GRADIENT-BASED TIMESTEP-PARTITIONED LORA FOR REWARD FINE-TUNING OF DIFFUSION MODELS

Anonymous authors
Paper under double-blind review



Figure 1: Qualitative comparison of baseline methods and GTP-LoRA trained using SDXL as base model, maximizing CLIP, HPS-v2.1, PickScore, ImageReward, and Aesthetic scores.

ABSTRACT

Reward-based fine-tuning enables diffusion models to optimize human preferences and task-specific objectives, but its optimization signals can be highly imbalanced across denoising timesteps. Prior work has identified this issue in policy-gradient-based alignment and addressed it through objective- or scheduler-level correction. We show that concentrated timestep-wise gradients also arise in direct-gradient optimization and that their temporal profile can differ from on-policy reward sensitivity. When a single LoRA adapter is shared across the entire denoising trajectory, concentrated gradients can bias its adaptation toward particular timestep regimes. We propose *Gradient-based Timestep-Partitioned LoRA* (GTP-LoRA), which partitions the trajectory into contiguous segments and assigns a separate LoRA expert to each segment. Segment boundaries are determined from a reference denoiser-output gradient profile such that each expert receives a comparable cumulative upstream gradient magnitude. This timestep-partitioned design confines gradient updates to their assigned denoising regimes, limiting the disproportionate influence of high-gradient timesteps on a single shared adapter. Experiments across multiple diffusion models, reward objectives, and optimization algorithms show that GTP-LoRA consistently improves reward performance while maintaining competitive sample diversity. The source code is included in the supplementary materials.

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

Diffusion models have emerged as a powerful class of generative models, enabling high-quality and diverse image synthesis from natural-language prompts Ho et al. (2020); Rombach et al. (2022). Nevertheless, models trained with conventional noise-prediction objectives are not necessarily aligned with downstream preferences or task-specific criteria. Recent work therefore performs reward-based fine-tuning using signals derived from text-alignment, human preferences, aesthetic quality, and other task-specific criteria Black et al. (2024); Fan et al. (2023); Prabhudesai et al. (2023); Clark et al. (2024); Wu et al. (2024); Lee & Ye (2026); Kang et al. (2026). These methods broadly fall into policy-gradient and direct-gradient optimization, which optimize terminal reward through policy-gradient estimation and direct backpropagation through the denoising trajectory, respectively.

Prior work has shown that policy-gradient-based diffusion fine-tuning can exhibit timestep-wise gradient imbalance induced by reverse-process coefficients Lee & Ye (2026). Extending this analysis to direct-gradient optimization, we find that both paradigms concentrate gradients toward late reverse-denoising steps, albeit with different temporal profiles. The policy-gradient signal contains an explicit local reverse-process scaling, whereas the direct-gradient signal additionally depends on transition Jacobians accumulated along the remaining trajectory. Accordingly, scheduler corrections tailored to policy-gradient do not capture the full structure of the direct-gradient signal.

We further compare these gradient profiles with on-policy reward sensitivity and observe contrasting temporal patterns: the measured reward sensitivity is higher at early reverse-denoising steps, whereas gradients under both optimization paradigms concentrate toward later steps. This contrast is particularly relevant in parameter-efficient fine-tuning, where gradients from all timesteps are aggregated through a single shared LoRA adapter. High-gradient late-stage regimes can exert a disproportionate influence on the adapter update, leaving lower-gradient early regimes, despite their higher measured on-policy reward sensitivity, comparatively underrepresented.

To mitigate the disproportionate influence of high-gradient timestep regimes under shared LoRA adaptation, we propose *Gradient-based Timestep-Partitioned LoRA* (GTP-LoRA). GTP-LoRA partitions the denoising trajectory into contiguous segments with comparable cumulative denoiser-output gradient magnitude and assigns a separate LoRA expert to each segment. By restricting gradients from each segment to its corresponding adapter subspace, GTP-LoRA limits the extent to which high-gradient regimes dominate a single adapter shared across timesteps. Because it operates in parameter space without modifying the reward objective or the sampling scheduler, GTP-LoRA is compatible with both policy-gradient and direct-gradient optimization methods.

Experiments across multiple base models, reward objectives, and policy- and direct-gradient optimization methods show that GTP-LoRA yields higher reward performance across all evaluated settings. Single-reward evaluations further show that these gains are achieved while maintaining competitive sample diversity. Since only one LoRA expert is active at most timesteps, with two experts mixed only at segment boundaries, GTP-LoRA also incurs minimal additional adapter computation at inference time.

Our main contributions are summarized as follows:

- We characterize the contrasting temporal profiles of on-policy reward sensitivity and the gradients induced by policy-gradient and direct-gradient optimization. We further derive the timestep-wise denoiser-output gradients for both paradigms, revealing the roles of reverse-process coefficients and, in direct-gradient optimization, accumulated transition Jacobians.
- We propose GTP-LoRA, which partitions the denoising trajectory into contiguous segments and assigns a separate LoRA expert to each segment. The segment boundaries are determined from a reference denoiser-output gradient profile to balance the cumulative upstream gradient magnitude across experts.
- Across multiple diffusion models, reward objectives, and policy- and direct-gradient methods, we show that GTP-LoRA improves reward performance across the evaluated settings while retaining competitive sample diversity and incurring minimal additional inference-time adapter computation.

2 Related Work

Timestep-wise Diffusion Experts. In conventional diffusion model training, a single denoiser is typically shared across the entire noise schedule, despite substantial variations in the input distribution and functional role of denoising across timesteps. Early studies Balaji et al. (2022); Lee et al. (2024); Zhang et al. (2024) therefore partition the denoising trajectory into multiple intervals and assign diffusion experts or stage-specific architectures to different noise regimes. More recent studies Zhuang et al. (2025); Zhao et al. (2026) achieve similar timestep specialization in a parameter-efficient manner, retaining a shared pretrained denoiser while introducing lightweight LoRA experts that are selected or combined according to the current timestep.

Existing approaches broadly adopt either fixed expert assignments or learnable routing. The former Balaji et al. (2022); Lee et al. (2024); Zhang et al. (2024) partition the noise schedule into fixed intervals defined by simple heuristics, or determine the interval boundaries offline based on the similarity between denoising tasks. The latter Zhuang et al. (2025); Zhao et al. (2026) learn timestep-conditioned routing weights to select or combine multiple experts, enabling soft specialization without relying exclusively on fixed boundaries.

Timestep-wise Optimization Interference. In diffusion training, gradients from distant timesteps are often weakly aligned or conflicting, such that optimizing one noise regime can degrade performance at others Hang et al. (2023); Go et al. (2023); Ma et al. (2025). Differences in gradient magnitude and loss scale have also been identified as sources of negative transfer, as stronger denoising tasks can dominate the aggregate update and suppress weaker ones Go et al. (2023). These timestep-wise disparities are mediated by the noise-conditioned objective, prediction parameterization, and effective loss weighting. In this work, we investigate an additional source of timestep-wise imbalance arising from reward-based diffusion optimization, where reverse-process terms explicitly modulate the gradient at each timestep.

3 Preliminaries

Diffusion Model Diffusion models Ho et al. (2020) define a forward process that gradually perturbs a data sample $\mathbf{x}_0$ with Gaussian noise. Given a predefined variance schedule $\{\beta_t\}_{t=1}^T$, the noisy sample at timestep t can be directly expressed as

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. The reverse process is parameterized by a neural network $\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})$, which predicts the noise contained in $\mathbf{x}_t$, conditioned on $\mathbf{t}$ and $\mathbf{c}$.

In conventional noise-prediction training, the model is optimized using the mean-squared error between the sampled noise and the predicted noise:

$$\mathcal{L}_{\mathrm{DDPM}}(\theta) = \mathbb{E}_{\mathbf{x}_0,\mathbf{c},t,\epsilon}\left[\|\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon_t\|_2^2\right]. \tag{2}$$

At inference time, generation starts from Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and progressively denoises it through DDIM-style reverse transition Song et al. (2021)

$$\mathbf{x}_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}\mathbf{x}_t + c_t\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) + \sigma_t\boldsymbol{\epsilon}_t \tag{3}$$

where c_t and σ_t are timestep-dependent coefficients determined by the underlying noise schedule.

Diffusion Models with Reward Optimization. Following the formulation in Black et al. (2024), diffusion-model generation can be viewed as a finite-horizon Markov decision process over the reverse denoising trajectory $\tau = (\mathbf{x}_T, \ldots, \mathbf{x}_0)$. At each timestep, the state and action are defined as $\mathbf{s}_t \triangleq (\mathbf{c}, t, \mathbf{x}_t)$ and $\mathbf{a}_t \triangleq \mathbf{x}_{t-1}$, respectively. The reward is assigned to the generated sample $\mathbf{x}_0$, such that $R(\mathbf{s}_t) \triangleq r(\mathbf{x}_0, \mathbf{c})$ when $t = 0$ and $R(\mathbf{s}_t) \triangleq 0$ otherwise.

Under this sequential formulation, existing methods can be broadly categorized according to how they estimate the gradient of the expected terminal reward: *policy-gradient optimization* and *direct-gradient optimization*. Both paradigms share the same expected terminal-reward objective,

$$J(\theta) = \mathbb{E}_{\mathbf{c},\,\tau\sim p_\theta(\tau|\mathbf{c})}\left[r(\mathbf{x}_0, \mathbf{c})\right], \tag{4}$$

where $p_\theta(\tau \mid \mathbf{c})$ denotes the trajectory distribution induced by the reverse denoising process.

Policy-gradient methods Black et al. (2024); Fan et al. (2023); Lee & Ye (2026) interpret each reverse denoising transition as a stochastic policy, $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) \triangleq p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$, and estimate the gradient of Eq. 4 using the score-function identity:

$$\nabla_\theta J_{PG}(\theta) = \mathbb{E}\left[r(\mathbf{x}_0, \mathbf{c}) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, t, \mathbf{c}) \right]. \tag{5}$$

Thus, the same terminal reward weights the local score-function gradient of every transition along the sampled trajectory.

Alternatively, direct-gradient methods Prabhudesai et al. (2023); Clark et al. (2024); Xu et al. (2023); Wu et al. (2024); Kang et al. (2026) reparameterize the reverse denoising trajectory using initial noise $\mathbf{x}_T$ and sampled transition noises $\boldsymbol{\epsilon}_{1:T}$. Denoting the resulting final sample by $\mathbf{x}_0(\theta)$ and assuming a differentiable reward, they estimate the gradient of Eq. 4 using the pathwise derivative:

$$\nabla_\theta J_{DG}(\theta) = \mathbb{E}\left[\frac{\partial r(\mathbf{x}_0, \mathbf{c})}{\partial \mathbf{x}_0} \frac{\partial \mathbf{x}_0}{\partial \theta} \right]. \tag{6}$$

Treating $\mathbf{x}_T$ and $\boldsymbol{\epsilon}_{1:T}$ as fixed with respect to θ, each transition in Eq. 3 can be written as $\mathbf{x}_{t-1} = f_t(\mathbf{x}_t; \theta, \mathbf{c}, \boldsymbol{\epsilon}_t)$. The final-sample gradient then follows the chain rule:

$$\frac{\partial \mathbf{x}_0}{\partial \theta} = \sum_{t=1}^{T} \left(\prod_{s=1}^{t-1} \frac{\partial f_s}{\partial \mathbf{x}_s} \right) \frac{\partial f_t}{\partial \theta}. \tag{7}$$

Therefore, whereas policy-gradient methods decompose the gradient into local score-function terms, direct-gradient methods obtain timestep-wise contributions by backpropagating the reward through the reverse denoising process. Eq. 5 and 6 describe canonical estimators for these two optimization families. While practical algorithms may incorporate weighting, clipping, truncated backpropagation, timestep subsampling, or surrogate objectives, we use these forms only to identify the reverse-process factors responsible for timestep-dependent gradient scaling.

4 Analysis of Timestep-wise Imbalance in Reward Optimization

Timestep-wise Reward Sensitivity. To characterize the reward sensitivity of the policy's one-step decisions at each timestep, we consider the action-wise variance of the Q function:

$$\mathrm{Var}_{\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t)} \left[Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{8}$$

where

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_\pi \left[r(\mathbf{x}_0, \mathbf{c}) \mid \mathbf{s}_t, \mathbf{a}_t \right] \tag{9}$$

denotes the expected terminal reward after taking action $\mathbf{a}_t$ at state $\mathbf{s}_t$ and following the current policy thereafter. Holding $\mathbf{s}_t$ fixed, the variance measures how much the expected terminal reward varies over the policy-induced action distribution at timestep t. A large variance therefore indicates that the denoising action taken at timestep t can substantially affect the expected terminal reward. For Fig. 2 (a), we average this state-conditional variance over reference states obtained from trajectories sampled across 10 prompts and 5 random seeds using the pretrained policy. Details of the noise-corrected nested Monte Carlo estimator are provided in Appendix A.

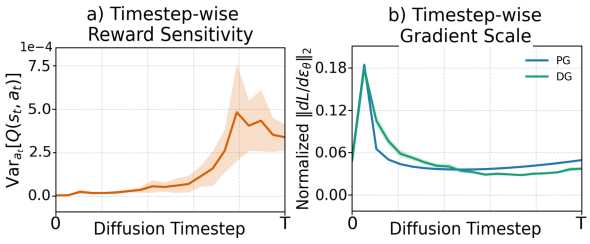


Figure 2: Contrasting timestep-wise profiles of on-policy reward sensitivity and optimization signal. (a) Monte Carlo estimate of the Q variance $\mathrm{Var}_{\mathbf{a}_t}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$ under the CLIP reward. A larger value indicates greater variation in the expected terminal reward over actions sampled from the current policy at timestep t. (b) Timestep-wise norm of the denoiser-output gradient $\|\partial \mathcal{L} / \partial \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})\|_2$ during policy-gradient (DDPO) and direct-gradient (SQDF) optimization. Each gradient profile is normalized across timesteps for visualization.

As shown in Fig. 2 (a), the Q variance increases substantially toward high diffusion timesteps, which correspond to the early stages of reverse denoising. This trend indicates that early denoising decisions exhibit greater reward influence under on-policy action perturbations than later refinements.

Gradient Imbalance in Reward Optimization. Prior work has shown that conventional diffusion training can exhibit cross-timestep gradient conflicts and scale differences. Following Eq. 2, we denote the gradient associated with timestep t as

$$\nabla_\theta^{(t)} \mathcal{L}_{\mathrm{DDPM}}(\theta) = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t^{\mathrm{DDPM}}\right], \tag{10}$$

where

$$\mathbf{J}_{\theta,t} = \frac{\partial \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})}{\partial \theta}, \qquad \boldsymbol{\delta}_t^{\mathrm{DDPM}} = 2\left(\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon_t\right). \tag{11}$$

Here, $\nabla_\theta^{(t)}$ denotes the gradient term obtained by conditioning Eq. 2 on timestep t. Thus, timestep-wise differences in conventional diffusion training can arise implicitly from the local parameter Jacobian $\mathbf{J}_{\theta,t}$ and the prediction residual $\boldsymbol{\delta}_t^{\mathrm{DDPM}}$, potentially producing gradient conflicts and scale imbalance. Since $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ has the same marginal distribution across timesteps, the standard unweighted objective introduces no explicit timestep-dependent coefficient in $\boldsymbol{\delta}_t^{\mathrm{DDPM}}$.

Reward-based optimization, by contrast, introduces reverse-process factors that explicitly modulate the denoiser-output gradient (δ_t) before it is mapped into parameter space. For policy-gradient optimization, the gradient associated with timestep t in Eq. 5 can be written as

$$\nabla_\theta^{(t)} J_{\mathrm{PG}}(\theta) = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t^{\mathrm{PG}}\right], \tag{12}$$

where the corresponding denoiser-output gradient is

$$\boldsymbol{\delta}_t^{\mathrm{PG}} = r(\mathbf{x}_0, \mathbf{c}) \frac{c_t}{\sigma_t} \boldsymbol{\epsilon}_t \tag{13}$$

Thus, policy-gradient optimization introduces the local reverse-scheduler factor c_t/σ_t, which explicitly rescales the upstream gradient across timesteps.

For direct-gradient optimization, the gradient associated with timestep t in Eq. 6 can similarly be expressed as

$$\nabla_\theta^{(t)} J_{\mathrm{DG}}(\theta) = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t^{\mathrm{DG}}\right]. \tag{14}$$

Defining

$$A_s = \sqrt{\frac{\bar{\alpha}_{s-1}}{\bar{\alpha}_s}}, \qquad \mathbf{J}_{x,s} = \frac{\partial \epsilon_\theta(\mathbf{x}_s, s, \mathbf{c})}{\partial \mathbf{x}_s}, \tag{15}$$

the corresponding denoiser-output gradient is

$$\boldsymbol{\delta}_t^{\mathrm{DG}} = c_t \left\{ \prod_{s=1}^{t-1} \left(A_s \mathbf{I} + c_s \mathbf{J}_{x,s}\right) \right\}^\top \nabla_{\mathbf{x}_0} r(\mathbf{x}_0, \mathbf{c}). \tag{16}$$

Unlike policy-gradient optimization, the direct-gradient signal is modulated by both the local reverse-process coefficient c_t and accumulated reverse-transition Jacobians along the remaining denoising trajectory. Detailed derivations of Eq. 12 – 16 are provided in Appendix C.

Figure 2 (b) shows that the denoiser-output gradients of both optimization families are concentrated at a few late reverse-denoising steps, opposite to the reward-sensitivity trend in Fig. 2 (a). This indicates that the optimization signal is stronger in timestep regimes that are comparatively less sensitive to the terminal reward. Consequently, when all timesteps update a shared adapter, regimes with persistently larger gradient norms can exert disproportionate influence on its optimization trajectory, leaving weaker timestep signals comparatively underrepresented. To ensure that this tendency persists after the denoiser-output gradient is mapped into parameter space, we additionally analyze timestep-wise parameter-gradient norms incorporating the local parameter Jacobian $\mathbf{J}_{\theta,t}$. As shown in Appendix D, the late-step concentration is largely preserved after this Jacobian mapping.

5 METHOD

In policy-gradient optimization, the local factor c_t/σ_t explicitly scales the gradient at each timestep. PCPO Lee & Ye (2026) addresses this imbalance by rescheduling σ_t to approximately equalize c_t/σ_t across timesteps. However, enforcing a uniform c_t/σ_t requires modifying σ_t and the coupled coefficient c_t, potentially departing substantially from the original reverse dynamics and affecting

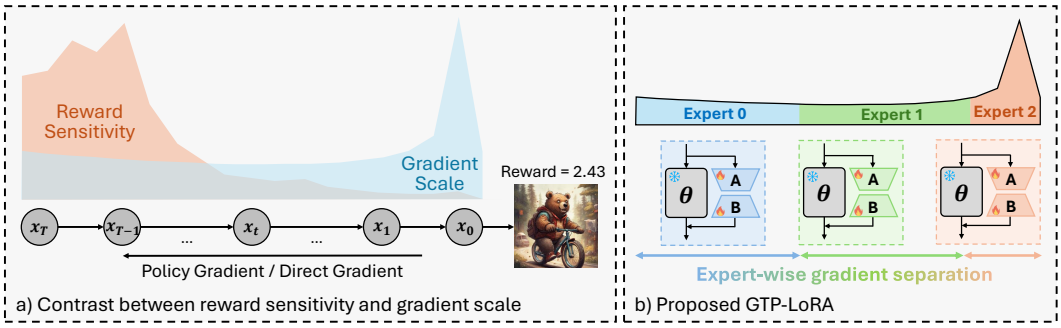


Figure 3: Overview of the proposed GTP-LoRA. (a) On-policy reward sensitivity is higher at early reverse-denoising steps, whereas optimization gradients tend to concentrate toward later steps. Their opposing temporal profiles indicate that the observed gradient concentration does not coincide with the timesteps showing greater reward sensitivity. (b) GTP-LoRA partitions the denoising trajectory into $K = 3$ contiguous, gradient-balanced segments and assigns a separate LoRA expert to each, separating updates across denoising regimes while preserving parameter sharing among neighboring timesteps.

optimization behavior. Also, this scheduler-based correction does not directly extend to direct-gradient optimization. As shown in Eq. 16, the direct-gradient signal depends on both the local coefficient c_t and accumulated reverse-transition Jacobians. Moreover, a valid DDIM stochasticity schedule that keeps c_t constant across the full trajectory does not generally exist under practical sampling regimes; we provide the exact feasibility condition in Appendix E.

This motivates us to develop a unified approach that directly addresses the observed timestep-wise gradient imbalance across optimization paradigms. Rather than modifying the sampling scheduler or reweighting the objective, we focus on the shared-adapter aggregation through which imbalanced timestep gradients interact. Specifically, we target the disproportionate influence of high-gradient timestep regimes on the aggregate update, which can bias a shared adapter toward those regimes.

Gradient-based Timestep-Partitioned LoRA. To this end, we propose *Gradient-based Timestep-Partitioned LoRA* (GTP-LoRA), which assigns separate LoRA experts to contiguous segments of the denoising trajectory. The segment boundaries are determined from a reference denoiser-output gradient profile such that each segment contains approximately the same cumulative upstream gradient magnitude. By confining timestep-wise gradient contributions to their corresponding expert subspaces, GTP-LoRA limits the influence of high-gradient regimes to their local experts, rather than allowing them to disproportionately shape a single adapter shared across the entire trajectory. Figure 3 provides an overview of the motivation and overall design of GTP-LoRA.

Before training, we generate a fixed reference set of trajectories using the initial model and measure the objective-specific denoiser-output gradient at each timestep.

$$\boldsymbol{\delta}_t^{(n)} = \frac{\partial \mathcal{L}^{(n)}}{\partial \epsilon_{\theta_0}\left(\mathbf{x}_t^{(n)}, t, \mathbf{c}^{(n)}\right)}, \tag{17}$$

where θ_0 denotes the initial pretrained model parameters and $\mathcal{L}^{(n)}$ denotes the sample-wise optimization loss. We then construct the timestep-wise gradient profile

$$\widehat{g}_t = \frac{1}{N} \sum_{n=1}^{N} \left\| \boldsymbol{\delta}_t^{(n)} \right\|_2 \tag{18}$$

Computed before mapping through the local parameter Jacobian $\mathbf{J}_{\theta,t}$, $\hat{g}_t$ captures the objective- and reverse-process-induced upstream imbalance without directly incorporating timestep variation in $\mathbf{J}_{\theta,t}$. We therefore use this initialization-time profile to set fixed expert boundaries; Appendix D further shows that this upstream imbalance is also reflected in parameter space after Jacobian mapping.

Given K LoRA experts, we partition the trajectory according to the cumulative gradient profile. Let

$$G(t) = \sum_{s=1}^{t} \widehat{g}_s \tag{19}$$

denote the cumulative gradient profile. Using a piecewise-linear interpolation of G, we define the k-th boundary b_k to satisfy

$$G(b_k) = \frac{k}{K} G(T), \qquad k = 1, \ldots, K-1, \tag{20}$$

with $b_0 = 0$ and $b_K = T$. Consequently, each segment $[b_{k-1}, b_k]$ contains the same cumulative denoiser-output gradient magnitude, $G(T)/K$, under the piecewise-linear reference profile.

If $b_k \in (t-1, t)$, timestep t straddles two segments and is routed to the adjacent experts in proportion to their interval overlap. Let $\rho_{t,k} \in [0, 1]$ denote the fraction assigned to expert k. The resulting adapter residual is

$$\Delta \mathbf{h}_t = \rho_{t,k} \Delta \mathbf{h}_t^{(k)} + (1 - \rho_{t,k})\, \Delta \mathbf{h}_t^{(k+1)}. \tag{21}$$

where $\Delta \mathbf{h}_t^{(k)}$ denotes the output of the k-th LoRA expert. This interpolation smooths the adapter transition at expert boundaries; all other timesteps use a single expert.

We use contiguous intervals to preserve the coarse-to-fine temporal structure of denoising, where different stages emphasize global composition, appearance, and fine details Wang & Vastola (2023); Tinaz et al. (2025); Falck et al. (2025). While GTP-LoRA does not assign fixed semantic roles, it shares parameters among neighboring timesteps while separating distant regimes, with gradient-balanced boundaries distributing the cumulative upstream signal across experts.

Because only boundary timesteps activate two experts, GTP-LoRA performs at most $T + K - 1$ expert evaluations, compared with T for vanilla LoRA. The resulting LoRA-specific overhead at inference time is at most $(K - 1)/T$.

6 Experiments

6.1 Experiment Setup

Base Models and Baselines. We evaluate GTP-LoRA under both policy-gradient and direct-gradient optimization. For policy-gradient optimization, we use DDPO Black et al. (2024) with Stable Diffusion v1.4 (SD1.4) and SDXL, and compare against PCPO Lee & Ye (2026), which mitigates timestep imbalance by rescheduling the reverse-process variance while retaining the same DDPO-style objective. The SDXL experiments further examine whether GTP-LoRA remains effective at a larger model scale.

For direct-gradient optimization, we use AlignProp Prabhudesai et al. (2023) and SQDF Kang et al. (2026) with SD1.4. Because an analogous rescheduling strategy is infeasible for direct-gradient objectives, we compare vanilla LoRA and GTP-LoRA within each optimization algorithm. Unless otherwise specified, GTP-LoRA uses three experts ($K = 3$) in all experiments.

Prompts and Reward Functions. Following prior work Hu et al. (2025); Lee & Ye (2026), we use the *Animal Activity* prompt dataset for training and evaluation. For the SD1.4 experiments, all models are optimized using the CLIP Radford et al. (2021) reward. For the SDXL experiments, we additionally examine scalability to multi-reward optimization by jointly optimizing CLIP, HPS-v2.1 Wu et al. (2023), PickScore Kirstain et al. (2023), ImageReward Xu et al. (2023), and Aesthetic scores Schuhmann (2022) (i.e., 5-Mix reward), following the same setup as Lee & Ye (2026).

Training and Evaluation. All models are trained for 400 optimization steps, except SQDF, which is trained for 2000 steps because it updates only one denoising step per sampled trajectory. We fine-tune the diffusion models exclusively through LoRA adapters and use the LoRA rank specified in the original implementation of each baseline method. Unless otherwise stated, the remaining method-specific hyperparameters follow the corresponding baseline configurations.

Table 1: Comparison of CLIP alignment and diversity for SD1.4 trained with policy-gradient methods using CLIP reward.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
SD1.4	0.352	0.454	0.451
PCPO	0.414	0.371	0.373
PCPO-base	0.406	0.369	<u>0.375</u>
+GTP-LoRA	**0.425**	<u>0.373</u>	0.353
DDPO	0.404	0.358	0.366
+GTP-LoRA	<u>0.420</u>	**0.413**	**0.384**

Table 2: Comparison of CLIP alignment and diversity for SD1.4 trained with direct-gradient methods using CLIP reward. † indicates training for 2000 optimization steps.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
SD1.4	0.352	0.454	0.451
SQDF†	0.399	0.315	0.319
+GTP-LoRA†	<u>0.407</u>	0.329	0.332
AlignProp	0.398	<u>0.345</u>	**0.375**
+GTP-LoRA	**0.421**	**0.376**	<u>0.356</u>

Table 3: Comparison of reward performance for SDXL trained using 5-Mix reward.

Model	CLIP ↑	HPS ↑	PickScore ↑	ImageReward ↑	Aesthetic ↑
SDXL	0.399	0.304	0.899	1.544	5.842
PCPO	<u>0.402</u>	0.354	0.963	1.950	**6.808**
PCPO-base	0.401	<u>0.356</u>	<u>0.964</u>	<u>1.956</u>	6.630
DDPO	0.391	0.339	0.941	1.889	6.521
+GTP-LoRA	**0.404**	**0.365**	**0.974**	**1.957**	<u>6.749</u>

We evaluate each model using the reward function employed during training with unseen prompts. For single reward experiments, we further examine whether reward optimization preserves sample diversity. Specifically, we measure prompt-wise diversity using pairwise feature distances derived from DINOv2 Oquab et al. (2023) and DreamSim Fu et al. (2023).

All results are reported as averages over three random seeds; the corresponding standard deviations are provided in Appendix I. The best and second-best results are highlighted in **bold** and <u>underline</u>, respectively. Full implementation details are provided in Appendix B.

6.2 Main Results

Policy-Gradient Results. As shown in Table 1, applying GTP-LoRA to DDPO provides the most balanced improvement in reward maximization and diversity preservation, achieving stronger alignment while substantially alleviating the diversity degradation observed with vanilla DDPO. We hypothesize that this diversity preservation arises because GTP-LoRA confines large late-stage gradients primarily to their corresponding local experts, thereby reducing their direct interference with the parameters used at earlier denoising stages. Since these early high-noise stages play an important role in establishing global composition and sample-level variation, preserving their adaptation could help retain greater output diversity.

To more directly compare GTP-LoRA with scheduler-based correction, we additionally apply it to PCPO-base, the log-ratio-based DDPO variant used in PCPO before introducing σ_t rescheduling. Even when applied to PCPO-base, GTP-LoRA yields stronger reward maximization than PCPO, suggesting that separating timestep-specific gradients across adapter subspaces can be more effective than correcting their explicit scheduler-induced scales alone. More qualitative results are provided in Appendix G.

Direct-Gradient Results. As shown in Table 2, GTP-LoRA also provides consistent benefits across different direct-gradient optimization methods. These results demonstrate that the effectiveness of GTP-LoRA is not specific to policy-gradient objectives, but generalizes across direct-gradient algorithms, yielding an overall improved trade-off between reward maximization and diversity preservation.

Large-Model and Multi-Reward Results. Table 3 evaluates GTP-LoRA on a larger diffusion model (SDXL) under multi-reward optimization. Applying GTP-LoRA to DDPO consistently improves all five constituent rewards, demonstrating that its benefits extend across heterogeneous reward signals rather than being confined to a single objective. In contrast, PCPO does not uniformly improve upon PCPO-base: although it increases the CLIP and Aesthetic scores, it slightly degrades HPS-v2.1, PickScore, and ImageReward. We hypothesize that this stems from early-stage training instability induced by changes to the reverse transition dynamics under PCPO rescheduling; enforcing a uniform c_t/σ_t ratio substantially alters both σ_t and the coupled coefficient c_t relative to the original DDIM schedule. A detailed analysis is provided in Appendix E.

6.3 Ablation Study

All ablation studies are conducted with SD1.4 under the DDPO objective using the CLIP reward. We provide additional ablation studies in Appendix F.

Number of Experts. Since GTP-LoRA uses three experts in the main experiments, we additionally evaluate configurations with two and four experts to examine the effect of the number of experts. Although four experts achieve a slightly higher CLIP score, three experts provide the best overall balance between reward performance and sample diversity and are therefore used in the main experiments.

Table 4: Ablation studies on the number of LoRA experts and boundary-timestep routing.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
Number of Experts			
2-Experts	0.416	<u>0.411</u>	0.377
4-Experts	**0.422**	0.391	<u>0.381</u>
3-Experts (Ours)	<u>0.420</u>	**0.413**	**0.384**
Boundary Routing			
Discrete	**0.422**	<u>0.373</u>	<u>0.362</u>
Stochastic	0.415	0.357	0.348
Mixing (Ours)	<u>0.420</u>	**0.413**	**0.384**

Boundary-Timestep Routing. When an expert boundary lies within a timestep interval, GTP-LoRA mixes the outputs of the two adjacent experts, as defined in Eq. 21. We compare this mixing strategy with two alternatives. The *discrete* strategy assigns the boundary timestep exclusively to the expert with the larger overlap, while the *stochastic* strategy samples one of the two adjacent experts according to their overlap ratios. As shown in Table 4, discrete routing slightly improves CLIP but substantially reduces diversity, whereas stochastic routing underperforms in both alignment and diversity. We therefore adopt fractional mixing for its better overall reward–diversity trade-off.

7 Conclusion

In this work, we characterized contrasting timestep-wise profiles in reward-based diffusion fine-tuning: on-policy reward sensitivity is higher at early reverse-denoising steps, whereas optimization gradients tend to concentrate toward later steps. Our analysis further shows how reverse-process coefficients and accumulated transition Jacobians shape these timestep-wise signals. To limit the disproportionate influence of high-gradient regimes on a shared LoRA adapter, we proposed Gradient-based Timestep-Partitioned LoRA (GTP-LoRA). GTP-LoRA partitions the denoising trajectory using a reference denoiser-output gradient profile and assigns separate experts to gradient-balanced segments. Across multiple models, reward objectives, and optimization methods, GTP-LoRA achieves higher mean reward scores while retaining competitive sample diversity with minimal additional active adapter computation. Future work will investigate whether the observed timestep-wise imbalance and GTP-LoRA generalize to continuous-time flow models.

Reproducibility statement

To support reproducibility, we describe the training configurations, GTP-LoRA initialization, baseline implementations, hardware, random seeds, and evaluation protocol in Appendix B. The prompt

construction and train–test split are provided in Appendix H. The source code is included in the supplementary materials.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for assist in the writing of proofs, refine hypotheses, provide feedback on research methodology, and implement methods. We have not used generative AI tools to help develop conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, design or provide feedback on experiments, assist with translation, support qualitative and thematic data analysis, interpret results, and generate synthetic data sets, clean and reformat dataset are not applicable to this work. Additionally, we used generative AI tools for draft parts of a research paper, sourcing/searching for information, edit a research paper to improve readability, and format references. We have reviewed all AI-assisted work. We manually checked the assumptions, intermediate steps, and logical consistency of AI-assisted mathematical derivations and proofs. Suggestions for refining hypotheses and research methodology were critically assessed before incorporation. AI-assisted code was reviewed for consistency with the algorithms described in the paper and tested through training and evaluation runs. References identified or formatted with AI assistance, along with the associated factual statements, were checked against the original sources. AI-generated or edited text was reviewed and revised for technical accuracy and consistency with the intended claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers. arXiv preprint arXiv:2211.01324, 2022. URL https://arxiv.org/abs/2211.01324.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training Diffusion Models with Reinforcement Learning. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, Vienna, Austria, 2024. OpenReview.net. URL https://arxiv.org/abs/2305.13301.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly Fine-Tuning Diffusion Models on Differentiable Rewards. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, Vienna, Austria, 2024. OpenReview.net. URL https://arxiv.org/abs/2309.17400.

Fabian Falck, Teodora Pandeva, Kiarash Zahirnia, Rachel Lawrence, Richard Turner, Edward Meeds, Javier Zazo, and Sushrut Karmalkar. A Fourier Space Perspective on Diffusion Models. arXiv preprint arXiv:2505.11278, 2025. URL https://arxiv.org/abs/2505.11278.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 79858–79885, New Orleans, LA, USA, 2023. Curran Associates, Inc. URL https://arxiv.org/abs/2305.16381.

Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. DreamSim: Learning New Dimensions of Human Visual Similarity using Synthetic Data. In *Advances in Neural Information Processing Systems*, volume 36, pp. 50742–50768, New Orleans, LA, USA, 2023. Curran Associates, Inc. URL https://arxiv.org/abs/2306.09344.

Hyojun Go, JinYoung Kim, Yunsung Lee, Seunghyun Lee, Shinhyeok Oh, Hyeongdon Moon, and Seungtaek Choi. Addressing Negative Transfer in Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023. URL https://arxiv.org/abs/2306.00354.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient Diffusion Training via Min-SNR Weighting Strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7441–7451, Paris, France, 2023. IEEE. URL https://arxiv.org/abs/2303.09556.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://arxiv.org/abs/2006.11239.

Zijing Hu, Fengda Zhang, Long Chen, Kun Kuang, Jiahui Li, Kaifeng Gao, Jun Xiao, Xin Wang, and Wenwu Zhu. Towards Better Alignment: Training Diffusion Models with Reinforcement Learning Against Sparse Rewards. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23604–23614, Nashville, TN, USA, June 2025. IEEE. doi: 10.1109/CVPR52734.2025.02198. URL https://arxiv.org/abs/2503.11240.

Hyeongyu Kang, Jaewoo Lee, Woocheol Shin, Kiyoung Om, and Jinkyoo Park. Diffusion Fine-Tuning via Reparameterized Policy Gradient of the Soft Q-Function. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, Rio de Janeiro, Brazil, 2026. OpenReview.net. URL https://arxiv.org/abs/2512.04559.

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 *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023. URL https://arxiv.org/abs/2305.01569.

Jeongjae Lee and Jong Chul Ye. PCPO: Proportionate Credit Policy Optimization for Aligning Image Generation Models. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, Rio de Janeiro, Brazil, 2026. OpenReview.net. URL https://arxiv.org/abs/2509.25774.

Yunsung Lee, Jin-Young Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-Architecture Multi-Expert Diffusion Models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 13427–13436. AAAI Press, 2024. URL https://arxiv.org/abs/2306.04990.

Qianli Ma, Xuefei Ning, Dongrui Liu, Li Niu, and Linfeng Zhang. Decouple-Then-Merge: Finetune Diffusion Models as Multi-Task Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23281–23290, Nashville, TN, USA, June 2025. IEEE. URL https://arxiv.org/abs/2410.06664.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. arXiv preprint arXiv:2304.07193, 2023. URL https://arxiv.org/abs/2304.07193.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning Text-to-Image Diffusion Models with Reward Backpropagation. arXiv preprint arXiv:2310.03739, 2023. URL https://arxiv.org/abs/2310.03739.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the Thirty-Eighth International Conference on Machine Learning (ICML)*, volume 139, pp. 8748–8763. PMLR, 2021. URL https://arxiv.org/abs/2103.00020.

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 (CVPR)*, pp. 10684–10695, New Orleans, LA, USA, 2022. IEEE. doi: 10.1109/CVPR52688.2022.01042. URL https://arxiv.org/abs/2112.10752.

Christoph Schuhmann. LAION-Aesthetics. LAION Blog, August 2022. URL https://laion.ai/blog/laion-aesthetics/.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *Proceedings of the Ninth International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021. URL https://arxiv.org/abs/2010.02502.

Berk Tinaz, Zalan Fabian, and Mahdi Soltanolkotabi. Emergence and Evolution of Interpretable Concepts in Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. URL https://arxiv.org/abs/2504.15473.

Binxu Wang and John J. Vastola. Diffusion Models Generate Images Like Painters: An Analytical Theory of Outline First, Details Later. arXiv preprint arXiv:2303.02490, 2023. URL https://arxiv.org/abs/2303.02490.

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. URL https://arxiv.org/abs/2306.09341.

Xiaoshi Wu, Yiming Hao, Manyuan Zhang, Keqiang Sun, Zhaoyang Huang, Guanglu Song, Yu Liu, and Hongsheng Li. Deep Reward Supervisions for Tuning Text-to-Image Diffusion Models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 15141, pp. 108–124, Milan, Italy, 2024. Springer. doi: 10.1007/978-3-031-73010-8_7. URL https://arxiv.org/abs/2405.00760.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 15903–15935, New Orleans, LA, USA, 2023. Curran Associates, Inc. URL https://arxiv.org/abs/2304.05977.

Huijie Zhang, Yifu Lu, Ismail Alkhouri, Saiprasad Ravishankar, Dogyoon Song, and Qing Qu. Improving Training Efficiency of Diffusion Models via Multi-Stage Framework and Tailored Multi-Decoder Architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7372–7381, Seattle, WA, USA, 2024. IEEE. URL https://arxiv.org/abs/2312.09181.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Hao Luo, Yibing Song, Gao Huang, Fan Wang, and Yang You. DyDiT++: Diffusion Transformers with Timestep and Spatial Dynamics for Efficient Visual Generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. URL https://arxiv.org/abs/2504.06803.

Shaobin Zhuang, Yiwei Guo, Yanbo Ding, Kunchang Li, Xinyuan Chen, Yaohui Wang, Fangyikang Wang, Ying Zhang, Chen Li, and Yali Wang. TimeStep Master: Asymmetrical Mixture of Timestep LoRA Experts for Versatile and Efficient Diffusion Models in Vision. In *Proceedings of the Forty-Second International Conference on Machine Learning (ICML)*, Vancouver, Canada, 2025. PMLR. URL https://arxiv.org/abs/2503.07416.

A TIMESTEP-WISE REWARD SENSITIVITY

Monte Carlo Estimation of Action-Wise Q Variance. We estimate the action-wise variance of the Q function using nested Monte Carlo rollouts. Specifically, for a fixed state $\mathbf{s}_t$, the target quantity is

$$\mathrm{Var}_{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)} \left[Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{22}$$

The estimator $\widehat{\mathrm{Var}}_t$ defined below approximates this quantity while correcting for the Monte Carlo noise introduced by finite suffix rollouts.

For each prompt $\mathbf{c}$ and random seed, we first generate a reference denoising trajectory. At each timestep t, we fix the state

$$\mathbf{s}_t = (\mathbf{x}_t, t, \mathbf{c}) \tag{23}$$

and independently sample K one-step transitions,

$$\mathbf{a}_t^{(k)} = \mathbf{x}_{t-1}^{(k)}, \qquad k = 1, \ldots, K. \tag{24}$$

For this experiment, we set $K = 8$. For each sampled action, we fix $\mathbf{x}_{t-1}^{(k)}$ and generate N_t independent suffix trajectories to $\mathbf{x}_0$. Let $r_{t,k,n}$ denote the terminal reward from the n-th suffix rollout of action k. We estimate the corresponding Q-value as

$$\widehat{Q}_{t,k} = \frac{1}{N_t} \sum_{n=1}^{N_t} r_{t,k,n}. \tag{25}$$

The raw action-wise Q variance is computed as the sample variance across the K Q-value estimates:

$$\widehat{\mathrm{Var}}_t^{\mathrm{raw}} = \frac{1}{K-1} \sum_{k=1}^{K} \left(\widehat{Q}_{t,k} - \overline{Q}_t \right)^2, \qquad \overline{Q}_t = \frac{1}{K} \sum_{k=1}^{K} \widehat{Q}_{t,k}. \tag{26}$$

Because each $\widehat{Q}_{t,k}$ is estimated from finitely many suffix rollouts, $\widehat{\mathrm{Var}}_t^{\mathrm{raw}}$ also contains Monte Carlo estimation noise. For $N_t > 1$, we compute the within-action reward variance as

$$\widehat{\sigma}_{t,k}^2 = \frac{1}{N_t - 1} \sum_{n=1}^{N_t} \left(r_{t,k,n} - \widehat{Q}_{t,k} \right)^2. \tag{27}$$

The estimated Monte Carlo noise in the action-wise Q variance is

$$\widehat{\mathrm{Var}}_t^{\mathrm{MC}} = \frac{1}{K} \sum_{k=1}^{K} \frac{\widehat{\sigma}_{t,k}^2}{N_t}. \tag{28}$$

We then obtain the corrected estimate

$$\widehat{\mathrm{Var}}_t = \max \left\{ \widehat{\mathrm{Var}}_t^{\mathrm{raw}} - \widehat{\mathrm{Var}}_t^{\mathrm{MC}}, 0 \right\}. \tag{29}$$

For each action, we generate $N_t = 64$ suffix rollouts. At the final denoising step, the sampled transition directly produces $\mathbf{x}_0$; therefore, we set $N_t = 1$ and $\widehat{\sigma}_{t,k}^2 = 0$. We repeat the measurement over 10 prompts and 5 random seeds and average Eq. 29 over the resulting 50 seed–prompt pairs.

This procedure generates 486,800 images for each pretrained base model. We evaluate the same generated samples using both the CLIP and 5-Mix rewards. Figure 4 shows the resulting Q-variance estimates, $\widehat{\mathrm{Var}}_t$, for SD1.4 and SDXL. Across all four settings, $\widehat{\mathrm{Var}}_t$ generally increases toward larger diffusion timesteps, indicating that, under the policy-induced action distribution, variations in early reverse-denoising decisions are associated with larger variations in the expected terminal reward.

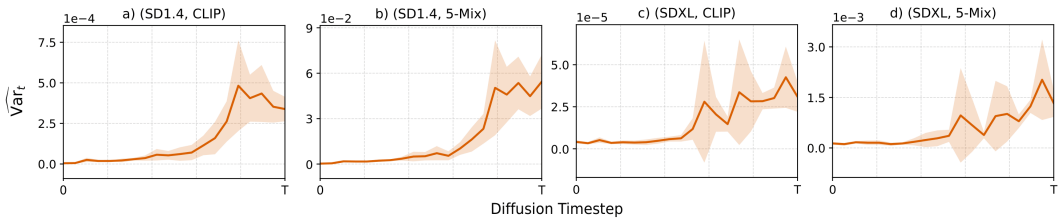


Figure 4: Estimated action-wise variance of the Q function across diffusion timesteps for pretrained SD1.4 and SDXL under CLIP and 5-Mix rewards. Each curve shows the corrected estimate $\widehat{\mathrm{Var}}_t$ averaged over 10 prompts and five random seeds, and the shaded region denotes the 95% confidence interval computed across the five seed-level means.

Action-Gradient Analysis of the Q Function. The action-wise Q variance in Eq. 22 reflects both the variation of the Q function over actions and the timestep-dependent spread of the policy-induced action distribution. To illustrate the possible contribution of the latter, let $\boldsymbol{\mu}_t = \mathbb{E}_\pi[\mathbf{a}_t \mid \mathbf{s}_t]$. A first-order Taylor approximation of Q^π around $\boldsymbol{\mu}_t$, together with the Gaussian transition covariance $\sigma_t^2\mathbf{I}$, gives

$$\mathrm{Var}_{\mathbf{a}_t}\left[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \sigma_t^2 \left\|\nabla_{\mathbf{a}} Q^\pi(\mathbf{s}_t, \boldsymbol{\mu}_t)\right\|_2^2 . \tag{30}$$

This approximation is used to illustrate that the Q-variance profile may be amplified by the timestep-dependent action spread and is not assumed to hold exactly over the full policy-induced action distribution. We therefore additionally measure a separate local sensitivity statistic,

$$S_t = \mathbb{E}_{\mathbf{s}_t,\, \mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)} \left[\frac{1}{d} \left\|\nabla_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)\right\|_2^2\right], \tag{31}$$

where d is the dimensionality of the latent action. This quantity averages the local change in expected terminal reward over actions sampled from the current policy and, unlike the Q variance, does not explicitly contain the multiplicative action-scale factor σ_t^2. It serves as a complementary local diagnostic rather than an exact decomposition of Eq. 22.

For each fixed reference state $\mathbf{s}_t$, we sample $K = 8$ native stochastic DDIM actions $\mathbf{a}_t^{(k)} \sim \pi(\cdot \mid \mathbf{s}_t)$. Holding each sampled action fixed, we generate $N = 32$ independent suffix trajectories and differentiate their terminal rewards with respect to the action:

$$\mathbf{g}_{t,k,n} = \nabla_{\mathbf{a}_t^{(k)}} r_{t,k,n}. \tag{32}$$

Because $\nabla_{\mathbf{a}_t} Q^\pi = \mathbb{E}_\pi[\nabla_{\mathbf{a}_t} r]$, we estimate the Q gradient at action k as

$$\overline{\mathbf{g}}_{t,k} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{g}_{t,k,n}. \tag{33}$$

The squared norm of this sample mean contains positive finite-suffix Monte Carlo bias. We therefore use the corrected estimator

$$\widehat{S}_{t,k} = \frac{1}{d}\left[\left\|\overline{\mathbf{g}}_{t,k}\right\|_2^2 - \frac{1}{N(N-1)} \sum_{n=1}^{N} \left\|\mathbf{g}_{t,k,n} - \overline{\mathbf{g}}_{t,k}\right\|_2^2\right]. \tag{34}$$

We average this estimate over the 8 sampled actions, 10 prompts, and 5 random seeds.

As shown in Fig. 5, the action-gradient generally increases toward large diffusion timesteps under both the CLIP and 5-Mix rewards. The same tendency is therefore observed in a local sensitivity measure that does not explicitly contain the multiplicative factor σ_t^2 and is corrected for finite-suffix gradient noise. This result indicates that the early-step increase in action-wise Q variance is not attributable solely to the larger stochastic transition scale; early reverse-denoising decisions also exhibit greater local variation of the expected terminal reward with respect to the action.

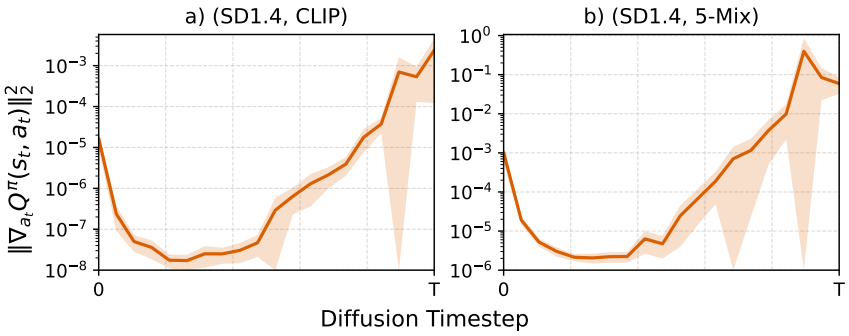


Figure 5: Squared action-gradient norm of the Q function across diffusion timesteps for pretrained SD1.4 under (a) CLIP and (b) 5-Mix rewards, where the shaded region denotes the 95% confidence interval computed across the five seed-level means. Large diffusion timesteps correspond to early stages of reverse denoising. This provides a local Q-sensitivity measure per unit action perturbation without an explicit multiplicative factor σ_t^2. Both rewards exhibit substantially larger values toward high diffusion timesteps, supporting that the Q-variance trend is not solely attributable to the stochastic transition scale.

Table 5: Training configurations for the underlying reward-optimization algorithms. A dash indicates that the corresponding parameter is not applicable.

Configuration	DDPO	PCPO / PCPO-base	SQDF	AlignProp
Base model	SD1.4 / SDXL	SD1.4 / SDXL	SD1.4	SD1.4
Total optimizer updates	400	400	2000	400
Batch per update	64	64	64	64
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	3×10^{-4}	3×10^{-4}	1×10^{-3}	1×10^{-4}
LoRA rank	4	4	4	4
Sampling steps	20	20	20	20
DDIM η / CFG scale	1.0/5.0	1.0/5.0	1.0/5.0	1.0/5.0
PCPO weight w_t	–	4.5	–	–
Reward discount γ	–	–	0.9	–
KL coefficient α	–	–	2.0	–
Loss scale	–	–	–	0.01

B Experiment Details

Training Setup and GTP-LoRA Configuration. We evaluate GTP-LoRA with two policy-gradient methods, DDPO and PCPO-base(a log-ratio-based DDPO variant), and two direct-gradient methods, SQDF and AlignProp. Table 5 summarizes the training configuration used for each underlying algorithm.

GTP-LoRA retains the objective and algorithm-specific hyperparameters of each underlying method in Table 5. It only replaces the shared LoRA adapter with timestep-specific LoRA experts, each using the same rank as the corresponding baseline. Before training, we sample a fixed reference set of 1,024 trajectories from the initial model and compute the timestep-wise denoiser-output gradient profile. The expert boundaries are determined from its cumulative magnitude and remain fixed throughout training. Unless otherwise specified, we use three experts ($K = 3$) in all experiments.

We train each method using three random seeds: 100, 300, and 500. All experiments are trained using four NVIDIA RTX PRO 6000 Blackwell GPUs. The per-step training time and one-time gradient-profile initialization cost for each method are summarized in Table 6.

Implementation of Timestep-wise LoRA Baselines. Because official implementations of TSM Zhuang et al. (2025) and TD-LoRA Zhao et al. (2026) were unavailable, we implemented

Table 6: Training-time comparison using four NVIDIA RTX PRO 6000 Blackwell GPUs. Avg. Time/Step denotes the average wall-clock time for one optimization step. Pre-initialization denotes the one-time cost of computing the reference gradient profile for GTP-LoRA.

Method	Avg. Time/Step (s)	Pre-initialization (s)
Policy-Gradient Optimization		
DDPO	17.2	–
PCPO-base	18.7	–
PCPO	18.7	–
DDPO + GTP-LoRA	24.3	185.0
PCPO-base + GTP-LoRA	25.9	192.5
Direct-Gradient Optimization		
AlignProp	15.1	–
AlignProp + GTP-LoRA	19.7	197.2
SQDF	8.2	–
SQDF + GTP-LoRA	9.5	181.4

both methods following their descriptions in the corresponding papers. Both implementations retain the DDPO configuration in Table 5 and modify only the LoRA architecture and routing mechanism.

For TSM, stage 1 trains four timestep-partitioned LoRA mixture-of-experts models with $n \in \{1, 2, 4, 8\}$ segments. In stage 2, all stage-1 models are frozen, and only the router is trained. Following the original method, the $n = 8$ model serves as the core expert and remains active without gating, while the models with $n \in \{1, 2, 4\}$ serve as context experts whose weights are determined by the router. The router produces the context-expert weights by combining a linear projection of the current latent $\mathbf{x}_t$ with a learned timestep lookup vector $\mathcal{E}(t) \in \mathbb{R}^{m-1}$, where $m = 4$ is the number of stage-1 models.

For TD-LoRA, the LoRA_B component is divided into three experts. A timestep embedding is passed through a two-layer MLP to produce expert logits, which are converted into mixture weights through a softmax. We use a 32-dimensional timestep embedding and a hidden dimension of 64 for the router.

The Uniform 3-Experts uniformly partitions the timestep axis into three non-overlapping intervals and assigns a separate vanilla LoRA expert to each interval; it can therefore be viewed as TSM stage 1 instantiated with $n = 3$.

Dataset and Evaluation. Following prior work Hu et al. (2025); Lee & Ye (2026), we use the *Animal Activity* prompt dataset, consisting of 135 prompts formed by combining 45 animal nouns (e.g., "cat," "dog," and "fish") with three activities: "washing dishes," "riding a bike," and "playing chess." We use 45 prompts for training and 90 prompts for evaluation. Full prompt lists are provided in Appendix H.

For evaluation, we generate 32 samples for each of the 90 evaluation prompts under each of the three training seeds, resulting in 2,880 samples per seed for each method. We measure prompt-wise diversity as the average pairwise ℓ_2 distance between the feature embeddings of samples generated for the same prompt, using DINOv2 Oquab et al. (2023) and DreamSim Fu et al. (2023) features.

C Derivation of Timestep-wise Gradients

Policy-Gradient. The policy-gradient objective is

$$J_{\mathrm{PG}}(\theta) = \mathbb{E}_{\mathbf{c},\, \tau \sim p_\theta(\tau|\mathbf{c})} \left[r(\mathbf{x}_0, \mathbf{c}) \right]. \tag{35}$$

Using the likelihood-ratio gradient,

$$\nabla_\theta J_{\mathrm{PG}}(\theta) = \mathbb{E}\left[r(\mathbf{x}_0, \mathbf{c}) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, t, \mathbf{c} \right) \right]. \tag{36}$$

Under the stochastic DDIM transition, the policy is Gaussian:

$$\pi_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, t, \mathbf{c}\right) = \mathcal{N}\left(\boldsymbol{\mu}_{\theta,t}, \sigma_t^2 \mathbf{I}\right), \tag{37}$$

where

$$\boldsymbol{\mu}_{\theta,t} = A_t \mathbf{x}_t + c_t \epsilon_\theta\left(\mathbf{x}_t, t, \mathbf{c}\right), \qquad A_t = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}. \tag{38}$$

Since the sampled transition can be written as

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\theta,t} + \sigma_t \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{39}$$

The gradient with respect to the transition mean is

$$\nabla_{\boldsymbol{\mu}_{\theta,t}} \log \pi_\theta = \frac{\mathbf{x}_{t-1} - \boldsymbol{\mu}_{\theta,t}}{\sigma_t^2} = \frac{\boldsymbol{\epsilon}_t}{\sigma_t} \tag{40}$$

Let

$$\mathbf{J}_{\theta,t} = \frac{\partial \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})}{\partial \theta}. \tag{41}$$

Because $\partial \boldsymbol{\mu}_{\theta,t} / \partial \epsilon_\theta = c_t \mathbf{I}$, we obtain

$$\nabla_\theta \log \pi_\theta = \mathbf{J}_{\theta,t}^\top \frac{c_t}{\sigma_t} \boldsymbol{\epsilon}_t. \tag{42}$$

Substituting this expression into Eq. 36 gives

$$\nabla_\theta J_{\mathrm{PG}}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{J}_{\theta,t}^\top r(\mathbf{x}_0, \mathbf{c}) \frac{c_t}{\sigma_t} \boldsymbol{\epsilon}_t\right]. \tag{43}$$

Thus, the timestep-wise gradient can be written as

$$\nabla_\theta^{(t)} J_{\mathrm{PG}}(\theta) = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top r(\mathbf{x}_0, \mathbf{c}) \frac{c_t}{\sigma_t} \boldsymbol{\epsilon}_t\right] = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t^{\mathrm{PG}}\right], \tag{44}$$

where the timestep-wise denoiser-output gradient is

$$\boldsymbol{\delta}_t^{\mathrm{PG}} = r(\mathbf{x}_0, \mathbf{c}) \frac{c_t}{\sigma_t} \boldsymbol{\epsilon}_t \tag{45}$$

Direct-Gradient. Direct-gradient methods reparameterize the final sample as a differentiable function of the initial noise and transition noises. The objective is

$$J_{\mathrm{DG}}(\theta) = \mathbb{E}_{\mathbf{c},\, \mathbf{x}_T,\, \boldsymbol{\epsilon}_{1:T}}\left[r\left(\mathbf{x}_0(\theta), \mathbf{c}\right)\right]. \tag{46}$$

Its gradient is

$$\nabla_\theta J_{\mathrm{DG}}(\theta) = \mathbb{E}\left[\left(\frac{\partial \mathbf{x}_0}{\partial \theta}\right)^\top \nabla_{\mathbf{x}_0} r(\mathbf{x}_0, \mathbf{c})\right]. \tag{47}$$

Let each reverse transition be written as

$$\mathbf{x}_{t-1} = f_t(\mathbf{x}_t; \theta, \mathbf{c}, \boldsymbol{\epsilon}_t) = A_t \mathbf{x}_t + c_t \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) + \sigma_t \boldsymbol{\epsilon}_t. \tag{48}$$

The parameter Jacobian of the final sample can then be decomposed as

$$\frac{\partial \mathbf{x}_0}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_{t-1}} \frac{\partial f_t}{\partial \theta}. \tag{49}$$

The direct parameter dependence of the transition is

$$\frac{\partial f_t}{\partial \theta} = c_t \mathbf{J}_{\theta,t}. \tag{50}$$

To express the remaining trajectory Jacobian, define

$$\mathbf{J}_{x,s} = \frac{\partial \epsilon_\theta(\mathbf{x}_s, s, \mathbf{c})}{\partial \mathbf{x}_s}. \tag{51}$$

The Jacobian of one reverse transition is

$$\frac{\partial f_s}{\partial \mathbf{x}_s} = A_s \mathbf{I} + c_s \mathbf{J}_{x,s}. \tag{52}$$

Since $\mathbf{x}_0 = (f_1 \circ \cdots \circ f_{t-1})(\mathbf{x}_{t-1})$, applying the chain rule along the remaining reverse transitions gives

$$\frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_{t-1}} = \prod_{s=1}^{t-1} \left(A_s \mathbf{I} + c_s \mathbf{J}_{x,s} \right), \tag{53}$$

where the factors are ordered with increasing s from left to right. Substituting Eqs. 49 and 53 into Eq. 47 yields

$$\nabla_\theta J_{\mathrm{DG}}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t^{\mathrm{DG}} \right]. \tag{54}$$

Thus, the timestep-wise gradient can be written as

$$\nabla_\theta^{(t)} J_{\mathrm{DG}}(\theta) = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t^{\mathrm{DG}} \right], \tag{55}$$

where

$$\boldsymbol{\delta}_t^{\mathrm{DG}} = c_t \left\{ \prod_{s=1}^{t-1} \left(A_s \mathbf{I} + c_s \mathbf{J}_{x,s} \right) \right\}^\top \nabla_{\mathbf{x}_0} r(\mathbf{x}_0, \mathbf{c}). \tag{56}$$

If optimization is implemented by minimizing $\mathcal{L} = -J$, the corresponding output gradients have the opposite sign, while their norms remain unchanged. As discussed in the main text, the expressions above describe canonical gradient estimators for the two optimization families, while practical algorithms may introduce method-specific modifications. Detailed modifications to the algorithms used in our analysis (DDPO and SQDF) are provided in Appendix D.

D Analysis of Scheduler Terms and Timestep-wise Gradient Profiles

The derivations in Appendix C decompose the timestep-wise gradients for policy-gradient and direct-gradient optimization as

$$\nabla_\theta^{(t)} J(\theta) = \mathbb{E}\left[\mathbf{J}_{\theta,t}^\top \boldsymbol{\delta}_t \right]. \tag{57}$$

This decomposition distinguishes the upstream denoiser-output gradient $\boldsymbol{\delta}_t$, which contains objective- and scheduler-specific timestep dependence, from its subsequent mapping into parameter space through the local parameter Jacobian $\mathbf{J}_{\theta,t}$.

In our experiments, these gradients are instantiated by DDPO for policy-gradient optimization and SQDF for direct-gradient optimization. Below, we briefly derive their practical denoiser-output gradients and examine (i) their empirical relationship with the explicit local scheduler terms and (ii) whether the resulting timestep-wise concentration remains visible after mapping into parameter space.

Policy-Gradient. As derived in Eq. 45, the canonical policy-gradient denoiser-output gradient is

$$\boldsymbol{\delta}_t^{\mathrm{PG}} = r(\mathbf{x}_0, \mathbf{c}) \frac{c_t}{\sigma_t} \boldsymbol{\epsilon}_t. \tag{58}$$

Thus, c_t/σ_t appears as an explicit deterministic scheduler factor, in addition to the sampled terminal reward and transition noise.

DDPO implements this estimator using a clipped importance-weighted surrogate. The reverse transition is represented as the Gaussian policy

$$\pi_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\theta,t}, \sigma_t^2 \mathbf{I} \right), \tag{59}$$

where

$$\boldsymbol{\mu}_{\theta,t} = A_t \mathbf{x}_t + c_t \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}). \tag{60}$$

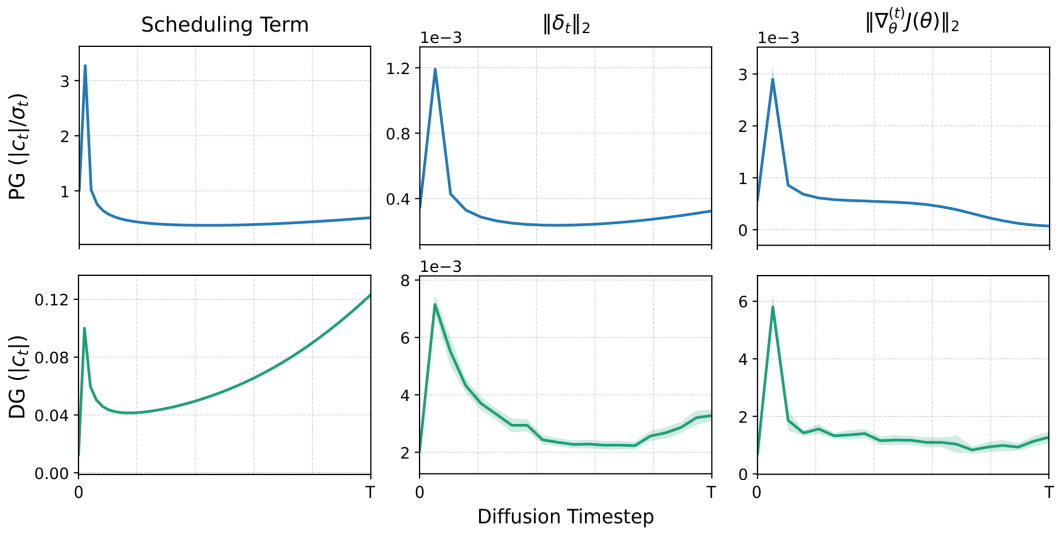


Figure 6: Empirical relationship between local reverse-process scheduler terms and timestep-wise gradients for policy-gradient optimization with DDPO (top) and direct-gradient optimization with SQDF (bottom) while training SD1.4 with CLIP reward. The left column shows $|c_t|/\sigma_t$ for DDPO and $|c_t|$, the local coefficient in the SQDF reward-gradient path. The middle column shows the corresponding algorithm-specific denoiser-output gradient norm $\|\boldsymbol{\delta}_t\|_2$. The right column shows $\|\nabla_\theta^{(t)} J(\theta)\|_2$. Small diffusion timesteps correspond to late stages of reverse denoising. In both cases, the dominant late-step concentration remains visible after mapping into parameter space.

For a trajectory sampled under θ_{old}, the transition-wise importance ratio is

$$\rho_t(\theta) = \frac{\pi_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})}{\pi_{\theta_{\text{old}}}(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})}. \tag{61}$$

Letting $\widehat{A}$ denote the normalized trajectory-level advantage, DDPO optimizes

$$J_{\text{DDPO}}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \min\Big(\rho_t(\theta)\widehat{A}, \; \text{clip}\left(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\widehat{A}\Big)\right]. \tag{62}$$

When the unclipped branch is active, differentiating the Gaussian log probability with respect to the denoiser output gives

$$\begin{aligned} \boldsymbol{\delta}_t^{\text{DDPO}} &= \rho_t(\theta)\widehat{A}\frac{c_t}{\sigma_t^2}\left(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{\theta,t}\right) \\ &= \rho_t(\theta)\widehat{A}\frac{c_t}{\sigma_t}\widetilde{\boldsymbol{\epsilon}}_{\theta,t}, \end{aligned} \tag{63}$$

where

$$\widetilde{\boldsymbol{\epsilon}}_{\theta,t} = \frac{\mathbf{x}_{t-1} - \boldsymbol{\mu}_{\theta,t}}{\sigma_t} \tag{64}$$

is the normalized transition residual under the current policy. When clipping selects a saturated clipped branch, the corresponding transition produces no gradient. At the data-collection policy, $\rho_t(\theta) = 1$ and $\widetilde{\boldsymbol{\epsilon}}_{\theta,t} = \boldsymbol{\epsilon}_t$, reducing Eq. 63 to the canonical form in Eq. 58, with the terminal reward replaced by the normalized advantage.

Therefore, the practical DDPO gradient also depends on the advantage, importance ratio, transition residual, and clipping, while c_t/σ_t remains its explicit deterministic scheduler factor. As shown in the left column of Fig. 6, the denoiser-output gradient closely follows its sharp late-step peak, indicating that c_t/σ_t is a major source of the observed policy-gradient imbalance.

Direct-Gradient. For canonical full-trajectory direct-gradient optimization, Eq. 56 gives

$$\boldsymbol{\delta}_t^{\mathrm{DG}} = c_t \mathbf{B}_t^\top \nabla_{\mathbf{x}_0} r(\mathbf{x}_0, \mathbf{c}), \qquad \mathbf{B}_t = \prod_{s=1}^{t-1} \left(A_s \mathbf{I} + c_s \mathbf{J}_{x,s}\right). \tag{65}$$

Here, c_t provides explicit local scheduler-dependent scaling, while $\mathbf{B}_t$ propagates the terminal reward gradient through the remaining reverse transitions.

SQDF instantiates direct-gradient optimization using a one-step soft-Q surrogate. Rather than backpropagating through the complete remaining trajectory, it samples a timestep t, treats the corresponding state $\mathbf{x}_t$ as fixed, and differentiates a single reverse transition:

$$\mathbf{x}_{t-1} = A_t \mathbf{x}_t + c_t \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) + \sigma_t \boldsymbol{\epsilon}_t. \tag{66}$$

A frozen soft-Q estimator maps the resulting action to a predicted clean sample,

$$\widehat{\mathbf{x}}_0 = h_\psi(\mathbf{x}_{t-1}, t-1, \mathbf{c}), \tag{67}$$

where ψ is fixed during diffusion-model fine-tuning. SQDF then optimizes the sampled-timestep objective

$$\begin{aligned} J_{\mathrm{SQDF}}(\theta) = \mathbb{E}_{t\sim q(t)}\Big[&\gamma_t r(\widehat{\mathbf{x}}_0, \mathbf{c}) \\ &- \alpha D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid \mathbf{x}_t, \mathbf{c}) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid \mathbf{x}_t, \mathbf{c})\right)\Big], \end{aligned} \tag{68}$$

where γ_t denotes the timestep-dependent discount used by SQDF and π_{ref} is the pretrained reference policy.

Defining the Jacobian of the frozen soft-Q estimator as

$$\mathbf{H}_{\psi,t-1} = \frac{\partial h_\psi(\mathbf{x}_{t-1}, t-1, \mathbf{c})}{\partial \mathbf{x}_{t-1}}, \tag{69}$$

the SQDF denoiser-output gradient becomes

$$\begin{aligned} \boldsymbol{\delta}_t^{\mathrm{SQDF}} =& \gamma_t c_t \mathbf{H}_{\psi,t-1}^\top \nabla_{\widehat{\mathbf{x}}_0} r(\widehat{\mathbf{x}}_0, \mathbf{c}) \\ &- \alpha \frac{\partial}{\partial \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})} D_{\mathrm{KL}}\left(\pi_\theta \| \pi_{\mathrm{ref}}\right). \end{aligned} \tag{70}$$

Because the current and reference reverse policies use the same covariance, their local KL divergence can be written as

$$\begin{aligned} D_{\mathrm{KL}}\left(\pi_\theta \| \pi_{\mathrm{ref}}\right) &= \frac{1}{2\sigma_t^2} \left\| \boldsymbol{\mu}_{\theta,t} - \boldsymbol{\mu}_{\mathrm{ref},t} \right\|_2^2 \\ &= \frac{c_t^2}{2\sigma_t^2} \left\| \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon_{\mathrm{ref}}(\mathbf{x}_t, t, \mathbf{c}) \right\|_2^2. \end{aligned} \tag{71}$$

The resulting output gradient is therefore

$$\begin{aligned} \boldsymbol{\delta}_t^{\mathrm{SQDF}} =& \gamma_t c_t \mathbf{H}_{\psi,t-1}^\top \nabla_{\widehat{\mathbf{x}}_0} r(\widehat{\mathbf{x}}_0, \mathbf{c}) \\ &- \alpha \frac{c_t^2}{\sigma_t^2} \left(\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon_{\mathrm{ref}}(\mathbf{x}_t, t, \mathbf{c})\right). \end{aligned} \tag{72}$$

Thus, c_t remains the explicit local scheduler factor in the SQDF reward-gradient path, but it does not fully determine the practical gradient. The reward component is additionally modulated by the discount, the soft-Q-estimator Jacobian, and the clean-sample reward gradient, while the KL component introduces a separate scheduler-dependent scale c_t^2/σ_t^2.

As shown in the right column of Fig. 6, the measured SQDF denoiser-output gradient broadly follows the temporal profile of c_t, including its pronounced late-step peak. The remaining differences arise from the discount, one-step soft-Q estimator, reward gradient, and KL regularization, showing that c_t provides important local scaling but does not fully determine the SQDF gradient profile.

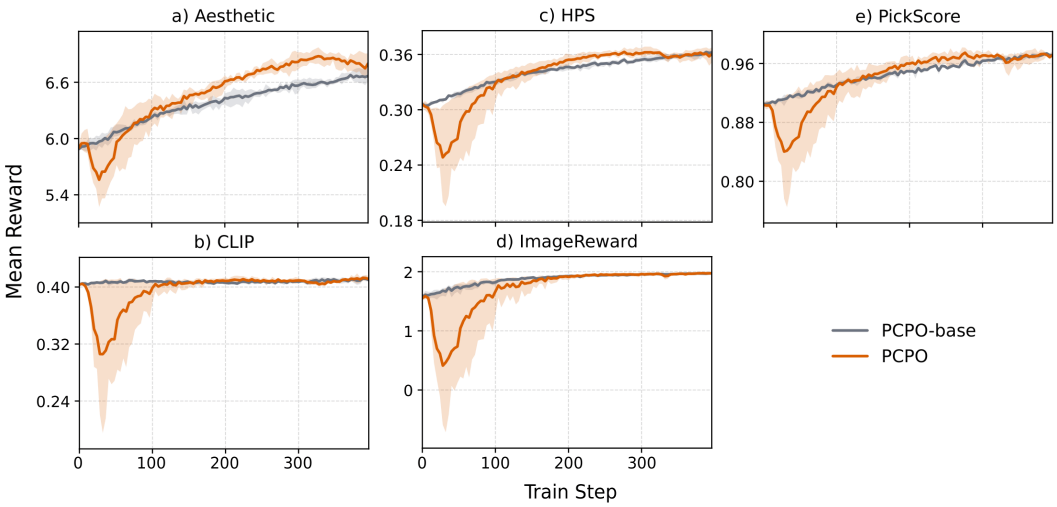


Figure 7: Training of PCPO-base and PCPO on SDXL under the 5-Mix reward. PCPO exhibits pronounced early-stage decreases across all five rewards before subsequently recovering, whereas PCPO-base shows smoother optimization trajectories. Curves show the mean over random seeds, with shaded regions indicating the min–max range

Mapping into Parameter Space. As shown in Eq. 57, the final timestep-wise gradient is obtained by mapping the denoiser-output gradient $\boldsymbol{\delta}_t$ into parameter space through the local parameter Jacobian $\mathbf{J}_{\theta,t}$. This mapping can amplify or attenuate individual timestep signals, and the resulting parameter-gradient profile therefore need not exactly reproduce the denoiser-output gradient profile.

Nevertheless, the bottom row of Fig. 6 exhibits a temporal profile closely aligned with the corresponding denoiser-output gradient for both DDPO and SQDF, including the dominant concentration near late reverse-denoising steps. This indicates that the algorithm-specific upstream gradient is the primary source of the observed timestep-wise imbalance, while the local parameter Jacobian mainly reshapes its relative magnitude. The persistence of this structure after mapping into parameter space motivates the use of the denoiser-output gradient profile for determining the gradient-balanced expert boundaries in GTP-LoRA.

E Analysis of Limitations of Rescheduling

As discussed in the main text, PCPO Lee & Ye (2026) addresses the timestep-wise imbalance in policy-gradient optimization by rescheduling σ_t to approximately equalize the local scheduler factor $|c_t|/\sigma_t$ across timesteps. Because c_t is coupled to σ_t through the DDIM transition, this procedure modifies not only the gradient scaling but also the underlying reverse-process dynamics. In this section, we examine (i) the potential optimization consequences of PCPO rescheduling in policy-gradient optimization and (ii) why an analogous correction does not directly extend to direct-gradient optimization.

Policy-Gradient Rescheduling and Training Stability. Figure 7 compares the training dynamics of PCPO-base and PCPO under the SDXL 5-Mix setting. PCPO exhibits pronounced early-stage decreases across all five rewards before gradually recovering, whereas PCPO-base follows substantially smoother optimization trajectories. Since the only modification from PCPO-base to PCPO in this comparison is the rescheduling of the reverse-process coefficients, the observed behavior suggests that the modified sampling dynamics may contribute to the initial training instability.

To avoid ambiguity, we denote the positive coefficient used in PCPO by C_t, which has the opposite sign to the signed reverse-transition coefficient c_t used throughout our main text and supplementary material:

$$C_t = -c_t = |c_t|. \tag{73}$$

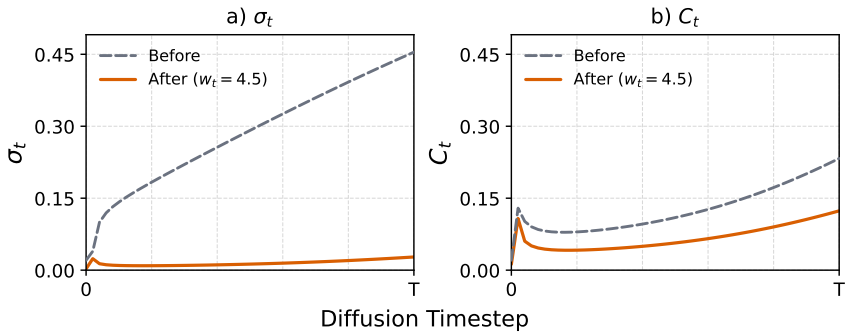


Figure 8: Reverse-process coefficients before and after PCPO rescheduling with the target ratio $w_t = C_t/\sigma_t = 4.5$. Rescheduling substantially reduces σ_t over most timesteps and also shifts the coupled denoiser coefficient C_t away from the original DDIM schedule.

Figure 8 compares the original DDIM coefficients with those obtained by enforcing $w_t = C_t/\sigma_t = 4.5$. The rescheduled σ_t is substantially reduced over most of the trajectory, while the coupled coefficient C_t also deviates considerably from its original schedule. These changes simultaneously alter the stochasticity of each transition and the magnitude of the coefficient applied to the denoiser prediction. We therefore hypothesize that this substantial departure from the pretrained reverse-process schedule contributes to the early-stage instability observed in Fig. 7.

Limitation for Direct-Gradient Optimization. The most direct analogue of PCPO for direct-gradient optimization would be to reschedule σ_t such that the local coefficient c_t remains constant across timesteps. This follows from the fact that c_t depends on σ_t in the DDIM transition. However, unlike equalizing the ratio c_t/σ_t, a constant c_t is not generally attainable under the valid range of DDIM stochasticity.

Let $\tau_0 < \tau_1 < \cdots < \tau_S$ denote the timesteps selected for an S-step sampler, such that reverse sampling applies the transitions $\mathbf{x}_{\tau_i} \to \mathbf{x}_{\tau_{i-1}}$ for $i = 1, \ldots, S$. We define

$$a_i = \bar{\alpha}_{\tau_i}, \qquad a_{i-1} = \bar{\alpha}_{\tau_{i-1}}, \tag{74}$$

where a_0 denotes the terminal cumulative alpha used by the scheduler for the final transition. Each selected DDIM transition can be written as

$$\mathbf{x}_{\tau_{i-1}} = \sqrt{\frac{a_{i-1}}{a_i}}\mathbf{x}_{\tau_i} + c_i\epsilon_\theta(\mathbf{x}_{\tau_i}, \tau_i, \mathbf{c}) + \sigma_i\boldsymbol{\epsilon}_i, \tag{75}$$

where

$$c_i(\sigma_i) = \sqrt{1 - a_{i-1} - \sigma_i^2} - \sqrt{\frac{a_{i-1}}{a_i}}\sqrt{1 - a_i}. \tag{76}$$

For notational convenience, define

$$B_i = \sqrt{\frac{a_{i-1}}{a_i}}\sqrt{1 - a_i}, \qquad D_i = \sqrt{1 - a_{i-1}}. \tag{77}$$

Equation 76 then becomes

$$c_i(\sigma_i) = \sqrt{D_i^2 - \sigma_i^2} - B_i. \tag{78}$$

A valid real-valued DDIM transition requires $0 \le \sigma_i \le D_i$. Consequently,

$$0 \le \sqrt{D_i^2 - \sigma_i^2} \le D_i, \tag{79}$$

and the set of coefficients attainable at transition i is exactly

$$c_i \in \mathcal{C}_i := [-B_i,\, D_i - B_i]\,. \tag{80}$$

Therefore, a constant coefficient $c^\star$ can be achieved over the complete sampling trajectory if and only if the feasible intervals have a nonempty intersection:

$$c^\star \in \bigcap_{i=1}^{S} \mathcal{C}_i. \tag{81}$$

Equivalently, an exact solution exists if and only if

$$\boxed{\max_{1\le i\le S}(-B_i) \le \min_{1\le i\le S}(D_i - B_i)}. \tag{82}$$

When Eq. 82 holds, any

$$c^\star \in \Big[\max_i(-B_i),\ \min_i(D_i - B_i)\Big] \tag{83}$$

defines a valid exact rescheduling, with the required stochasticity given by

$$\sigma_i^\star = \sqrt{D_i^2 - (c^\star + B_i)^2}. \tag{84}$$

Conversely, if the lower endpoint in Eq. 82 exceeds the upper endpoint, no valid choice of $\{\sigma_i\}_{i=1}^S$ can make c_i constant over the complete trajectory.

Since $a_{i-1} > a_i$ for a nondegenerate reverse transition, $B_i > D_i$ and hence $c_i < 0$. The same feasibility condition may therefore be expressed in terms of the magnitude $w^\star = -c^\star > 0$ as

$$\boxed{\max_{1\le i\le S}(B_i - D_i) \le \min_{1\le i\le S} B_i}. \tag{85}$$

For the canonical terminal convention $a_0 = 1$, the final transition has $D_1 = 0$, so its feasible interval collapses to $\mathcal{C}_1 = \{-B_1\}$. The common coefficient is therefore fixed to $c^\star = -B_1$, and exact uniformization is possible only if this value also belongs to every remaining interval $\mathcal{C}_i$. For the SD1.4 DDIM scheduler with 1,000 training timesteps, a scaled-linear β schedule, and leading timestep spacing, the feasibility condition is satisfied only when the sampler uses at least 167 denoising steps. Consequently, the sampling step configuration used in Table 5 admits no valid constant-c_t solution across the complete trajectory. Moreover, 167 steps is substantially denser than the sampling configurations typically used in reward-based diffusion fine-tuning because of the high cost of trajectory generation.

This analysis shows that constant-c_t rescheduling is not generally available as a practical direct-gradient counterpart to PCPO. Even when the feasibility condition holds, it controls only the local coefficient c_t, whereas the complete direct-gradient signal also depends on the propagated reward gradient and algorithm-specific gradient estimation. Consequently, PCPO-style rescheduling cannot, in general, equalize the complete timestep-wise gradient in direct-gradient optimization.

F Additional Ablation Studies

F.1 Analysis of Fixed Boundaries

GTP-LoRA determines its expert boundaries from the reference denoiser-output gradient profile measured during the pre-initialization stage and keeps them fixed throughout subsequent fine-tuning. We analyze this behavior for GTP-LoRA trained with the DDPO objective on SD1.4 using the CLIP reward across three random seeds. The two pre-initialized boundaries, averaged across seeds, are located at 0.391 and 0.825 when the timestep axis is normalized to $[0, 1]$. Although the gradient profile may evolve during training, Fig. 9 shows that the proportion of total gradient mass assigned to each expert remains relatively stable, indicating that the initial boundaries continue to provide a balanced partition of the upstream signal.

To further examine whether continuously adapting the boundaries provides additional benefits, we evaluate an adaptive variant of GTP-LoRA. The boundaries are initialized using the same reference profile but are subsequently updated after every optimizer step. Given the current continuous boundaries, the router computes the gradient mass G_k assigned to each expert and updates the boundaries using a separate dual Adam optimizer to reduce deviations from the mean mass. These boundary

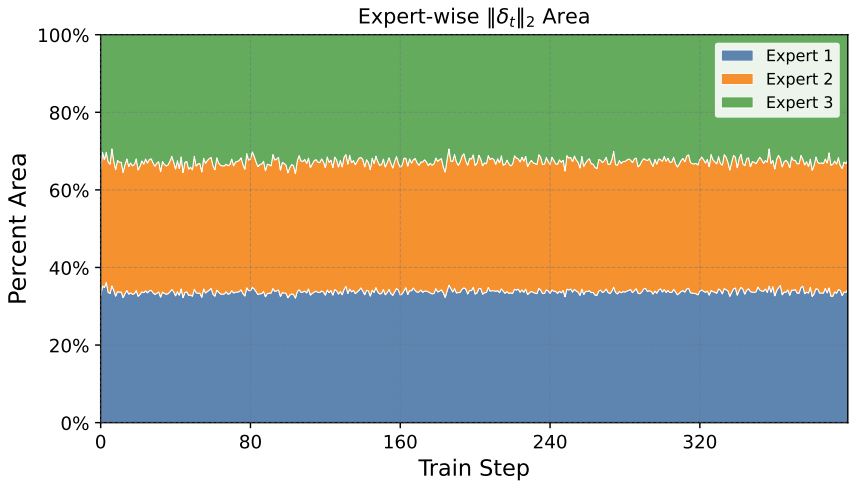


Figure 9: Evolution of the denoiser-output gradient mass assigned to each GTP-LoRA expert during DDPO training of SD1.4 with the CLIP reward, averaged over three random seeds. For each optimization step, the overlap-weighted gradient mass of each expert is normalized by the total mass across experts. The expert-wise proportions remain relatively stable throughout training, supporting the use of fixed boundaries determined from the initial gradient profile.

variables are optimized independently of the expert LoRA parameters and do not receive gradients directly from the DDPO objective. The adaptive variant uses a dual-Adam learning rate of 0.05 with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The expert LoRA parameters are trained using the same DDPO optimizer as the fixed-boundary model, while the boundaries are updated separately at each of the 400 optimization steps.

As shown in Table 7, adaptive GTP-LoRA achieves a CLIP score nearly identical to that of GTP-LoRA, but substantially decreases both diversity metrics. We hypothesize that continuously moving the boundaries creates a nonstationary assignment of timesteps to experts: as timesteps migrate between adjacent segments, each expert is repeatedly exposed to a changing denoising regime. This moving target can disrupt stable temporal specialization and allow reward-dominant regions to influence multiple experts over the course of training.

Table 7: Comparison of adaptive and fixed expert boundaries under the SD1.4 DDPO setting with CLIP reward. Adaptive boundary updates provide a negligible improvement in CLIP alignment but substantially reduce sample diversity.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
Adaptive GTP-LoRA	$\mathbf{0.421 \pm 0.005}$	$\underline{0.351 \pm 0.015}$	$\underline{0.353 \pm 0.013}$
GTP-LoRA	$\underline{0.420 \pm 0.003}$	$\mathbf{0.413 \pm 0.019}$	$\mathbf{0.384 \pm 0.014}$

F.2 Effect of Gradient-Balanced Segmentation.

To verify that the gains of GTP-LoRA do not arise merely from increased adapter capacity or computation, we compare GTP-LoRA with timestep-wise LoRA expert approaches originally developed for standard diffusion training Zhuang et al. (2025); Zhao et al. (2026), adapting their expert-routing mechanisms to the same DDPO reward-fine-tuning setting. We also introduce Uniform 3-Experts, a capacity-matched baseline that divides the denoising trajectory into three equal-length segments and assigns one expert to each segment.

As shown in Table 8, GTP-LoRA outperforms Uniform 3-Experts across all three metrics, indicating that its gains do not arise solely from separating adapter parameters across timesteps. GTP-LoRA also shows the strongest overall balance between reward, diversity, and inference-time computa-

Table 8: Comparison with existing timestep-wise LoRA Mixture-of-Experts methods. Cost denotes the relative adapter computation normalized by vanilla LoRA.

Model	Cost ↓	CLIP ↑	DINOv2 ↑	DreamSim ↑
Vanilla LoRA	$\times 1.0$	0.404 ± 0.002	0.358 ± 0.005	0.366 ± 0.004
TD-LoRA	$\times 2.0$	0.406 ± 0.002	$\underline{0.392 \pm 0.003}$	$\underline{0.383 \pm 0.003}$
Uniform Experts	$\times 1.0$	0.417 ± 0.003	0.380 ± 0.021	0.363 ± 0.019
TSM	$\times 4.0$	$\mathbf{0.424 \pm 0.002}$	0.343 ± 0.017	0.333 ± 0.017
GTP-LoRA	$\times 1.1$	$\underline{0.420 \pm 0.003}$	$\mathbf{0.413 \pm 0.019}$	$\mathbf{0.384 \pm 0.014}$

tional cost among the evaluated expert-based approaches. Although TSM attains a slightly higher CLIP score, it requires approximately $4\times$ the adapter computation. TD-LoRA likewise incurs $2\times$ the computation but underperforms GTP-LoRA across all metrics. Overall, GTP-LoRA provides the best cost–performance trade-off, achieving competitive reward and the strongest diversity with less than $1.1\times$ the adapter computation of vanilla LoRA.

F.3 Effect of the Number of Denoising Steps

To examine whether the effectiveness of GTP-LoRA depends on the number of denoising steps, we conduct additional experiments with 50 denoising steps instead of the 20 used in the main experiments. We evaluate three configurations using SD1.4 with the CLIP reward: DDPO, DDPO + GTP-LoRA, and PCPO. This comparison assesses the effect of gradient-balanced segmentation under a finer discretization of the denoising trajectory.

Table 9 shows that GTP-LoRA remains effective when the number of denoising steps is increased from 20 to 50. Compared with DDPO, GTP-LoRA improves reward performance while substantially alleviating the degradation in sample diversity. It also outperforms PCPO across all three metrics, suggesting that the benefits of gradient-balanced segmentation persist under a finer discretization of the denoising trajectory.

Table 9: Comparison of baselines trained with 20 and 50 denoising steps under CLIP reward.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
20 denoising steps			
SD1.4	0.352 ± 0.001	0.454 ± 0.002	0.451 ± 0.002
PCPO	$\underline{0.414 \pm 0.002}$	$\underline{0.371 \pm 0.009}$	$\underline{0.373 \pm 0.009}$
DDPO	0.404 ± 0.002	0.358 ± 0.005	0.366 ± 0.004
+GTP-LoRA	$\mathbf{0.420 \pm 0.003}$	$\mathbf{0.413 \pm 0.019}$	$\mathbf{0.384 \pm 0.014}$
50 denoising steps			
SD1.4	0.354 ± 0.001	0.446 ± 0.003	0.446 ± 0.002
PCPO	$\underline{0.419 \pm 0.001}$	$\underline{0.381 \pm 0.011}$	$\underline{0.376 \pm 0.010}$
DDPO	0.399 ± 0.004	0.373 ± 0.006	0.377 ± 0.004
+GTP-LoRA	$\mathbf{0.424 \pm 0.004}$	$\mathbf{0.433 \pm 0.014}$	$\mathbf{0.413 \pm 0.015}$

G More Qualitative Results



Figure 10: Qualitative comparison of PCPO-base, PCPO, and PCPO-base + GTP-LoRA trained on SD1.4 using the CLIP reward. For each prompt, corresponding samples across methods are generated using the same random seed and sample index.

H Prompt Lists

Table 10: Training and test prompt split for the Animal Activity dataset. Prompts follow the template "a [animal] [activity]." Each animal appears with one activity in the training set and with the other two activities in the test set, yielding 45 training and 90 test prompts.

Training Activity	Animals	Test Activities	# Train	# Test
Washing dishes	cat, dog, horse, monkey, rabbit, zebra, spider, bird, sheep, deer, cow, goat, lion, tiger, bear	Riding a bike; playing chess	15	30
Riding a bike	raccoon, fox, wolf, lizard, beetle, ant, butterfly, fish, shark, whale, dolphin, squirrel, mouse, rat, snake	Washing dishes; playing chess	15	30
Playing chess	turtle, frog, chicken, duck, goose, bee, pig, turkey, fly, llama, camel, bat, gorilla, hedgehog, kangaroo	Washing dishes; riding a bike	15	30
Total	45 animals	Three activities	45	90

I Detailed Quantitative Results

In this section, we provide detailed quantitative results, including the corresponding standard deviations over three random seeds, for the main experiments and ablation studies.

Table 11: Comparison of CLIP alignment and diversity for SD1.4 trained with policy-gradient methods using CLIP reward.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
SD1.4	0.352 ± 0.001	0.454 ± 0.002	0.451 ± 0.002
PCPO	0.414 ± 0.002	0.371 ± 0.009	0.373 ± 0.009
PCPO-base	0.406 ± 0.004	0.369 ± 0.002	$\underline{0.375 \pm 0.004}$
+GTP-LoRA	$\mathbf{0.425 \pm 0.002}$	$\underline{0.373 \pm 0.016}$	0.353 ± 0.011
DDPO	0.404 ± 0.002	0.358 ± 0.005	0.366 ± 0.004
+GTP-LoRA	$\underline{0.420 \pm 0.003}$	$\mathbf{0.413 \pm 0.019}$	$\mathbf{0.384 \pm 0.014}$

Table 12: Comparison of CLIP alignment and diversity for SD1.4 trained with the direct-gradient methods using CLIP reward. † indicates training for 2000 optimization steps.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
SQDF†	0.399 ± 0.003	0.315 ± 0.017	0.319 ± 0.017
+GTP-LoRA†	$\underline{0.407 \pm 0.002}$	0.329 ± 0.013	0.332 ± 0.018
AlignProp	0.398 ± 0.008	$\underline{0.345 \pm 0.005}$	$\mathbf{0.375 \pm 0.013}$
+GTP-LoRA	$\mathbf{0.421 \pm 0.002}$	$\mathbf{0.376 \pm 0.015}$	$\underline{0.356 \pm 0.008}$

Table 13: Comparison of reward performance for SDXL trained using 5-Mix reward.

Model	CLIP ↑	HPS ↑	PickScore ↑	ImageReward ↑	Aesthetic ↑
SDXL	0.399 ± 0.000	0.304 ± 0.001	0.899 ± 0.000	1.544 ± 0.010	5.842 ± 0.004
PCPO	$\underline{0.402 \pm 0.003}$	0.354 ± 0.006	0.963 ± 0.002	1.950 ± 0.003	$\mathbf{6.808 \pm 0.084}$
PCPO-base	0.401 ± 0.003	$\underline{0.356 \pm 0.003}$	$\underline{0.964 \pm 0.002}$	$\underline{1.956 \pm 0.009}$	6.630 ± 0.083
DDPO	0.391 ± 0.001	0.339 ± 0.002	0.941 ± 0.001	1.889 ± 0.001	6.521 ± 0.023
+GTP-LoRA	$\mathbf{0.404 \pm 0.002}$	$\mathbf{0.365 \pm 0.003}$	$\mathbf{0.974 \pm 0.003}$	$\mathbf{1.957 \pm 0.013}$	$\underline{6.749 \pm 0.063}$

Table 14: Ablation studies on the number of LoRA experts and boundary-timestep routing.

Model	CLIP ↑	DINOv2 ↑	DreamSim ↑
Number of Experts			
2-Experts	0.416 ± 0.002	$\underline{0.411 \pm 0.019}$	0.377 ± 0.016
4-Experts	$\mathbf{0.422 \pm 0.002}$	0.391 ± 0.011	$\underline{0.381 \pm 0.009}$
3-Experts (Ours)	$\underline{0.420 \pm 0.003}$	$\mathbf{0.413 \pm 0.019}$	$\mathbf{0.384 \pm 0.014}$
Boundary Routing			
Discrete	$\mathbf{0.422 \pm 0.002}$	$\underline{0.373 \pm 0.021}$	$\underline{0.362 \pm 0.019}$
Stochastic	0.415 ± 0.002	0.357 ± 0.020	0.348 ± 0.014
Mixing (Ours)	$\underline{0.420 \pm 0.003}$	$\mathbf{0.413 \pm 0.019}$	$\mathbf{0.384 \pm 0.014}$