi$ is defined as the probability-flow ODE of a linear interpolant between $\mathcal{N}(0, I)$ and the target mixture. Fig. 2(a) illustrates the initial noise samples and the corresponding reward landscape, where darker regions indicate higher rewards. As a baseline, Fig. 2(b) directly maps the initial particles through the frozen flow ODE without any reward-based correction. For per-sample optimization, Fig. 2(c) applies per-sample 30 step optimization, following ZeNO (Kim et al., 2026), with 32 perturbation particles per step, requiring 960 generator evaluations per sample. In contrast, LeNO in Fig. 2(d) uses one-step residual correction with a lightweight 35K-parameter MLP trained for 400 iterations.

Despite replacing iterative per-sample optimization with a learned corrector, our method consistently redirects samples toward the high-reward modes specified by each condition. While per-sample optimization improves rewards for each condition to $0.90$ and $0.83$, our amortized correction attains $0.84$ and $0.86$, showing that condition-dependent reward-improving corrections can be effectively amortized into a compact noise space transport with substantially reduced inference time computation. Figure 2(e) further visualizes the evolution of the learned correction during training. As training progresses, the transport gradually shifts the initial noise samples toward condition-dependent high-reward regions, increasing the average reward.

## 4.2 Text-to-Image Generation

**Experimental Setup.** We evaluate our method on text-to-image generation under two complementary reward-alignment settings: single reward optimization and compositional multi-reward optimization. Across all settings, the pretrained image generator $G_\phi$ remains frozen, and only the residual noise transport is optimized. We parameterize the residual correction $\mu_\theta$ using the same




Figure 3: Qualitative comparison of one-step generations before (left) and after LeNO noise correction (right). Top: SANA-Sprint with compositional multi-reward optimization. Bottom: SDXL-Turbo with PickScore optimization.

backbone architecture as the corresponding generator, with LoRA adapters injected into the model while keeping the underlying pretrained weights frozen. We use SDXL-Turbo (Sauer et al., 2024) for the single-reward and SANA-Sprint (Chen et al., 2025) for the compositional multi-reward setting. For single-reward optimization, we train separate residual transports for three reward functions: aesthetic score (Schuhmann, 2022), PickScore (Kirstain et al., 2023), and ImageReward (Xu et al., 2024). For compositional multi-reward optimization, we optimize a composite reward consisting of ImageReward, PickScore, HPSv2 (Wu et al., 2023), and CLIPScore (Hessel et al., 2021) proposed in ReNO (Eyring et al., 2024) and use GenEval (Ghosh et al., 2023) solely as a held-out benchmark to evaluate compositional generalization. Full experimental details are provided in Appendix D.

**Baselines.** We compare against both *inference-time* and *amortized* alignment methods. For inference time alignment, we include Best-of-$N$, $\psi$-Sampler (Yoon et al., 2026), and ZeNO (Kim et al., 2026). These methods require no additional model training, but improve reward through additional generator and reward evaluations at test time. In particular, ZeNO performs iterative gradient-free optimization independently for each initial noise, providing a direct per-sample counterpart to the optimization that our method aims to amortize. For amortized alignment, we compare with HyperNoise (Eyring et al., 2026), which learns to modify the initial noise distribution through differentiable reward feedback. We additionally report GoldenNoise (Zhou et al., 2025) as a reference baseline. Unlike the target reward alignment methods considered here, GoldenNoise is trained from an offline dataset of favorable noise-sample pairs rather than directly optimizing the evaluation reward. For the held-out GenEval comparison, we further include a broad set of previously reported text-to-image models and alignment methods as reference baselines (Rombach et al., 2022; Podell et al., 2023; Rafailov et al., 2023; Batifol et al., 2025; Esser et al., 2024).

**Comparison with Baselines.** Table 1 reports direct optimization results for aesthetic score, PickScore, and ImageReward. With a single residual correction, LeNO improves all three rewards over frozen SDXL-Turbo and achieves the best results among the evaluated amortized methods, outperforming HyperNoise without reward gradients. GoldenNoise, trained with different offline supervision, yields only marginal gains. We further evaluate cross-reward transfer, held-out quality metrics, and within-prompt diversity throughout training (Appendix E.1), which indicate that these gains are not obtained at the cost of perceptual quality or sample diversity, suggesting no evident reward hacking.

Inference-time methods achieve higher absolute rewards with substantially larger search budgets: Best-of-$N$, $\psi$-Sampler, and ZeNO use 800 generator evaluations per sample, compared with two NFEs for LeNO. LeNO amortizes this optimization into training, avoiding repeated reward evaluations at inference. Combining LeNO with Best-of-$N$ over 50 corrected candidates achieves competitive or better rewards than these 800-NFE baselines using only 100 NFEs, demonstrating more efficient inference-time scaling.

Table 2 evaluates transfer to GenEval, which is not directly optimized during multi-reward training. On SANA-Sprint, LeNO improves the mean score from 0.67 to 0.70, with gains in two-object generation, counting, position, and attribute binding. This is comparable to HyperNoise (0.71),

Table 1: Single reward optimization results on text-to-image generation with SDXL-Turbo.

| Method | $\nabla R$ | NFE ↓ | Aes. ↑ | Pick. ↑ | Img. ↑ |
|---|---|---|---|---|---|
| SDXL-Turbo | – | 1 | 5.339 | 21.932 | 0.761 |
| *Inference-Time* | | | | | |
| BoN | ✗ | 800 | 5.714 | 22.993 | 1.201 |
| $\psi$-Sampler | ✓ | 800 | 5.945 | 23.182 | 1.421 |
| ZeNO | ✗ | 800 | 5.750 | 23.038 | 1.260 |
| *Amortized* | | | | | |
| GoldenNoise[1] | – | 1 | 5.359 | 21.908 | 0.762 |
| HyperNoise | ✓ | 2 | 5.382 | 21.951 | 0.939 |
| **LeNO** | ✗ | 2 | 5.460 | 22.715 | 1.019 |
| **LeNO + BoN** | ✗ | 100 | **5.952** | **23.191** | **1.444** |

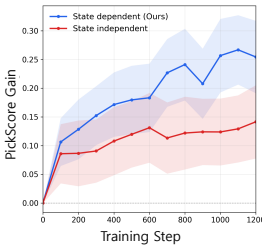



Figure 4: Ablation for state-dependent training.

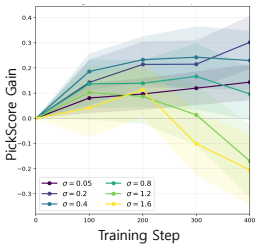



Figure 5: Ablation for exploration scale $\sigma$.

Table 2: GenEval results for held-out compositional evaluation. LeNO and HyperNoise are trained from SANA-Sprint with the same multi-reward objective; GenEval is not used for training. HyperNoise requires reward gradients, whereas LeNO uses only scalar rewards.

| Model | Params (B) | Time (s) ↓ | Mean ↑ | Single ↑ | Two ↑ | Counting ↑ | Colors ↑ | Position ↑ | Attribution ↑ |
|---|---|---|---|---|---|---|---|---|---|
| SD v2.1 | 0.8 | 1.9 | 0.50 | 0.98 | 0.51 | 0.44 | 0.85 | 0.07 | 0.17 |
| SDXL | 2.6 | 6.9 | 0.55 | 0.98 | 0.74 | 0.39 | 0.85 | 0.15 | 0.23 |
| DPO-SDXL | 2.6 | 6.9 | 0.59 | 0.99 | 0.84 | 0.49 | 0.87 | 0.13 | 0.24 |
| Hyper-SDXL | 2.6 | 0.3 | 0.56 | 1.00 | 0.76 | 0.43 | 0.87 | 0.10 | 0.21 |
| Flux-dev | 12.0 | 23.0 | 0.68 | 0.99 | 0.85 | 0.74 | 0.79 | 0.21 | 0.48 |
| SD3-Medium | 2.0 | 4.4 | 0.70 | 1.00 | 0.90 | 0.72 | 0.87 | 0.31 | 0.66 |
| SANA-Sprint | 0.6 | 0.2 | 0.67 | 1.00 | 0.68 | 0.60 | 0.85 | 0.40 | 0.47 |
| + HyperNoise | 0.9 | 0.3 | 0.71 | 1.00 | 0.75 | 0.67 | 0.84 | 0.48 | 0.53 |
| **+ LeNO** | 0.9 | 0.3 | 0.70 | 1.00 | 0.74 | 0.66 | 0.85 | 0.41 | 0.54 |

despite using only scalar reward evaluations rather than reward gradients, and suggests transfer beyond the training reward ensemble.

**State Dependency Ablation.** We further evaluate whether conditioning the residual correction on the current initial noise is necessary for effective amortization by comparing the standard state-dependent model $\mu_\theta(\boldsymbol{z}, c)$ with a state-independent variant $\mu_\theta(c)$. To preserve the same network architecture, the latter is implemented by feeding a fixed dummy noise $\boldsymbol{z}_{\text{fixed}}$ in place of the current noise input, while keeping the actual initial noise used for generation and the Local GRPO training procedure unchanged. As shown in Fig. 4, the state-dependent model continues to improve throughout training, whereas the state-independent variant shows only a modest initial gain before quickly saturating. This behavior aligns with the intuition from Sec. 3.3, where a shared correction aggregates reward-improvement signals across the initial noise population, while $\mu_\theta(\boldsymbol{z}, c)$ can represent distinct local corrections for different noise states.

**Exploration Scale Analysis.** Fig. 5 analyzes the effect of the exploration scale $\sigma$ using SDXL-Turbo with PickScore as the training reward. We observe a clear trade-off between exploration and locality. When $\sigma$ is too small (e.g., $\sigma \leq 0.1$), the sampled particles remain close to the predicted correction, yielding weak reward contrast and slower improvement. Increasing $\sigma$ broadens the explored neighborhood and strengthens the relative reward signal, with the best performance obtained around $\sigma = 0.2$. However, excessively large exploration degrades training; when $\sigma$ exceeds approximately 1.2, the reward gain collapses and can even become negative. At this scale, the stochastic perturbation dominates the predicted correction, weakening the local correspondence between sampled particles and the reward landscape around the shared anchor. These results suggest that effective Local GRPO requires sufficient exploration to reveal reward differences while preserving a coherent local neighborhood for relative credit assignment.

### 4.3 Protein Structure Generation

Protein backbone generation is the setting that most directly motivates LeNO. Evaluating a backbone requires designing compatible amino-acid sequences and predicting their folded structures through an external pipeline, so the reward is both expensive and non-differentiable with respect to the generator. Gradient-based amortization such as HyperNoise is therefore inapplicable, while per-

[1]GoldenNoise is trained on a pre-constructed noise–sample pair dataset and is not directly comparable to methods that optimize the target reward. We evaluate the SDXL-turbo model via zero-shot transfer from SDXL.

Table 3: Comparison of the base models and LeNO. Results are averaged over lengths 60–128, with 10 backbones per length and 8 sequences per backbone.

| Model | NFE | #Query | Designability | | Diversity | |
|---|---|---|---|---|---|---|
| | | | $< 2$Å (↑) | scRMSD (↓) | Max. Cluster (↑) | Pairwise scTM (↓) |
| FrameDiff | 5 | 0 | 0.09 | 6.19 | 0.54 | 0.24 |
| FrameFlow | 5 | 0 | 0.04 | 6.53 | 0.68 | 0.22 |
| RMF-M | 1 | 0 | 0.15 | 4.91 | 0.66 | 0.34 |
| + ZeNO | 480 | 480 | 0.28 | 3.54 | 0.65 | 0.35 |
| + **LeNO** | 2 | **0** | **0.31** | **3.27** | 0.64 | 0.38 |
| RMF-L | 1 | 0 | 0.29 | 3.38 | 0.66 | 0.35 |
| + ZeNO | 480 | 480 | 0.40 | **2.60** | 0.71 | 0.36 |
| + **LeNO** | 2 | **0** | **0.42** | 2.95 | 0.66 | 0.38 |

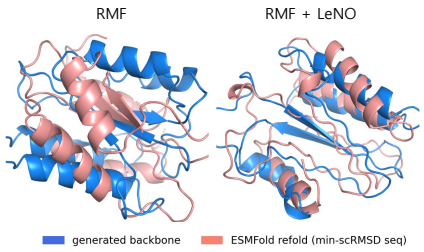



Figure 6: Visualization of protein structures generated by the base RMF and LeNO.

sample reward-guided search must invoke the full pipeline many times for every test sample. LeNO instead uses this feedback only during training and applies a single learned correction to new samples without invoking the evaluator at inference.

We optimize *designability* using the standard self-consistency pipeline. For each generated backbone, ProteinMPNN (Dauparas et al., 2022) designs compatible amino-acid sequences, which are then refolded using ESMFold (Lin et al., 2023). A backbone is considered designable when the self-consistency RMSD (scRMSD) between the generated and refolded structures is below 2 Å. Each reward query thus involves multiple sequence design and structure prediction calls, which dominates the cost of generation itself.

We apply LeNO to a pretrained one-step Riemannian MeanFlow (RMF) (Woo et al., 2026) model while keeping the generator frozen, parameterizing the residual transport with LoRA adapters trained using the same Local GRPO objective. As the primary baselines, we compare against per-sample test-time ZeNO (Kim et al., 2026) applied to the same frozen RMF. We additionally report FrameDiff (Yim et al., 2023b) and FrameFlow (Yim et al., 2023a), on which RMF is built, as few-step reference models. During training, protein lengths are uniformly sampled from 60 to 128 residues, and all methods are evaluated over the same length range.

**Results.** Table 3 reports designability and diversity for few-step protein structure generation. FrameDiff and FrameFlow degrade substantially in the few-step regime, whereas LeNO markedly improves the designability of both RMF variants with a single residual correction. Notably, on RMF-M, LeNO outperforms per-sample ZeNO in both designability and scRMSD, even though ZeNO performs 30 optimization steps with 16 particles, i.e., 480 black-box reward queries per sample, each invoking ProteinMPNN and ESMFold. LeNO instead requires no reward queries at inference, matching or exceeding explicit per-sample search at a small fraction of its test-time cost. Its one-time training cost (1k iterations $\times$ 36 particles $\approx$ 36k queries) is thus amortized after generating roughly 75 backbones, beyond which LeNO is strictly cheaper than per-sample optimization. These gains come at a modest cost in structural diversity: MaxCluster (Herbert & Sternberg, 2008) remains comparable to the base models, while pairwise scTM increases slightly, suggesting that LeNO shifts the backbone distribution toward more designable regions without collapsing it. Qualitative examples are shown in Fig. 6, and implementation details are provided in Appendix D.2.

## 5 Conclusion

We presented LeNO as an efficient framework for learning reward-improving corrections directly in the initial noise space of frozen deterministic generators. Across text-to-image and protein structure generation, our results show that a lightweight stochastic residual transport can effectively amortize costly per-sample reward optimization while supporting non-differentiable black-box objectives. LeNO also maintains favorable held-out behavior, including compositional generalization in image generation and structural diversity in protein generation. A remaining limitation is that training still requires multiple generator and reward evaluations per anchor, which can be expensive when the reward pipeline itself is costly. Nevertheless, the learned correction introduces only modest inference overhead, making noise space optimization particularly attractive when extremely fast few-step or one-step generation is essential.

### AI USE STATEMENT

In this work, we used generative AI tools for sentence-level editing of texts written by authors to improve readability. We have not used generative AI tools for other aspects, such as generating synthetic data, developing theoretical models, formulating mathematical claims or proposing research hypothesis. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, pp. arXiv–2506, 2025.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16185–16195. IEEE, 2025.

J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187. URL https://www.science.org/doi/abs/10.1126/science.add2187.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. ReNO: Enhancing one-step text-to-image models through reward-based noise optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=MXY0qsGgeO.

Luca Eyring, Shyamgopal Karthik, Alexey Dosovitskiy, Nataniel Ruiz, and Zeynep Akata. Noise hypernetworks: Amortizing test-time compute in diffusion models. *Advances in Neural Information Processing Systems*, 38:95429–95462, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=Wbr51vK331.

Dailan He, Guanlin Feng, Xingtong Ge, Yazhe Niu, Yi Zhang, Bingqi Ma, Guanglu Song, Yu Liu, and Hongsheng Li. Neighbor grpo: Contrastive ode policy optimization aligns flow models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6033–6042, 2026.

A. Herbert and M. Sternberg. Maxcluster: a tool for protein structure comparison and clustering. https://ic.ac.uk, 2008.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5128–5137. IEEE, 2021.

Jeongsol Kim, Hongeun Kim, Jian Wang, and Jong Chul Ye. Gradient-free noise optimization for reward alignment in generative models. *arXiv preprint arXiv:2605.11347*, 2026.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=G5RwHpBUv0.

Jeongjae Lee and Jong Chul Ye. Pcpo: Proportionate credit policy optimization for aligning image generation models. *arXiv preprint arXiv:2509.25774*, 2025.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574. URL https://www.science.org/doi/abs/10.1126/science.ade2574.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=oCBKGw5HNf.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024.

Christoph Schuhmann. Improved aesthetic predictor. https://github.com/christophschuhmann/improved-aesthetic-predictor, 2022. GitHub repository.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Ming Tao, Hao Tang, Fei Wu, Xiao-Yuan Jing, Bing-Kun Bao, and Changsheng Xu. Df-gan: A simple and effective baseline for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16515–16525, 2022.

Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 2555–2563, 2023.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014. URL http://jmlr.org/papers/v15/wierstra14a.html.

Dongyeop Woo, Marta Skreta, Seonghyun Park, Kirill Neklyudov, and Sungsoo Ahn. Riemannian meanflow, 2026. URL https://arxiv.org/abs/2602.07744.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023a.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023b.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024.

TaeHoon Yoon, Yunhong Min, Kyeongmin Yeo, and Minhyuk Sung. $\psi$-sampler: Initial particle sampling for SMC-based inference-time reward alignment in score models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=slVqJAI5sT.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zikai Zhou, Shitong Shao, Lichen Bai, Shufei Zhang, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17688–17697. IEEE, 2025.

## A Relation to Inference-Time Zeroth-Order Estimator

We analyze the gradient of the expected reward under the stochastic residual transport, omitting the residual-magnitude regularizer and considering the additive transition before norm projection. Fix an anchor $\boldsymbol{z}$ and condition $c$, and define the mean-shifted anchor

$$\tilde{\boldsymbol{z}} := \boldsymbol{z} + \eta\mu_\theta(\boldsymbol{z}, c). \tag{18}$$

The expected reward under Gaussian action sampling is

$$J_R(\boldsymbol{z}; c) := \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\mathcal{R}(\tilde{\boldsymbol{z}} + \sigma\epsilon; c)\right] = \mathcal{R}_\sigma(\tilde{\boldsymbol{z}}; c). \tag{19}$$

We assume the integrability conditions required for differentiating the Gaussian smoothing hold. For brevity, $\nabla_{\mu_\theta}$ denotes differentiation with respect to the predicted residual mean $\mu_\theta(\boldsymbol{z}, c)$, rather than the network parameters.

By the chain rule and the Gaussian score identity,

$$\begin{aligned}\nabla_{\mu_\theta} J_R(\boldsymbol{z}; c) &= \eta\, \nabla_{\tilde{\boldsymbol{z}}} \mathcal{R}_\sigma(\tilde{\boldsymbol{z}}; c) \\ &= \frac{\eta}{\sigma}\mathbb{E}_\epsilon\left[\mathcal{R}(\tilde{\boldsymbol{z}} + \sigma\epsilon; c)\epsilon\right].\end{aligned} \tag{20}$$

Thus, the reward gradient with respect to the residual mean equals $\eta$ times the zeroth-order gradient of the smoothed reward at the mean-shifted anchor.

Since $\mathbb{E}_\epsilon[\epsilon] = 0$, a baseline independent of the sampled perturbation does not change the expected gradient. For $K > 1$ independent perturbations, define

$$R_i = \mathcal{R}(\tilde{\boldsymbol{z}} + \sigma\epsilon_i; c), \qquad \bar{R} = \frac{1}{K}\sum_{i=1}^{K} R_i. \tag{21}$$

An unbiased centered estimator is

$$\widehat{\nabla_{\mu_\theta} J_R} = \frac{\eta}{(K-1)\sigma}\sum_{i=1}^{K}(R_i - \bar{R})\epsilon_i. \tag{22}$$

The factor $1/(K-1)$ corrects for the dependence introduced by including $R_i$ in $\bar{R}$. Equivalently, this estimator uses the leave-one-out baseline $b_{-i} = (K-1)^{-1}\sum_{j\neq i} R_j$, since

$$R_i - b_{-i} = \frac{K}{K-1}(R_i - \bar{R}). \tag{23}$$

Equation 22 has the same form as Eq. (3), evaluated at $\tilde{\boldsymbol{z}}$ and scaled by $\eta$.

Local GRPO instead uses standardized group-relative advantages,

$$A_i = \frac{R_i - \bar{R}}{s_R + \varepsilon}. \tag{24}$$

At the sampling policy $\theta = \theta_{\text{old}}$, the unclipped reward-gradient estimate with respect to the residual mean is

$$\frac{\eta}{K\sigma}\sum_{i=1}^{K} A_i\epsilon_i = \frac{K-1}{K(s_R+\varepsilon)}\widehat{\nabla_{\mu_\theta} J_R}. \tag{25}$$

For a fixed sampled group, the multiplier is positive, so normalization preserves the direction of this local estimate. Across groups, however, normalization changes their relative weights and does not generally preserve the population reward gradient. Subsequent policy updates introduce likelihood-ratio weighting and potentially PPO clipping, so the resulting GRPO updates need not remain proportional to the zeroth-order estimate.

## B Proof of Proposition 1

**Proposition 1.** *Let $p_0 = \mathcal{N}(0, I)$ and let $q_\theta$ be a corrected noise distribution induced by the norm-preserving transition $\mathcal{T}$ in Eq. (7). For $\sigma > 0$ and $\mathbb{E}_{\boldsymbol{z}\sim p_0}\|\mu_\theta(\boldsymbol{z}, c)\|_2^2 < \infty$, the following upper bound holds:*

$$\begin{aligned}D_{\mathrm{KL}}(q_\theta(\cdot \mid c)\|p_0) &\le \mathbb{E}_{\boldsymbol{z}\sim p_0} D_{\mathrm{KL}}\big(\mathcal{N}(\eta\mu_\theta(\boldsymbol{z}, c), \sigma^2 I)\,\|\,\mathcal{N}(0, \sigma^2 I)\big) \\ &= \frac{\eta^2}{2\sigma^2}\mathbb{E}_{\boldsymbol{z}\sim p_0}\|\mu_\theta(\boldsymbol{z}, c)\|_2^2.\end{aligned} \tag{12}$$

*Proof.* Fix a condition $c$ and $\sigma > 0$. Let $p_0 = \mathcal{N}(0, I_d)$ and

$$\mathcal{T}(\boldsymbol{z}, \boldsymbol{a}) = \|\boldsymbol{z}\|_2 \frac{\boldsymbol{z} + \boldsymbol{a}}{\|\boldsymbol{z} + \boldsymbol{a}\|_2}.$$

We first show that zero-mean Gaussian actions preserve the prior, then apply data processing to establish the KL bound.

**Preservation of the Gaussian prior.** Let $\boldsymbol{z}, \epsilon \sim \mathcal{N}(0, I_d)$ be independent. The output under zero-mean residual actions is

$$\boldsymbol{y} = \mathcal{T}(\boldsymbol{z}, \sigma\epsilon) = \|\boldsymbol{z}\|_2 \frac{\boldsymbol{z} + \sigma\epsilon}{\|\boldsymbol{z} + \sigma\epsilon\|_2}. \tag{26}$$

The denominator is nonzero almost surely, and the transition may be defined arbitrarily on the null set where it vanishes.

Write $\boldsymbol{z} = r\boldsymbol{u}$, where $r = \|\boldsymbol{z}\|_2$. For an isotropic Gaussian, $\boldsymbol{u}$ is uniform on the unit sphere and independent of $r$. Conditional on $r > 0$, the vector $r\boldsymbol{u} + \sigma\epsilon$ is rotationally invariant: for every orthogonal matrix $Q$, the pairs $(Q\boldsymbol{u}, Q\epsilon)$ and $(\boldsymbol{u}, \epsilon)$ have the same conditional distribution. Its normalized direction is therefore uniform on the unit sphere conditional on $r$. Since this conditional distribution does not depend on $r$, the output direction is independent of the radius.

The transition also preserves the radius exactly, $\|\boldsymbol{y}\|_2 = r$. Thus, $\boldsymbol{y}$ has the same radial distribution and independent uniform direction as $\boldsymbol{z}$, giving

$$\mathcal{T}(\boldsymbol{z}, \sigma\epsilon) \sim \mathcal{N}(0, I_d) = p_0. \tag{27}$$

**Bounding the marginal KL divergence.** Define the joint distributions of anchors and residual actions under the learned and zero-mean transports by

$$\begin{aligned} P_\theta(d\boldsymbol{z}, d\boldsymbol{a} \mid c) &= p_0(d\boldsymbol{z})\, \mathcal{N}\big(d\boldsymbol{a}; \eta\mu_\theta(\boldsymbol{z}, c), \sigma^2 I\big)\,, \\ P_{\mathrm{ref}}(d\boldsymbol{z}, d\boldsymbol{a}) &= p_0(d\boldsymbol{z})\, \mathcal{N}\big(d\boldsymbol{a}; 0, \sigma^2 I\big)\,. \end{aligned} \tag{28}$$

Both joint distributions share the anchor marginal $p_0$. The chain rule for KL divergence therefore gives

$$\begin{aligned} D_{\mathrm{KL}}(P_\theta(\cdot \mid c) \| P_{\mathrm{ref}}) &= \mathbb{E}_{\boldsymbol{z}\sim p_0} D_{\mathrm{KL}}\big(\mathcal{N}(\eta\mu_\theta(\boldsymbol{z}, c), \sigma^2 I) \,\|\, \mathcal{N}(0, \sigma^2 I)\big) \\ &= \frac{\eta^2}{2\sigma^2} \mathbb{E}_{\boldsymbol{z}\sim p_0} \|\mu_\theta(\boldsymbol{z}, c)\|_2^2. \end{aligned} \tag{29}$$

The second equality follows from the Gaussian KL formula for identical covariances. The assumed finite second moment makes this divergence finite.

Applying the same map $(\boldsymbol{z}, \boldsymbol{a}) \mapsto \mathcal{T}(\boldsymbol{z}, \boldsymbol{a})$ to $P_\theta(\cdot \mid c)$ produces $q_\theta(\cdot \mid c)$. By Eq. (27), applying it to $P_{\mathrm{ref}}$ produces $p_0$. The data-processing inequality consequently yields

$$\begin{aligned} D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p_0) &\le D_{\mathrm{KL}}(P_\theta(\cdot \mid c) \| P_{\mathrm{ref}}) \\ &= \frac{\eta^2}{2\sigma^2} \mathbb{E}_{\boldsymbol{z}\sim p_0} \|\mu_\theta(\boldsymbol{z}, c)\|_2^2. \end{aligned} \tag{30}$$

This proves the proposition without requiring an invertible transition or evaluating its Jacobian. □

**From the reward-tilted target to the auxiliary objective.** Let $\lambda > 0$ and assume $0 < Z(c) < \infty$ and $\mathbb{E}_{\boldsymbol{z}'\sim q_\theta(\cdot|c)}[|\mathcal{R}(\boldsymbol{z}'; c)|] < \infty$. From the definition of $p^\star$,

$$\log \frac{q_\theta(\boldsymbol{z}' \mid c)}{p^\star(\boldsymbol{z}' \mid c)} = \log \frac{q_\theta(\boldsymbol{z}' \mid c)}{p_0(\boldsymbol{z}')} - \frac{\mathcal{R}(\boldsymbol{z}'; c)}{\lambda} + \log Z(c). \tag{31}$$

Taking expectations under $q_\theta(\cdot \mid c)$ gives

$$\begin{aligned} D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p^\star(\cdot \mid c)) &= D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p_0) \\ &\quad - \frac{1}{\lambda} \mathbb{E}_{\boldsymbol{z}'\sim q_\theta(\cdot|c)}[\mathcal{R}(\boldsymbol{z}'; c)] + \log Z(c). \end{aligned} \tag{32}$$

Since $Z(c)$ is independent of $\theta$, minimizing this divergence is equivalent to maximizing $\mathcal{L}(q_\theta; c)$.

Applying Proposition 1 to the prior-relative KL term yields

$$\begin{aligned}\mathcal{L}(q_\theta; c) &= \mathbb{E}_{\boldsymbol{z}' \sim q_\theta(\cdot|c)}[\mathcal{R}(\boldsymbol{z}'; c)] - \lambda D_{\mathrm{KL}}(q_\theta(\cdot \mid c) \| p_0) \\ &\geq \mathbb{E}_{\boldsymbol{z}' \sim q_\theta(\cdot|c)}[\mathcal{R}(\boldsymbol{z}'; c)] - \frac{\lambda \eta^2}{2\sigma^2} \mathbb{E}_{\boldsymbol{z} \sim p_0} \|\mu_\theta(\boldsymbol{z}, c)\|_2^2 \\ &= \mathcal{L}_{\mathrm{aux}}(\theta; c).\end{aligned} \tag{33}$$

This establishes Eq. (13). The same stochastic transport and sampling scale are used at inference, so the bound applies to the corrected noise distribution used for generation.

## C State-Dependent versus Shared Corrections

**First-order surrogate.** Fix a condition $c$, consider the additive transition before projection, and set $\eta = 1$ for notational simplicity. We write $\mu$ for an unrestricted correction function, independently of its neural parameterization. Under this transition, the corrected noise is $\boldsymbol{z}' = \boldsymbol{z} + \mu(\boldsymbol{z}, c) + \sigma\epsilon$, and the expected reward over the Gaussian action equals the smoothed reward at the mean-shifted anchor (Eq. (19)). The auxiliary objective in Eq. (13) then reads

$$\mathcal{L}_{\mathrm{aux}}(\mu; c) = \mathbb{E}_{\boldsymbol{z} \sim p_0}\left[\mathcal{R}_\sigma\big(\boldsymbol{z} + \mu(\boldsymbol{z}, c); c\big) - \frac{\lambda}{2\sigma^2}\|\mu(\boldsymbol{z}, c)\|_2^2\right]. \tag{34}$$

Linearizing $\mathcal{R}_\sigma$ around the anchor, $\mathcal{R}_\sigma(\boldsymbol{z} + \mu; c) \approx \mathcal{R}_\sigma(\boldsymbol{z}; c) + g(\boldsymbol{z}, c)^\top \mu(\boldsymbol{z}, c)$ with $g(\boldsymbol{z}, c) = \nabla_{\boldsymbol{z}} \mathcal{R}_\sigma(\boldsymbol{z}; c)$ as in Eq. (3), and dropping the $\mu$-independent term gives

$$J_{\mathrm{lin}}(\mu; c) := \mathbb{E}_{\boldsymbol{z} \sim p_0}\left[g(\boldsymbol{z}, c)^\top \mu(\boldsymbol{z}, c) - \frac{\alpha}{2}\|\mu(\boldsymbol{z}, c)\|_2^2\right], \qquad \alpha = \frac{\lambda}{\sigma^2}. \tag{35}$$

The following statements are exact for this surrogate, rather than for the original nonlinear reward objective.

**Lemma 1.** *Assume $\mathbb{E}_{p_0}\|g(\boldsymbol{z}, c)\|_2^2 < \infty$. Let $J^\star_{\mathrm{state}}$ and $J^\star_{\mathrm{shared}}$ denote the optimal values of Eq. (35) over square-integrable state-dependent corrections $\mu(\boldsymbol{z}, c)$ and shared corrections $\mu(c)$, respectively. The optimal corrections are $\mu^\star_{\mathrm{state}}(\boldsymbol{z}, c) = g(\boldsymbol{z}, c)/\alpha$ and $\mu^\star_{\mathrm{shared}}(c) = \mathbb{E}_{p_0}[g(\boldsymbol{z}, c)]/\alpha$, and*

$$\Delta J := J^\star_{\mathrm{state}} - J^\star_{\mathrm{shared}} = \frac{1}{2\alpha}\,\mathrm{Tr}(\mathrm{Cov}_{\boldsymbol{z} \sim p_0}[g(\boldsymbol{z}, c)]) \geq 0, \tag{36}$$

*with equality if and only if $g(\boldsymbol{z}, c)$ is constant $p_0$-almost surely.*

*Proof.* For each $\boldsymbol{z}$, completing the square gives

$$g^\top \mu - \frac{\alpha}{2}\|\mu\|_2^2 = \frac{1}{2\alpha}\|g\|_2^2 - \frac{\alpha}{2}\left\|\mu - \frac{g}{\alpha}\right\|_2^2, \tag{37}$$

where we omit the arguments $(\boldsymbol{z}, c)$. The pointwise maximizer is $\mu^\star_{\mathrm{state}} = g/\alpha$, which is square-integrable by the finite second-moment assumption, and $J^\star_{\mathrm{state}} = \frac{1}{2\alpha}\mathbb{E}_{p_0}\|g\|_2^2$. For a shared correction $\mu(c)$, the objective reduces to $\mathbb{E}_{p_0}[g]^\top \mu(c) - \frac{\alpha}{2}\|\mu(c)\|_2^2$, and the same argument gives $\mu^\star_{\mathrm{shared}} = \mathbb{E}_{p_0}[g]/\alpha$ and $J^\star_{\mathrm{shared}} = \frac{1}{2\alpha}\|\mathbb{E}_{p_0}[g]\|_2^2$. Subtracting,

$$\Delta J = \frac{1}{2\alpha}\left(\mathbb{E}_{p_0}\|g\|_2^2 - \|\mathbb{E}_{p_0}[g]\|_2^2\right) = \frac{1}{2\alpha}\,\mathrm{Tr}(\mathrm{Cov}_{p_0}[g]) \geq 0, \tag{38}$$

with equality if and only if $\mathrm{Cov}_{p_0}[g] = 0$, i.e., $g$ is constant $p_0$-almost surely. □

# D Implementation Details

All experiments are conducted using NVIDIA RTX 4090 GPUs with 24GB of memory.

## D.1 Text-to-Image Generation

**Hyperparameters.** We use the same hyperparameters for all text-to-image Local GRPO experiments. We set the correction scale to $\eta = 0.3$ and the exploration standard deviation to $\sigma = 0.2$, using 32 particles per group, 16 prompt anchors per update, and four GRPO epochs per rollout. Advantages are normalized using the group mean and standard deviation and clipped to $[-5, 5]$. The PPO probability ratio is clipped to $[0.8, 1.2]$, while the log-ratio is clipped to $[-10, 10]$ for numerical stability.

Table 4: GRPO training hyperparameters.

| **Hyperparameter** | **Value** |
|---|---|
| Correction scale $\eta$ | 0.3 |
| Exploration std. $\sigma$ | 0.2 |
| KL regularization weight $\beta$ | 0.01 |
| Global particle group size | 32 |
| Prompt anchors per update | 16 |
| GRPO epochs per rollout | 4 |
| Advantage normalization | Group mean/std |
| Advantage clipping | $[-5, 5]$ |
| PPO ratio clipping | $[0.8, 1.2]$ ($\epsilon = 0.2$) |
| Log-ratio clipping | $[-10, 10]$ |

**Single Reward Optimization.** For single-reward optimization, we use SDXL-Turbo as the frozen generator and consider three reward functions: the aesthetic score predictor, PickScore, and ImageReward. We train a separate LeNO model for each reward using only the corresponding scalar reward signal. For a fair comparison, we also retrain HyperNoise independently for all three reward functions rather than reusing a single pretrained checkpoint. In each case, we follow the default HyperNoise training configuration and change only the target reward function. The Local GRPO hyperparameters and model-specific training configurations are summarized in Tab. 4 and Tab. 5, respectively.

**Compositional Multi-Reward Optimization.** For compositional optimization, we follow the multi-reward formulation of ReNO (Eyring et al., 2024) and use SANA-Sprint as the frozen generator. Specifically, the training reward combines ImageReward, HPSv2, PickScore, and CLIPScore, which provide complementary signals for human preference, perceptual quality, and text–image consistency. Following ReNO, we use the weighted objective

$$R_{\text{multi}} = 1.0\, R_{\text{ImageReward}} + 5.0\, R_{\text{HPSv2}} + 0.05\, R_{\text{PickScore}} + 1.0\, R_{\text{CLIPScore}}, \tag{39}$$

where the weighting accounts for the substantially different numerical scales of the individual reward models. The corresponding Local GRPO and training configurations are reported in Tab. 4 and Tab. 5.

To evaluate whether improvements transfer beyond the reward models used for training, we additionally report GenEval (Ghosh et al., 2023) as a held-out evaluation. The GenEval prompts and evaluator are not used during LeNO training, and the GenEval score is not included in the multi-reward objective. We evaluate the trained model only after optimization using the standard GenEval benchmark, which measures compositional text–image alignment across single-object generation, two-object composition, counting, color, spatial relations, and attribute binding. Thus, improvements on GenEval indicate that the learned noise correction generalizes beyond direct optimization of the four training rewards.

Table 5: Training configurations for HyperNoise and LeNO under single- and multi-reward optimization.

| | Single Reward | | Multi Reward | |
|---|---|---|---|---|
| | **HyperNoise** | **LeNO** | **HyperNoise** | **LeNO** |
| Model | SDXL-Turbo | SDXL-Turbo | SANA-Sprint | SANA-Sprint |
| Denoising Step | 1 | 1 | 1 | 1 |
| Learning rate | $1 \times 10^{-3}$ | $5 \times 10^{-5}$ | $1 \times 10^{-3}$ | $5 \times 10^{-5}$ |
| GradNorm Clipping | 1.0 | 1.0 | 1.0 | 1.0 |
| LoRA rank | 16 | 16 | 128 | 128 |
| LoRA alpha | 16 | 16 | 256 | 256 |
| Optimizer | SGD | Adam | SGD | Adam |
| Batch size | 16 | 16 | 16 | 16 |
| Accumulation steps | 3 | 3 | 3 | 3 |
| Training steps | 60k | 5k | 60k | 5k |
| Number of training prompts | 55k | 55k | 55k | 55k |
| Image size | $512 \times 512$ | $512 \times 512$ | $1024 \times 1024$ | $1024 \times 1024$ |

### D.2 Protein Structure Generation

**Generator and Residual Correction.** We use the pretrained both RMF-M and RMF-L checkpoint at epoch 1600 as a frozen one-step backbone generator. For a protein of length $N$, the initial state consists of residue-wise translations and rotations, $X = (T, R) \in \mathbb{R}_0^{N\times 3} \times \mathrm{SO}(3)^N$, where the translation subspace has zero center of mass. Translations follow a centered Gaussian prior with standard deviation $\tau_T = 10\,\text{Å}$, and rotations are sampled independently from the Haar measure on $\mathrm{SO}(3)$. We parameterize the residual correction using the difference between LoRA-adapted and frozen RMF velocity fields:

$$(U_T, U_R) = f_\theta^{\mathrm{LoRA}}(X; 0, 1) - f_0(X; 0, 1). \tag{40}$$

Rank-32 LoRA adapters with $\alpha = 32$ are inserted into 284 trunk linear layers, giving approximately 17.2M trainable parameters. Only the adapters are optimized, and their output matrices are initialized to zero so that the initial predicted correction is zero.

**Stochastic Residual Transport.** Let $\mathcal{C}$ remove the translation center of mass and $\mathrm{clip}_c$ limit each residue-wise vector to norm $c$. We define the translation and rotation action means as

$$\begin{aligned} M_T &= \mathcal{C}\big(\mathrm{clip}_{c_T}(\eta_T U_T)\big)\,, \\ M_R &= \mathrm{clip}_{c_R}(\eta_R U_R), \end{aligned} \tag{41}$$

with $(\eta_T, \eta_R) = (0.30, 0.15)$, $c_T = 2.0\,\text{Å}$, and $c_R = 0.10\,\text{rad}$. The stochastic transport samples residual actions

$$\begin{aligned} A_T &= M_T + \sigma_T \mathcal{C}(\epsilon_T), \\ A_R &= M_R + \sigma_R \epsilon_R, \end{aligned} \tag{42}$$

where $\epsilon_T$ and $\epsilon_R$ are independent standard Gaussian tensors, $\sigma_T = 2.0\,\text{Å}$, and $\sigma_R = 0.20\,\text{rad}$. The corrected initial frames are

$$\begin{aligned} T^+ &= \|T\|_F \frac{\mathcal{C}(T + A_T)}{\|\mathcal{C}(T + A_T)\|_F}, \\ R_i^+ &= R_i \,\mathrm{Exp}(\widehat{A}_{R,i}), \end{aligned} \tag{43}$$

where $\widehat{\cdot}$ maps a tangent vector to a skew-symmetric matrix. The translation update preserves the Frobenius norm of each centered Gaussian anchor while changing its direction, and the rotation update remains on $\mathrm{SO}(3)$.

During training, we sample multiple independent action pairs for each anchor to construct group-relative advantages. At inference, we sample one action pair using the same $\sigma_T$ and $\sigma_R$ and apply a single residual transport step before generation.

Table 6: Training configuration for protein generation.

| **Hyperparameter** | **Value** |
|---|---|
| Frozen generator | RMF-M / RMF-L |
| Generator sampling steps | 1 |
| Residual transport steps | 1 |
| Training protein length | $N \sim \mathcal{U}\{60, \dots, 128\}$ |
| LoRA rank / alpha | $32/32$ |
| Trainable parameters | 7.1M / 17.2M |
| Initial states per iteration | 1 |
| Local particles per state | 36 |
| Optimization epochs per group | 2 |
| Gradient accumulation | 1 |
| Surrogate clipping | 0.20 |
| Advantage clipping | $[-5, 5]$ |
| KL coefficient | $5 \times 10^{-3}$ |
| Optimizer | Adam |
| Learning rate | $2 \times 10^{-5}$ |
| Adam betas | $(0.9, 0.999)$ |
| Weight decay | 0 |
| GradNorm clipping | 1.0 |
| Training steps | 1k |
| Sequences per backbone (train / eval.) | $4/8$ |
| Sequence sampling temperature | 0.1 |
| Reward soft-min temperature | 1.0 Å |

Proposition 1 extends directly to the protein structure generation setting. The translation prior is an isotropic Gaussian on the zero-center-of-mass subspace, and the centered noise $\mathcal{C}(\epsilon_T)$ lies in the same subspace, so the rotational-invariance argument applies within it. For rotations, $R_i$ is Haar-distributed and independent of the zero-mean action, so $R_i \operatorname{Exp}(\widehat{\sigma_R \epsilon_{R,i}})$ remains Haar-distributed by invariance of the Haar measure. Clipping acts only on the predicted mean and does not affect the argument.

**Local GRPO Training.** Each training iteration samples one initial state with $N \sim \mathcal{U}\{60, \dots, 128\}$ and explores 36 local particles. Rewards are standardized within each group using the population standard deviation with $\epsilon = 10^{-6}$, and advantages are clipped to $[-5, 5]$. Exploration likelihoods are evaluated on the stored pre-projection residuals $(A_T, A_R)$. We perform two optimization epochs per group with surrogate clipping $0.20$ and a Gaussian mean-shift KL regularizer toward zero residual correction, weighted by $5 \times 10^{-3}$. We use Adam with learning rate $2 \times 10^{-5}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, no weight decay, and gradient-norm clipping at $1.0$. We report the checkpoint at iteration 1k. The corresponding training configurations are reported in Tab. 6.

**Designability Reward and Evaluation.** Each corrected initial state is decoded into a backbone by the frozen one-step RMF generator. ProteinMPNN samples $K = 4$ sequences per backbone at temperature $0.1$, and ESMFold (`facebook/esmfold_v1`) refolds them. Given their aligned self-consistency RMSDs $\{d_k\}_{k=1}^{K}$, we maximize the negative normalized soft minimum:

$$\mathcal{R} = \tau \log \left( \frac{1}{K} \sum_{k=1}^{K} \exp(-d_k/\tau) \right), \qquad \tau = 1.0\,\text{Å}. \tag{44}$$

All particles within a local group share the same ProteinMPNN seed to reduce reward variance; no pLDDT term is included. For final evaluation, we sample eight sequences per backbone and classify it as designable when the minimum scRMSD is below 2 Å.

# E Additional Experiments

## E.1 Quality and Diversity Preservation

We investigate whether optimizing the initial noise for a single reward leads to diversity collapse or degradation in perceptual image quality. We evaluate the PickScore-, Aesthetic-, and ImageReward-optimized models on 100 held-out TempFlow prompts, using eight fixed initial-noise seeds per prompt. The same prompt–seed pairs are shared across all methods and checkpoints. Since the Aesthetic model has the shortest training horizon, we compare all models up to 800 training steps. The shaded regions in Fig. 7 denote 95% confidence intervals obtained by paired bootstrap resampling over prompts.

We report two complementary diversity metrics and two no-reference image quality metrics:

- **LPIPS diversity** (Zhang et al., 2018): the average pairwise LPIPS distance among the eight images generated for the same prompt. A higher value indicates greater perceptual variation across initial-noise samples.
- **DINO diversity** (Oquab et al., 2023): the average pairwise cosine distance between DINOv2 image features. This metric captures diversity at a more semantic and content-level representation than pixel-level distances.
- **MUSIQ** (Ke et al., 2021): a no-reference image quality assessment metric that measures overall perceptual quality without requiring reference images. Higher values indicate better image quality.
- **CLIP-IQA** (Wang et al., 2023): a CLIP-based no-reference image quality metric that assesses perceptual quality using semantic visual representations. Higher values indicate better quality.

For the diversity metrics, we additionally report retention relative to the unoptimized SDXL-Turbo baseline. For the quality metrics, we report the absolute score together with its improvement over the baseline.

Table 7: Diversity and image quality after 800 training steps. Values in parentheses denote diversity retention relative to SDXL-Turbo for LPIPS and DINO, and absolute improvement over SDXL-Turbo for MUSIQ and CLIP-IQA.

| Method | LPIPS (ret.) ↑ | DINO div. (ret.) ↑ | MUSIQ (Δ) ↑ | CLIP-IQA (Δ) ↑ |
|---|---|---|---|---|
| SDXL-Turbo | 0.5984 (100.0%) | 0.1952 (100.0%) | 70.96 (+0.00) | 0.6600 (+0.0000) |
| PickScore (step 800) | 0.6054 (101.2%) | 0.1942 (99.5%) | 72.67 (+1.72) | 0.7192 (+0.0592) |
| Aesthetic (step 800) | 0.5938 (99.2%) | 0.1947 (99.7%) | 72.51 (+1.55) | 0.7335 (+0.0734) |
| ImageReward (step 800) | 0.6275 (104.9%) | 0.2059 (105.5%) | 73.93 (+2.97) | 0.7306 (+0.0706) |

As shown in Fig. 7 and Tab. 7, reward optimization does not induce diversity collapse. At step 800, PickScore retains 101.2% and 99.5% of the baseline LPIPS and DINO diversity, respectively. The Aesthetic model shows only a marginal reduction, retaining 99.2% LPIPS diversity and 99.7% DINO diversity. ImageReward increases both measures to 104.9% and 105.5%, respectively. Across the entire optimization trajectory, the minimum observed retention is 97.7% for LPIPS and 98.0% for DINO, indicating that the seed-dependent variation is largely preserved.

At the same time, all three optimized models improve both no-reference quality metrics. At step 800, MUSIQ increases by $+1.72$, $+1.55$, and $+2.97$ for PickScore, Aesthetic, and ImageReward, respectively. CLIP-IQA similarly improves by $+0.0592$, $+0.0734$, and $+0.0706$. These improvements appear throughout training rather than only at the final checkpoint. Overall, the results show that Local GRPO improves the target reward without sacrificing sample diversity and simultaneously improves no-reference perceptual quality. We note that these measurements characterize within-prompt diversity and perceptual quality, rather than distribution-level fidelity such as FID.

**Cross-reward evaluation.** To further examine whether gains on the target reward come at the expense of other preference signals, we evaluate each single-reward LeNO model on all three rewards

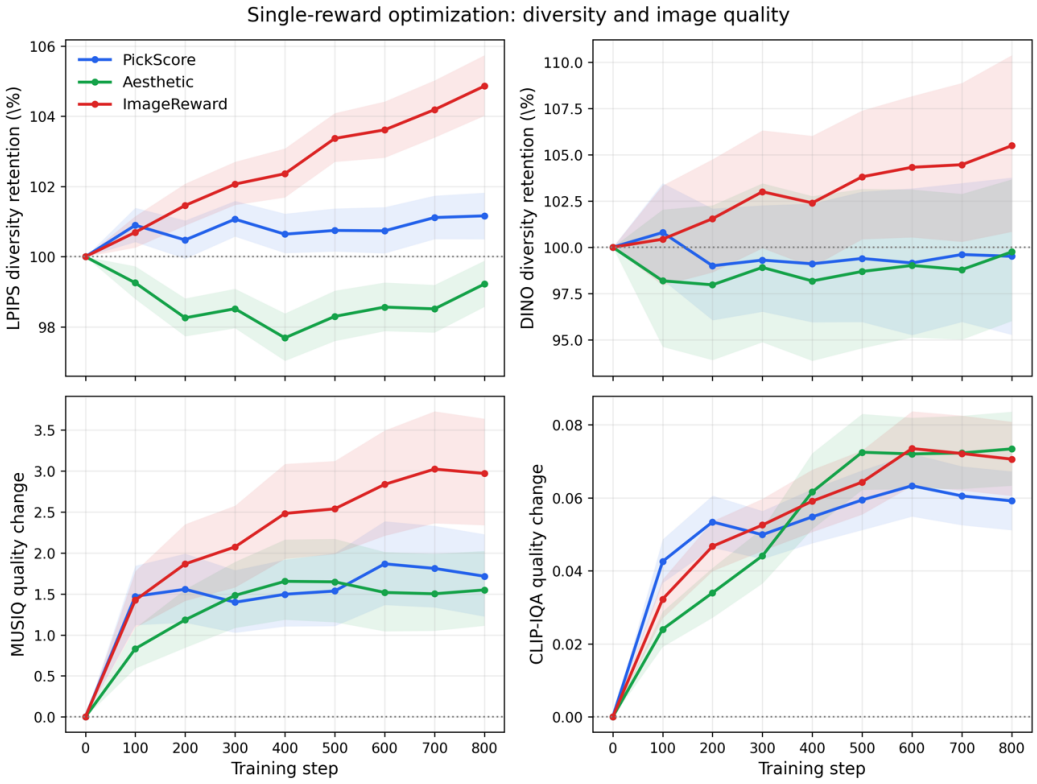



Figure 7: Diversity and no-reference image quality throughout single-reward optimization. LPIPS and DINO diversity are normalized by the unoptimized SDXL-Turbo baseline, while MUSIQ and CLIP-IQA are shown as absolute improvements over the baseline. All methods preserve nearly all of the baseline diversity, while consistently improving perceptual image quality. Shaded regions indicate 95% paired bootstrap confidence intervals over prompts.

Table 8: Cross-reward evaluation of single-reward LeNO models on SDXL-Turbo. Rows denote the training reward and columns the evaluation reward. Values in parentheses denote improvement over SDXL-Turbo; shaded cells indicate the target reward.

| Training reward | Evaluation reward | | |
|---|---|---|---|
| | Aesthetic ↑ | PickScore ↑ | ImageReward ↑ |
| SDXL-Turbo (no training) | 6.1174 | 21.8608 | 0.9710 |
| Aesthetic | **6.1796** (+0.0622) | 21.9728 (+0.1120) | 1.0297 (+0.0587) |
| PickScore | 6.1688 (+0.0514) | **22.3308** (+0.4701) | 1.1306 (+0.1596) |
| ImageReward | 6.1558 (+0.0384) | 21.9092 (+0.0484) | **1.1984** (+0.2274) |

using the same held-out prompt–seed pairs. As shown in Tab. 8, each model achieves its largest improvement on its own training reward (diagonal), while the two non-target rewards also increase in every case. For example, the PickScore-optimized model improves ImageReward by $+0.160$, and the ImageReward-optimized model improves both Aesthetic and PickScore. No non-target reward decreases on average, indicating that LeNO does not exploit idiosyncrasies of a single reward model but instead shifts generations toward outputs that are broadly preferred.

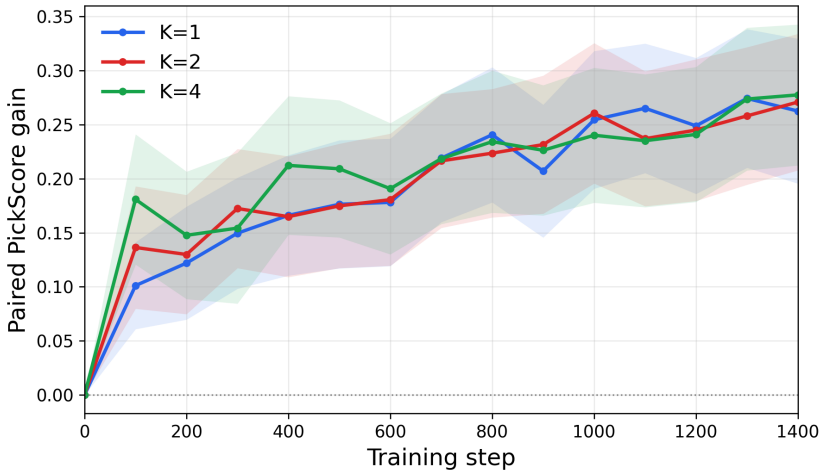



Figure 8: Ablation for multi-step transport reward gain.

### E.2 Multi-step Correction Ablation

We additionally investigate whether applying the learned noise correction over multiple residual steps provides finer-grained optimization than the single-step formulation used in the main experiments. We consider $k \in \{1, 2, 4\}$ residual transport steps. For the multi-step variants, we use a branching strategy in which stochastic perturbations are introduced at only one of the $k$ steps, while the remaining steps follow deterministic residual transitions. Concretely, at the selected branching step, multiple local particles are generated around the predicted correction and evaluated using the terminal reward; before and after this branching point, the transport is applied deterministically. This preserves the local reward comparison used by Local GRPO without exponentially branching the trajectory at every step.

Figure 8 compares the reward gain throughout training for $k = 1, 2$, and $4$. All three settings exhibit nearly identical improvement curves, and increasing the number of residual steps does not lead to a clear additional gain. Although a multi-step transport is in principle more expressive and could progressively refine the corrected noise, we do not observe such a benefit in this setting. One possible explanation is that the single-step state-dependent correction already captures most of the useful local reward-improvement direction. Moreover, under the branching strategy, only one step receives stochastic local exploration in each branched trajectory, while the remaining steps compose deterministic transformations. Therefore, increasing $k$ does not necessarily provide additional independent reward-improvement signals. These results suggest that iterative refinement is not essential for the tasks considered here, motivating the use of the simpler single-step transport in our main experiments.

### E.3 KL Regularization Ablation

We study the effect of the KL regularization weight $\beta$, which controls how strongly the learned residual transport is constrained to remain close to the reference noise distribution. We evaluate $\beta \in \{0, 0.01, 0.1, 1.0\}$ using ImageReward as the optimization objective.

As shown in Fig. 9, a small amount of KL regularization improves both the final reward gain and training stability. Without KL regularization ($\beta = 0$), LeNO still improves the reward substantially, but achieves a lower final gain ($+0.1889 \pm 0.1038$) and win rate ($69\%$) than $\beta = 0.01$, which obtains the best result ($+0.2504 \pm 0.0887$, $76\%$ win rate). Increasing the regularization further reduces the attainable improvement: $\beta = 0.1$ yields a gain of $+0.1433 \pm 0.0543$, while $\beta = 1.0$ nearly suppresses the reward improvement entirely.

These results suggest that the KL term plays more than a purely reward-regularization role. When its weight is zero, the residual transport can move the corrected-noise distribution more freely away from the reference prior, which may weaken the local structure assumed by the residual parame-

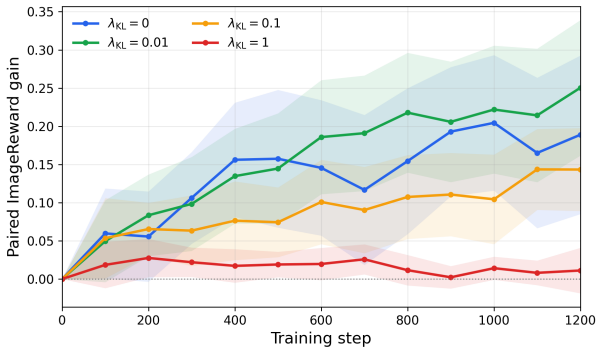



Figure 9: Ablation of the KL regularization weight $\beta$.

Table 9: Effect of $\beta$ on ImageReward gain and win rate.

| $\beta$ | ImageReward Gain | Win Rate |
|---|---|---|
| 0 | 0.1889 | 69% |
| 0.01 | **0.2504** | **76%** |
| 0.1 | 0.1433 | 68% |
| 1.0 | 0.0112 | 59% |

terization and lead to less stable optimization. A small KL weight acts as a trust-region constraint, allowing sufficient movement toward high-reward regions while preventing excessively large distributional shifts. Conversely, overly strong regularization restricts the correction magnitude and prevents the transport from reaching high-reward noise states. Accordingly, an intermediate KL weight provides the best trade-off between reward improvement and preservation of the reference noise structure. This behavior is consistent with the reward-tilted target, which balances reward maximization against deviation from the reference noise distribution rather than optimizing reward alone.

### E.4 Generalization to a One-Step GAN Generator

We further evaluate whether LeNO is applicable beyond diffusion-based generators by applying it to DF-GAN (Tao et al., 2022), a one-step prompt-conditioned GAN. We use the same LeNO training setup as in the main experiments, keeping the pretrained generator frozen and optimizing only the residual correction from scalar reward evaluations. Since LeNO operates entirely in the initial noise space and does not require gradients through either the generator or the reward function, no modification of the generator architecture or its internal sampling process is required.

Figure 10 shows qualitative examples before and after LeNO correction to PickScore. Across different prompts, the corrected initial noises produce samples with improved reward-aligned visual characteristics while preserving the underlying generation capability of the frozen DF-GAN. Importantly, this experiment uses the same optimization procedure despite the substantially different generator family. The result therefore supports that LeNO is not tied to a particular diffusion formulation or denoising trajectory, but can be applied to deterministic prompt-conditioned generators whenever they expose an initial noise input and can be evaluated by a scalar reward oracle. This provides additional evidence for the black-box applicability of LeNO to both the generator and the reward function.

## F Additional Results




Figure 10: Qualitative comparison of DF-GAN generated samples before (left) and after LeNO noise correction (right).

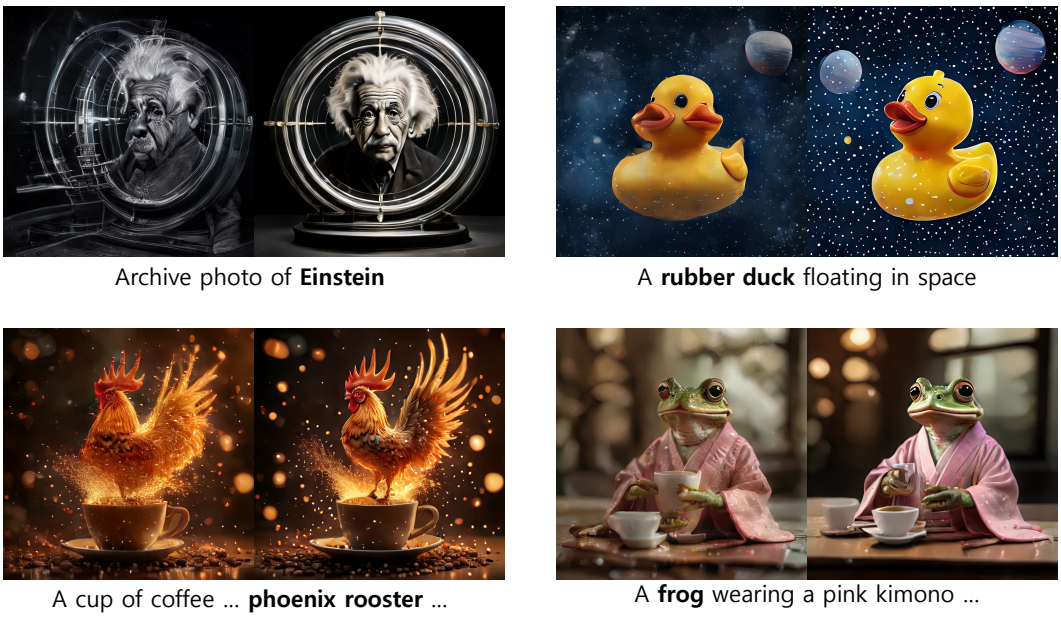



Figure 11: Additional qualitative comparison of SANA-Sprint optimized with compositional multi-rewards.



**Water bottle**



A beautiful **plant**, **aesthetic**, ...




**Frozen food logo**



... mountains with some **forests**, small **lake** ...

Figure 12: Additional qualitative comparison of SDXL-Turbo optimized with Image Rewards.




Red stop sign, school bus ...



A head ... Studio Harcourt style ...



Watercolor painting of woodpecker



A man as a were wolf ...

Figure 13: Additional qualitative comparison of SDXL-Turbo optimized with PickScore.

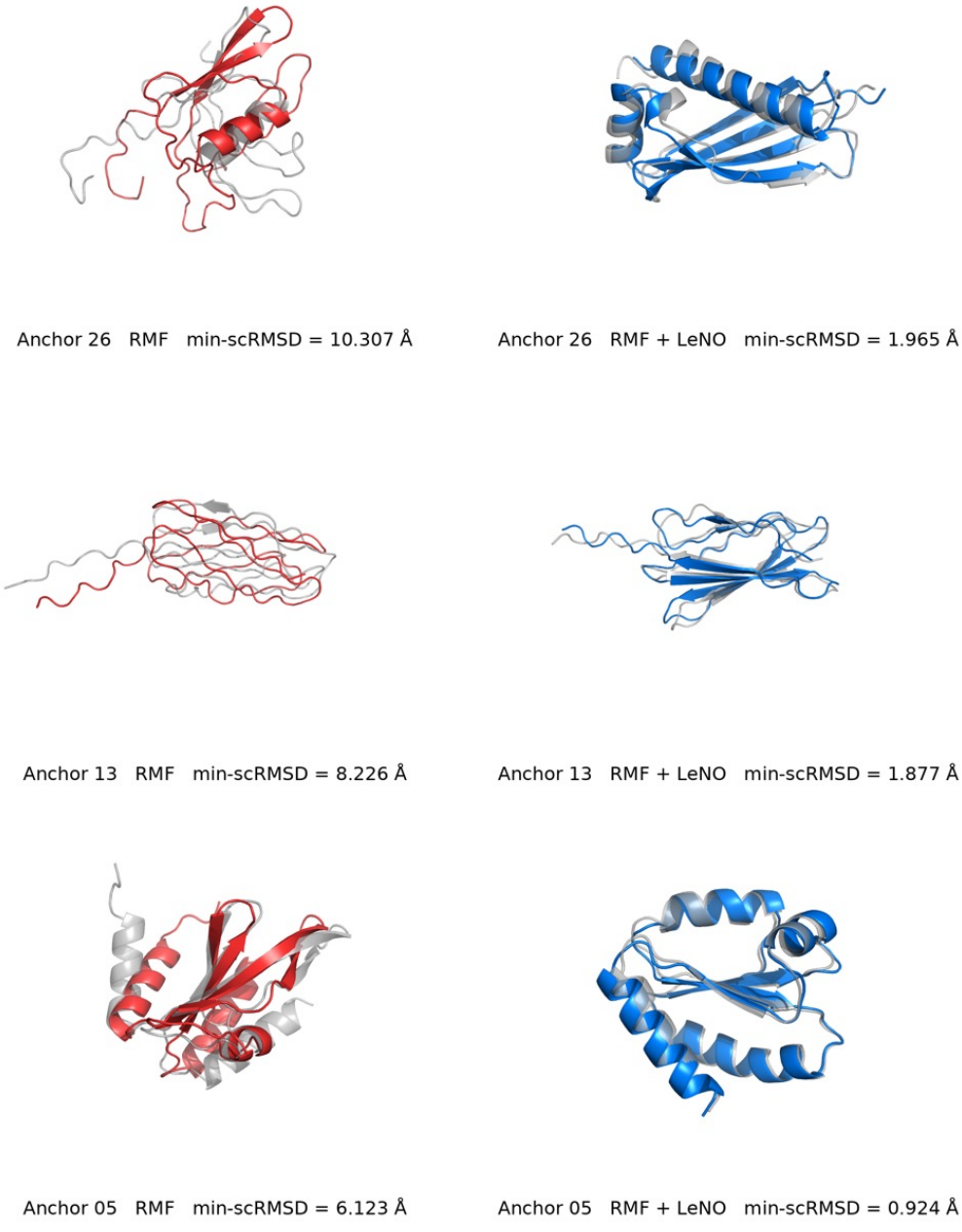



Figure 14: Additional qualitative self-consistency comparison with RMF at protein length $L$=96.