Reward Score Matching: Unifying Reward-based Fine-tuning for Flow and Diffusion Models

Jeongjae Lee* **Jinho Chang*** **Jeongsol Kim†** **Jong Chul Ye†**
Graduate School of AI
KAIST, Korea
{jaylee2000, jinhojsk515, jeongsol, jong.ye}@kaist.ac.kr
*First authors †Corresponding authors

Abstract

Reward-based fine-tuning steers a pretrained diffusion or flow-based generative model toward higher-reward samples while remaining close to the pretrained model. Although existing methods are derived from different perspectives, we show that many can be written under a common framework, which we call *reward score matching* (RSM). Under this view, alignment becomes score matching against a value-guided target, and the main differences across methods reduce to the construction of the *value-guidance estimator* and the *effective optimization strength* across timesteps. This unification clarifies the bias–variance–compute tradeoffs of existing designs, and distinguishes core optimization components from auxiliary mechanisms that add complexity without clear benefit. Guided by this perspective, we develop simpler, more efficient redesigns across representative differentiable and black-box reward alignment tasks. Overall, RSM turns a seemingly fragmented collection of reward-based fine-tuning methods into a smaller, more interpretable, and more actionable design space.

Code: https://github.com/jaylee2000/rsm

1 Introduction

Reward-based fine-tuning steers a pretrained diffusion or flow-based generative model toward higher-reward samples without deviating too far from the pretrained model. Recent methods approach this problem from seemingly different perspectives, including entropy-regularized reinforcement learning (Soft RL) [3, 10, 13, 23, 24, 27, 57], optimal control [8, 29], direct reward backpropagation [7, 55], and GFlowNets [28, 30, 59]. As a result, the literature appears fragmented: across different objectives, derivations, and stabilization heuristics, it is difficult to distinguish fundamental differences from less consequential ones.

We show that these apparently different methods admit a common L_2 score-matching objective, which we term *Reward Score Matching* (RSM; see Table 1). RSM casts reward fine-tuning as regression toward a value-guided target score: the reference score plus optimal value guidance. Under this view, existing methods differ primarily along three design axes: how they estimate value guidance, how strongly they apply it across timesteps, and how they realize the trust region around the old policy.

The central challenge is that optimal value guidance is intractable in high dimensions. RSM reveals that gradient-based methods construct first-order estimates of optimal value guidance, whereas gradient-free methods construct zeroth-order estimates. These estimators exhibit different bias–variance–compute tradeoffs. First-order estimators exhibit low variance but are severely biased at low-SNR timesteps, whereas zeroth-order estimators are unbiased yet increasingly noisy at high-SNR timesteps. Furthermore, the temporal weighting used by existing methods can be understood as compensating for the timestep-dependent failure modes of their estimators.

This perspective reframes the design of various methods as a common optimization problem: at which timesteps should guidance be applied, and how should computation be allocated to make that guidance

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

Table 1: **Various methods admit special cases of RSM.** The common RSM loss $\mathcal{L}(\theta)$ is given by $\mathbb{E}\left[C_1(t_i)\left(\|\boldsymbol{s}_{t_i}^{\theta} - (\boldsymbol{s}_{t_i}^{\text{ref}} + \gamma(t_i)\hat{\boldsymbol{\Psi}}_{t_i})\|^2 + C_2(t_i)\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\theta^\dagger}\|^2\right)\right]$. The value gradient estimator $\hat{\boldsymbol{\Psi}}_{t_i}$ is specified by estimator type, lookahead depth j, and branching width K_i. The *Normalized Influence Metric* $h(t_i)$ summarizes guidance strength.

Method	$\hat{\boldsymbol{\Psi}}_{t_i}(j, K_i)$	$\gamma(t_i)$	$C_1(t_i)$	$C_2(t_i)$	$h(t_i)^\dagger$
First-order estimators					
VGG-Flow‡	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{CS},1}[\mathbf{J}_{\text{Tw}}]$	$(1-t_i)^2\delta(t_i)$	$\frac{1}{d\delta(t_i)^2}$	0	$\frac{1}{\alpha d}(1-t_i)^2$
ReFL	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{CS},1}[\mathbf{I}]$	$\frac{1}{2\sqrt{\bar{\alpha}_{t_i}}}$	$\frac{\alpha}{\delta(t_i)^2}$	0	$\frac{1}{2\sqrt{\bar{\alpha}_{t_i}}\delta(t_i)}$
SQDF	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},1}(i-1,1)$	$\tilde{\gamma}_{\text{SQDF}}^{Nt_i}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	0	$\tilde{\gamma}_{\text{SQDF}}^{Nt_i}\frac{\sigma_{t_i}w(t_i)}{2}$
Residual ∇-DB‡	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},1}(i-1,1)$	$\bar{\alpha}_{t_{i-1}}$	$\frac{1}{d}\frac{\Omega(t_i)^2}{\sigma_{t_i}^4}$	$\frac{(1-\bar{\alpha}_{t_i})\sigma_{t_i}^4}{\Omega(t_i)^2}\frac{w_R}{w_F}$	$\frac{1}{\alpha d}\bar{\alpha}_{t_{i-1}}\frac{w(t_i)}{\sigma_{t_i}}$
DRaFT	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},1}(0,1)$	$\frac{\delta(t_i)^2\Omega(t_i)^2}{2\sigma_{t_i}^2}$	$\frac{\alpha}{\delta(t_i)^2}$	0	$\frac{\delta(t_i)\Omega(t_i)}{2}$
DRaFT-K	$\mathbf{1}[i\le K]\,\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},1}(0,1)$	$\frac{\delta(t_i)^2\Omega(t_i)^2}{2\sigma_{t_i}^2}$	$\frac{\alpha}{\delta(t_i)^2}$	0	$\mathbf{1}[i\le K]\frac{\delta(t_i)\Omega(t_i)}{2}$
Adjoint Matching	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{Adj}} \triangleq -\frac{1}{\alpha}\mathbf{a}_{t_i}$	1	$\frac{2}{\delta(t_i)}$	0	—
Zeroth-order estimators					
REINFORCE + KL reg.	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,1)$	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	0	$\frac{\sigma_{t_i}w(t_i)}{2}$
PPO / GRPO / PCPO-base *	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,1)$	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma_{t_i}w(t_i)}{2}$
PCPO-reweight (diffusion)	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,1)$	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma'_{t_i}w'(t_i)}{2}$
PCPO-reweight (flow)	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,1)$	$\frac{w'(t_i)}{w(t_i)}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\gamma(t_i)\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma_{t_i}w'(t_i)}{2}$
Branch-GRPO‡	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,b_i)$	1	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{\sigma_{t_i}w(t_i)}{2}$
TempFlow-GRPO	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,6)$	$\frac{9}{4}\sigma_{t_i}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	$\gamma(t_i)\frac{r(\boldsymbol{x}_0)}{\alpha}$	$\frac{9\sigma_{t_i}^2w(t_i)}{8}$
GRPO-Guard	$\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA},0}(0,1)$	$\frac{\sigma_{t_i}\Omega(t_i)}{\Delta t_i}$	$\frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$	0	$\frac{\sigma_{t_i}^3w(t_i)^2}{2(\Delta t_i)\delta(t_i)}$

† $w(t_i) = \Omega(t_i)\delta(t_i)/\sigma_{t_i}$ (see Appendix D.3).
‡ The reduction becomes explicit after pruning redundant auxiliary components (see Appendix G.1).
‡ For Branch-GRPO, $b_i = 2$ if $i \in \mathcal{B}$ and $b_i = 1$ otherwise.
* DDPO [3], DPOK [10], Flow-GRPO [27], DanceGRPO [57], and CPS [51] are instances of PPO / GRPO / PCPO-base.

reliable? It also reveals that some seemingly essential mechanisms reduce to coefficient choices or removable auxiliary components. Guided by this analysis, we redesign representative differentiable and black-box reward fine-tuning methods. For zeroth-order methods, we allocate estimator budget according to timestep-dependent estimator quality and make trust region realizations comparable across timesteps. For first-order methods, we replace biased local guidance with reward gradients evaluated on clean samples, and reallocate optimization strength toward previously suppressed low-SNR timesteps.

Across four representative baselines, our simple redesigns improve reward optimization efficiency while maintaining competitive tradeoffs between reward and distribution shift. In particular, our zeroth-order redesign reaches a GenEval [12] score of 0.97 with an estimated $5\times$ reduction in training cost relative to the reported TempFlow-GRPO baseline [13]. Together, these results establish RSM not only as a common language for existing methods, but also as a practical framework for designing simpler and more efficient reward fine-tuning algorithms.

2 Preliminaries

2.1 Flow-based Models

Let $p_0(\boldsymbol{x}_0)$ and $p_1(\boldsymbol{x}_1)$ denote the target and source distributions. Flow-based models transport $\boldsymbol{x}_1 \sim p_1$ to $\boldsymbol{x}_0 \sim p_0$ along a probability path $p_t(\boldsymbol{x}_t)$ induced by a flow map $\psi_t : \mathbb{R}^d \to \mathbb{R}^d$ on $t \in [0,1]$, with $\psi_1(\boldsymbol{x}_1) = \boldsymbol{x}_1$ and $\psi_0(\boldsymbol{x}_1) = \boldsymbol{x}_0$. The corresponding trajectory $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_1)$ is governed by the flow ODE

$$d\boldsymbol{x}_t = \boldsymbol{v}_t(\boldsymbol{x}_t)dt, \tag{1}$$

where $\boldsymbol{v}_t(\boldsymbol{x}) = \dot{\psi}_t(\psi_t^{-1}(\boldsymbol{x}))$ is the velocity field. Prior work [1, 8, 27, 43] identified a reverse-time flow SDE that preserves the marginal distribution as:

$$d\boldsymbol{x}_t = \Big(\boldsymbol{v}_t(\boldsymbol{x}_t) - \frac{\tilde{\sigma}_t^2}{2}\nabla \log p_t(\boldsymbol{x}_t)\Big)dt + \tilde{\sigma}_t d\boldsymbol{w}. \tag{2}$$

We denote the diffusion coefficient in the SDE by $\tilde{\sigma}_t$, and its discretized counterpart by $\sigma_t = \tilde{\sigma}_t\sqrt{\Delta t}$. In conditional flow matching [26], one defines a conditional flow $\psi_t(\cdot \mid \boldsymbol{x}_0) : \mathbb{R}^d \to \mathbb{R}^d$ with conditional velocity field $\boldsymbol{v}_t(\boldsymbol{x} \mid \boldsymbol{x}_0)$, and marginal velocity $\boldsymbol{v}_t(\boldsymbol{x}) = \int \boldsymbol{v}_t(\boldsymbol{x} \mid \boldsymbol{x}_0)\, p(\boldsymbol{x}_0 \mid \boldsymbol{x}_t)\, d\boldsymbol{x}_0$.

2.2 Affine Conditional Flow

A widely used family of conditional flows is the affine conditional flow, defined as

$$\psi_t(\boldsymbol{x}_1|\boldsymbol{x}_0) = a_t\boldsymbol{x}_0 + b_t\boldsymbol{x}_1 \triangleq \boldsymbol{x}_t \tag{3}$$

where $a_t \in \mathbb{R}$ and $b_t \in \mathbb{R}$ denote time-dependent variables. From the definition of marginal velocity field and $p_1 = \mathcal{N}(0, \mathbf{I})$, the flow-ODE constructed with the affine flow is [1]

$$d\boldsymbol{x}_t = \frac{\dot{a}_t}{a_t}\boldsymbol{x}_t dt + \left(\frac{\dot{a}_t b_t^2 - a_t\dot{b}_t b_t}{a_t}\right)\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)dt. \tag{4}$$

We establish notation generally using the affine conditional flow, since it subsumes major generative paradigms including diffusion models (see Appendix B.1).

For brevity, we denote the score function as $\boldsymbol{s}_t \equiv \boldsymbol{s}(\boldsymbol{x}_t, t) \triangleq \nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)$ for the remainder of the paper. Then, the transition density of the discrete-time affine flow SDE takes the general form:

$$p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) = \mathcal{N}(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}_{t_i}, \sigma(t_i)^2\mathbf{I}), \tag{5}$$

where $\kappa(t_i)$ is fully determined by the definition of affine flow, and $\Omega(t_i), \sigma(t_i)$ are also affected by the stochastic noise schedule. Proof is in Appendix B.2 [2].

2.3 Training-based Reward Alignment

Soft RL formulates alignment as an entropy-regularized Markov decision process over discretized reverse-time transitions [2, 3, 10, 47]. With timesteps $0 = t_0 < t_1 < \cdots < t_N = 1$, the state is $\boldsymbol{x}_t \in \mathbb{R}^d$, the time-indexed policy is a learnable reverse transition $p^\theta_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})$, and the reward r is given only at the terminal state. This yields the objective

$$\max_\theta \mathbb{E}_{\tau\sim p^\theta}[r(\boldsymbol{x}_0) - \alpha\sum_i \mathcal{D}_{\mathrm{KL}}(p^\theta_{t_i}(\cdot|\boldsymbol{x}_{t_i})\|p^{\mathrm{ref}}_{t_i}(\cdot|\boldsymbol{x}_{t_i}))]. \tag{6}$$

The optimal marginal probability density is expressed with the *value function* $V_t(\boldsymbol{x}_t)$ related to the expected return,

$$p^\star_t(\boldsymbol{x}_t) = \frac{p^{\mathrm{ref}}_t(\boldsymbol{x}_t)}{Z}\exp\Big(\frac{V_t(\boldsymbol{x}_t)}{\alpha}\Big), \tag{7}$$

where Z is a normalizing constant independent of t [47].

Soft RL subsumes Residual ∇-DB [30], SQDF [21], policy gradients, etc., and provides the canonical reward-tilted target underlying RSM's formulation. Other reward-based fine-tuning methods arise from seemingly different derivations: VGG-Flow [29] directly regresses the model velocity field toward a reward-guided target, DRaFT [7] and ReFL [55] perform direct backpropagation through the reward, and Adjoint Matching [8] propagates reward gradients via the adjoint method. Several recent methods further introduce timestep-dependent weighting [4, 13, 23, 52], analogous to diffusion pretraining [15]. Despite these differences in derivation and heuristic temporal reweighting, we show that they admit the same RSM form, extending the score-matching view induced by Soft RL beyond methods originally derived from the Soft RL framework [3].

[1] For brevity, we write $\nabla_{\boldsymbol{x}_t} f(\boldsymbol{x}_t)$ as shorthand notation of $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_t}$ for the remainder of this paper.

[2] The network may instead parameterize a time-dependent affine transform of the score, absorbed into $\delta(t_i)$; see Appendix D.2. See also Appendix D.1 for derivations of $\Omega(t_i)$ for common samplers.

[3] DiffusionNFT [61] is closely related through reward-weighted regression, but does not reduce exactly to the same formulation; see Appendix H.2.1.

3 Reward Score Matching: A Unified Framework

3.1 Optimal Guidance and Common Objective

Eq. (7) implies that the optimal intermediate marginal $p_t^\star$ differs from the reference marginal p_t^{ref} by an exponential tilt of the value function V_t. Accordingly, the optimal score takes the form

$$\boldsymbol{s}_t^\star(\boldsymbol{x}_t) = \boldsymbol{s}_t^{\mathrm{ref}}(\boldsymbol{x}_t) + \frac{1}{\alpha}\nabla_{\boldsymbol{x}_t} V_t, \tag{8}$$

which leads to the following first order solver with the optimal value guidance:

Proposition 3.1 (Optimal value guidance)**.** *Consider a first-order solver with transitional kernel:*

$$p_{t_i}^{\Psi}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \mathcal{N}\Big(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\big(\boldsymbol{s}_{t_i}^{ref}(\boldsymbol{x}_{t_i}) + \boldsymbol{\Psi}_{t_i}\big),\ \sigma_{t_i}^2\mathbf{I}\Big).$$

The optimal value guidance $\boldsymbol{\Psi}_{t_i}$ *that matches the optimal transitional kernel* $p_{t_i}^\star(\cdot \mid \boldsymbol{x}_{t_i})$ *is*

$$\boldsymbol{\Psi}_{t_i}^\star(\boldsymbol{x}_{t_i}) \triangleq \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p_{t_i}^\star(\cdot|\boldsymbol{x}_{t_i})}\Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\Big]. \tag{9}$$

The optimal value guidance is generally intractable due to the integration with respect to the posterior. Therefore, practical methods use an *optimal value guidance estimate* $\hat{\boldsymbol{\Psi}}_{t_i}(\boldsymbol{x}_{t_i}) \approx \boldsymbol{\Psi}_{t_i}^\star(\boldsymbol{x}_{t_i})$ (see Appendix B.6 for details) and often apply a heuristic temporal weight $\gamma(t_i)$, yielding the *effective guidance* $\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i}) \triangleq \gamma(t_i)\hat{\boldsymbol{\Psi}}_{t_i}(\boldsymbol{x}_{t_i})$.

Since score networks are trained successfully by regression to target scores, the score function is parameterized by a score network and trained to satisfy $\boldsymbol{s}_t^\theta \approx \boldsymbol{s}_t$. In particular, Theorem 3.2 shows that existing methods can be unified by the regression problem of $\boldsymbol{s}_t^\theta$ toward the optimal value guidance estimate with $\boldsymbol{\Psi}_{t_i}$, yielding a common RSM objective. See Appendix C for proofs.

Theorem 3.2 (Reward Score Matching)**.** *The methods summarized in Table 1, including Soft RL methods, VGG-Flow, DRaFT, DRaFT-K, ReFL, and Adjoint Matching, admit the common loss*

$$\mathcal{L}(\theta) = \mathbb{E}_{t_i,\boldsymbol{x}_{t_i},\boldsymbol{\epsilon}}\Big[C_1(t_i)\Big(\|\boldsymbol{s}_{t_i}^\theta - (\boldsymbol{s}_{t_i}^{ref} + \boldsymbol{\Psi}_{t_i})\|^2 + C_2(t_i)\|\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger}\|^2\Big)\Big] \tag{10}$$

where C_1, C_2 *are weights, and* $\boldsymbol{s}^\theta, \boldsymbol{s}^{\theta^\dagger}, \boldsymbol{s}^{ref}$ *are the current, previous, and reference score functions* [4].

Remark 3.3. In the first on-policy update[5], the old-policy anchor is inactive. In presence of PPO clipping, the update is suppressed by setting $\boldsymbol{\Psi}_{t_i} \leftarrow \mathbf{0}$ and $C_2(t_i) \leftarrow 0$.

The gradient of Eq. (10) reveals update dynamics. We derive a *Canonical Gradient Form*:

$$\nabla_\theta\mathcal{L}(\theta) = 2\mathbb{E}[\mathbf{J}_\theta(\boldsymbol{s}_{t_i}^\theta)^\top \mathcal{G}(\boldsymbol{x}_{t_i})] \tag{11}$$

$$\mathcal{G}(\boldsymbol{x}_{t_i}) = C_1(t_i)\Big(-\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i}) + \overbrace{(\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\mathrm{ref}})}^{\text{KL Reg.}} + C_2(t_i)\overbrace{(\boldsymbol{s}_{t_i}^\theta - \boldsymbol{s}_{t_i}^{\theta^\dagger})}^{\text{Trust Region}}\Big), \tag{12}$$

where $\mathbf{J}_\theta$ denotes the Jacobian. This decomposition highlights three competing forces: *Effective Guidance* steers generation toward high rewards, while *KL Regularization* and the *Trust Region* term prevent model collapse by anchoring the model to the reference and previous policies, respectively. Section 3.1 is written in the entropy-regularized setting for generality, but Eq. (12) also reconciles the $\alpha = 0$ case for SQDF [21] and policy-gradient methods; see Appendix C for details.

Eq. (12) also clarifies the hierarchy of RSM's design dimensions. The primary source of variation between methods lies in the way $\hat{\boldsymbol{\Psi}}_{t_i}$ is constructed. By contrast, temporal weighting and trust-region realization act only as secondary coefficient-level modifiers via $\gamma(t_i), C_1(t_i), C_2(t_i)$ and clipping. We therefore focus first on estimator design, and defer these coefficient-level choices to Section 4.

[4] When $C_2(t) = 0$, Eq. (10) reduces to entropy-regularized optimal control (Appendix H.1).

[5] Throughout this paper, we use the relaxed notion of *on-policy* common in online RL algorithms, where data is collected under the old policy $\theta^\dagger$ and reused for one or more updates (e.g., DDPO [3]). When we need to refer specifically to the case $\theta = \theta^\dagger$, we distinguish it explicitly, e.g., by referring to it as the *first on-policy update* or *strictly on-policy*.

3.2 Estimating Optimal Value Guidance $\hat{\Psi}_{t_i}$

This section, which is one of the most important contributions of this paper, shows that the seemingly different forms of reward-based finetuning methods can be unified as the zeroth and first order estimate of the current or lookahead estimator of the optimal value guidance estimate. Specifically, Eq. (9) suggests two primary practical families of value-guidance estimators: *current-state* and *lookahead* [6].

Current-state Estimator. The first equality in Eq. (9) suggests a *current-state* estimator, available only in the first-order setting; VGG-Flow [29] and ReFL [55] fall in this family. Specifically, since we do not have access to the value function V but only to the terminal reward r, we approximate the value gradient via a Tweedie estimate.

Definition 3.4 (Current-state Estimator)**.** For a local pullback matrix A_{t_i}, the first-order current-state estimator is

$$\hat{\Psi}^{\mathrm{CS},1}_{t_i}[A_{t_i}] = \frac{1}{\alpha} A_{t_i}^{\top} \nabla_{\hat{\boldsymbol{x}}_{0|t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}) \;\approx\; \frac{1}{\alpha} \nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \tag{13}$$

VGG-Flow uses the full Tweedie pullback $A_{t_i} = \mathbf{J}_{\mathrm{Tw}} \triangleq \frac{\partial \hat{\boldsymbol{x}}_{0|t_i}}{\partial \boldsymbol{x}_{t_i}}$, whereas ReFL uses the identity pullback $A_{t_i} = \mathbf{I}$, with the remaining scalar Jacobian factor absorbed into $\gamma(t_i)$. Current-state estimators avoid stochastic rollouts, but incur substantial bias at low-SNR timesteps, as will be discussed later. They are therefore best viewed as simpler but potentially more biased alternatives to the lookahead estimators introduced below, which subsume most existing methods and expose the main design tradeoffs.

Lookahead Estimator. Lookahead estimators roll out from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_j}$ with $j < i$ and extract reward information there. First-order estimators require differentiable rewards, whereas zeroth-order estimators apply to arbitrary black-box rewards.

Definition 3.5 (Lookahead Estimators)**.** For *lookahead depth* $j < i$ and branching width K_i, the first-order and zeroth-order lookahead estimators of the existing reward-alignment methods are

$$\hat{\Psi}^{\mathrm{LA},1}_{t_i}(j, K_i) = \frac{\sigma_{t_i}^2}{\alpha \Omega(t_i)} \frac{1}{K_i} \sum_{k=1}^{K_i} \nabla_{\boldsymbol{x}^{(k)}_{t_{i-1}}} r(\hat{\boldsymbol{x}}^{(k)}_{0|t_j}) \approx \frac{\sigma_{t_i}^2}{\alpha \Omega(t_i)} \mathbb{E}_{\boldsymbol{x}_{t_{i-1}:t_j}} \Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_j}) \Big], \tag{14}$$

$$\hat{\Psi}^{\mathrm{LA},0}_{t_i}(j, K_i) = \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \frac{1}{K_i} \sum_{k=1}^{K_i} r(\hat{\boldsymbol{x}}^{(k)}_{0|t_j}) \, \boldsymbol{\epsilon}^{(k)}_{t_i} \quad \approx \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} \mathbb{E}_{\boldsymbol{x}_{t_{i-1}:t_j}} \Big[r(\hat{\boldsymbol{x}}_{0|t_j}) \, \boldsymbol{\epsilon}_{t_i} \Big], \tag{15}$$

where $\hat{\boldsymbol{x}}_{0|t_j} = \mathbb{E}[\boldsymbol{x}_0 \mid \boldsymbol{x}_{t_j}] \triangleq \mathbb{E}_{\boldsymbol{x}_0 \sim p_{t_j}(\boldsymbol{x}_0|\boldsymbol{x}_{t_j})}[\boldsymbol{x}_0]$ is the Tweedie estimate of the posterior mean.

Existing local first-order lookahead methods (Residual ∇-DB [30], SQDF [21]) use one-step lookahead ($j = i - 1$), and therefore inherit substantial Tweedie bias from $\hat{\boldsymbol{x}}_{0|t_j}$ at low-SNR timesteps, akin to current-state methods ($j = i$).

By contrast, DRaFT-K [7] uses full-lookahead guidance ($j = 0$) over the final K reverse steps, while truncating gradient propagation beyond them. Equivalently, it applies full-lookahead guidance at those K timesteps and sets the effective guidance to zero at all remaining timesteps[7].

Several local first-order methods (Residual ∇-DB, VGG-Flow) attempt to correct the bias of their local value-gradient approximation by learning the Jensen gap via a small neural network g_ϕ. Proposition 3.6 shows that, if this residual parameterization were exact, both local current-state and one-step first-order estimators would be unbiased; see Appendix B.4 for the proof.

Proposition 3.6 (Conditional Consistency of First-Order Estimators)**.** *Let* $j \in \{i, i-1\}$. *For*

$$\nabla_{\boldsymbol{x}_{t_j}} V_{t_j}(\boldsymbol{x}_{t_j}) = \gamma_{t_i} \nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j}), \tag{16}$$

define the residual-corrected first-order estimator:

$$\hat{\Psi}^{\mathrm{FO,res}}_{t_i}(j) \triangleq \left(\frac{1}{\alpha} \iota[j = i] + \frac{\sigma_{t_i}^2}{\alpha \Omega(t_i)} \iota[j = i - 1] \right) \Big(\gamma_{t_i} \nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j}) \Big), \tag{17}$$

[6] Adjoint Matching [8] provides an alternative first-order construction that propagates reward gradients via the adjoint method; see Appendix C.1.6.

[7] Untruncated DRaFT applies full-lookahead guidance ($j = 0$) at every timestep, but is impractical because backpropagating through the entire Jacobian chain incurs prohibitive memory costs.

where $\iota[\cdot]$ denotes the indicator function, and for $j = i-1$ we take $\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$, while for $j = i$ we simply set $\boldsymbol{x}_{t_j} = \boldsymbol{x}_{t_i}$. Then, $\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) \;\middle|\; \boldsymbol{x}_{t_i}\right]$ is an unbiased estimate of the value gradient.

However, as we later verify in Appendix G.1, auxiliary corrections g_ϕ turn out to be effectively negligible in the existing methods, so the resulting local current-state and one-step first-order estimators remain biased.

Existing zeroth-order (policy-gradient) methods typically use full rollout ($j = 0$)[8], which is an unbiased value gradient estimate. The following theorem, following from Stein's identity [44], completes the theoretical unification of zeroth- and first-order methods.

Theorem 3.7 (Consistency of Policy Gradient)**.** *For the full-rollout $j = 0$, the lookahead zeroth-order estimator of the existing reward-alignment methods is an unbiased estimator of the value gradient:*

$$\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i} \approx \frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p^\star_{t_{i:1}},\boldsymbol{\epsilon}_{t_{i:1}}\overset{\mathrm{iid}}{\sim}\mathcal{N}(\mathbf{0},\mathbf{I})}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}] = \mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}}[\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})]. \tag{18}$$

See Appendix B.5 for the proof. Lookahead estimators introduce two design dimensions for variance reduction. *Branching* specifies per-step branch widths K_i, with $K_i = 1$ denoting a point estimate at t_i; larger K_i reduce variance at higher compute cost [13, 25]. *Stochasticity localization* changes the underlying stochastic control problem by keeping only selected reverse steps stochastic [13, 24].

4 Practical Design Choices of RSM from Optimization Perspectives

Section 3 established exact reductions of many existing methods to a common RSM objective. We now shift to the practical optimization view: under finite compute and noisy gradient estimates, performance is governed by three coupled dimensions: *estimator design*, *temporal weighting*, and *trust-region realization*. These determine the quality of value guidance and how compute, optimization mass, and regularization are allocated across timesteps.

Estimator design determines the quality of $\hat{\boldsymbol{\Psi}}_{t_i}$. Its main knobs are lookahead depth j, branching width $\{K_i\}$, and stochasticity localization. Deeper lookahead reduces bias at higher compute cost, while larger branching widths reduce finite-sample variance at higher cost. Stochasticity localization can likewise improve estimator quality, but unlike branching, it also changes the underlying stochastic control problem. Reward centering, e.g., subtracting the sample mean, may reduce variance for free without changing the underlying gradients. Under a fixed compute budget, two allocation problems emerge: how to distribute the total compute budget *across timesteps*, and how to use compute *within each timestep* to obtain a useful guidance estimate.

An important example is the zeroth-order estimator $\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i}$, whose conditional variance under independent descendant rollouts scales as:

$$\mathrm{Cov}\left(\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i} \mid \boldsymbol{x}_{t_i}\right) = \frac{1}{K_i}\left(\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\right)^2 \mathrm{Cov}(r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i} \mid \boldsymbol{x}_{t_i}) \propto \frac{1}{K_i}\left(\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\right)^2. \tag{19}$$

This scaling shows that, under a fixed branching width K_i, some timesteps are intrinsically noisier than others and may therefore require larger K_i to reduce variance, at the cost of additional compute.

Temporal weighting. Another key question is how strongly the reward signal enters the update at each timestep. We seek a common coefficient measuring its timestep-wise optimization strength. For current-state estimators, Eq. (13) shows that $\hat{\boldsymbol{\Psi}}_{t_i}$ is proportional to a reward gradient estimate at the current state. For lookahead estimators, Theorem 3.7 together with Eqs. (14)–(15) yields:

$$\widehat{\nabla r(\boldsymbol{x}_{t_i})} = \frac{1}{\sigma_{t_i}}\mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p^\star_{t_{i:1}},\boldsymbol{\epsilon}_{t_{i:1}}\overset{\mathrm{iid}}{\sim}\mathcal{N}(\mathbf{0},\mathbf{I})}\left[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}\right] = \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p^\star_{t_{i:1}}}\left[\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)\right]. \tag{20}$$

[8] In principle, first-order consistency can also be achieved by full rollout, i.e. by selecting $j = 0$ and differentiating the terminal reward through the entire denoising trajectory. However, this requires the full Jacobian chain and is therefore computationally impractical and potentially unstable in high dimension.

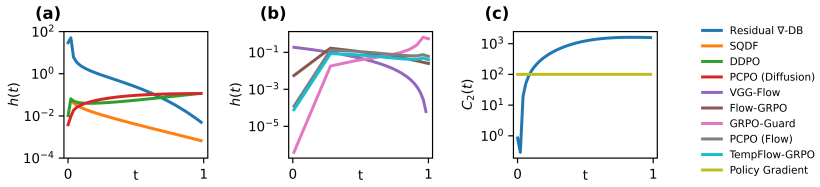

Figure 1: **Temporal Optimization Strength.** (a, b) Many first-order methods reduce value guidance at low-SNR timesteps; improved zeroth-order methods reduce value guidance at high-SNR timesteps: (a) diffusion, (b) flow-matching. (c) Residual ∇-DB enforces stronger trust-region constraints for low-SNR timesteps. Policy Gradient's $C_2(t)$ used constants $r(\boldsymbol{x}_0) = 1, \alpha = 10^{-2}$.

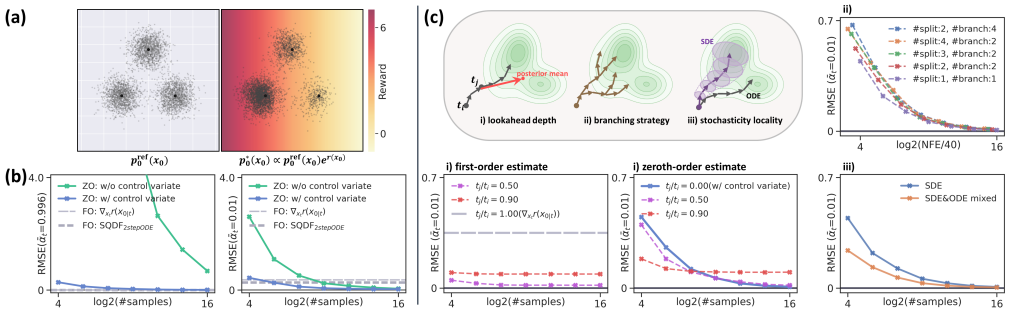

Figure 2: **Toy analysis of estimator quality under fixed compute.** (a) Reference and reward-tilted distributions. (b) RMSE of representative first-order (FO) and zeroth-order (ZO) estimators[9] at two timesteps. (c) RMSE–NFE for estimators with varying lookahead depth, branching, and stochasticity localization. *#split* is the number of recursive branching stages; *#branch* is the number of child trajectories per stage. Mixed SDE-ODE sampling follows TempFlow-GRPO [13].

Thus, all cases admit a common reward-gradient signal. Using the parameterization-conversion factor $\delta(t_i)$ (Appendix D.2), defined by $-\delta(t)(\boldsymbol{g}_t^\theta - \boldsymbol{g}_t^{\text{ref}}) = \boldsymbol{s}_t^\theta - \boldsymbol{s}_t^{\text{ref}}$ for $\boldsymbol{g} \in \{\boldsymbol{\epsilon}, \boldsymbol{v}\}$, Eq. (11) becomes

$$\nabla_\theta \mathcal{L}(\theta) = 2\mathbb{E}[\mathbf{J}_\theta(\boldsymbol{g}_{t_i}^\theta)^\top \tilde{\mathcal{G}}(\boldsymbol{x}_{t_i})] \tag{21}$$

$$\tilde{\mathcal{G}}(\boldsymbol{x}_{t_i}) = h(t_i)\widehat{\nabla r(\boldsymbol{x}_{t_i})} + \delta(t_i)^2 C_1(t_i)\Big(\overbrace{(\boldsymbol{g}_{t_i}^\theta - \boldsymbol{g}_{t_i}^{\text{ref}})}^{\text{KL Reg.}} + C_2(t_i)\overbrace{(\boldsymbol{g}_{t_i}^\theta - \boldsymbol{g}_{t_i}^{\theta^\dagger})}^{\text{Trust Region}}\Big). \tag{22}$$

$$h(t_i) := \delta(t_i)C_1(t_i) \times \begin{cases} \gamma(t_i)/\alpha, & \text{(current-state)}, \\ \gamma(t_i)\sigma_{t_i}^2/(\alpha\Omega(t_i)), & \text{(lookahead)}. \end{cases} \tag{23}$$

The *Normalized Influence Metric* $h(t)$ measures the effective scaling of the reward gradient after accounting for temporal weighting and parameterization. For representative methods (Fig. 1), $h(t)$ reveals a clear contrast: representative first-order methods concentrate mass on high-SNR timesteps, suppressing low-SNR steps where bias is largest, whereas improved zeroth-order methods do the opposite, suppressing high-SNR steps where variance is largest under fixed branching width (Eq. (19)). Note that, while omitted for simplicity, temporal weighting also subsumes any rescaling of reward-gradients (first-order) or rewards (zeroth-order).

Trust-region realization appears through $C_2(t)$ and clipping. Optimization strength depends not only on $h(t)$, but also on how much of the reward gradient survives regularization and clipping.

5 Experiments

We study the practical design space of RSM in two stages. First, we isolate estimator design at a single timestep under fixed compute, asking which local allocations estimate $\boldsymbol{\Psi}_{t_i}^\star$ most effectively. Second, we study end-to-end design under fixed total compute, asking how estimator design, temporal weighting, and trust-region realization should be chosen jointly across timesteps.

5.1 Isolated Estimator Analysis

Setup. We construct a 2D toy problem with a Gaussian-mixture reference distribution $p_0^{\text{ref}}(\boldsymbol{x}_0)$ and simple $r(x)$, yielding an analytically tractable $\boldsymbol{\Psi}_{t_i}^\star$. We compare estimators $\hat{\boldsymbol{\Psi}}_{t_i}^{\text{LA}}$ from Eqs. (14)–(15) against ground truth $\boldsymbol{\Psi}_{t_i}^\star$ by RMSE under matched compute. See Appendix E.1 for details.

[9] $\text{SQDF}_{2stepODE}$ refers to the 2-step ODE estimator proposed in SQDF [21].

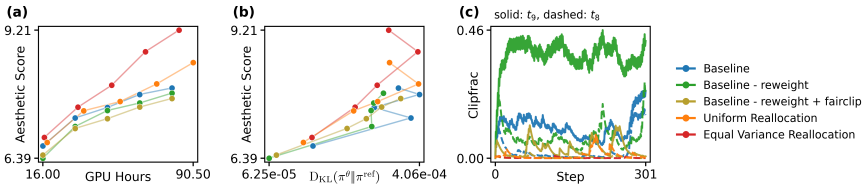

Figure 3: **Improving high-SNR timesteps is better than merely suppressing them.** Making clipping timestep-fair and reallocating budget improves performance and stability. (a) Aesthetic Score vs. GPU hours. (b) Aesthetic Score vs. $\mathcal{D}_{\mathrm{KL}}$. (c) Clip fraction for t_9 (solid) and t_8 (dashed).

Results. Fig. 2(b) shows that one-step first-order lookahead is severely biased at low SNR, whereas full-rollout zeroth-order guidance is unbiased but requires many more samples. Reward centering reduces RMSE at no additional cost. Fig. 2(c-i) shows that lookahead depth j induces a bias–variance–compute tradeoff. Although deeper lookahead reduces bias, under limited samples shallower lookahead can be both more accurate and more compute-efficient, especially for zeroth-order estimators, because averaging over posterior uncertainty lowers variance relative to full rollout. At fixed lookahead depth, first-order estimators typically improve faster with sample count than zeroth-order estimators[10]. Fig. 2(c-ii) shows that, under fixed total sample budget, different branching patterns yield similar RMSE–NFE trends[11]. Fig. 2(c-iii) shows that stochasticity localization improves estimator quality under fixed compute by making the induced value function easier to predict[12].

Findings. Under finite compute, the best local estimator need not be the least biased one: shallower lookahead estimators may outperform unbiased full-rollout estimators. Our results also suggest that point estimates are too noisy. Once branching is used at all, the specific choice of (#split, #branch) is relatively less important. In addition, subtracting the sample mean substantially reduces variance.

5.2 End-to-End Design

We now test whether these design principles improve end-to-end optimization under matched compute. Our improvements require comparable or less per-epoch compute than baselines; see Appendices E, F.1 and I for experiment details, more ablations and qualitative results, respectively.

Mechanistic Analysis. We begin with a case study on the zeroth-order flow-matching baseline TempFlow-GRPO. Eq. (19) suggests that, under fixed branching width, high-SNR timesteps are intrinsically noisier. Yet the baseline allocates no additional budget to them, relying instead on strong clipping and downweighting for stability.

To address this inefficiency, we propose a redesign in three steps. First, we make clipping comparable across timesteps by moving the timestep-dependent scale outside the clipping operator:

$$\mathrm{clip}_{\xi}\left(\frac{\|\mu_{t_i}^{\theta}-\mu_{t_i}^{\theta^{\dagger}}\|^2}{2\tilde{\sigma}_{t_i}^2\Delta t_i}\right) \longrightarrow \frac{1}{\tilde{\sigma}_{t_i}^2\Delta t_i}\mathrm{clip}_{\xi}\left(\frac{\|\mu_{t_i}^{\theta}-\mu_{t_i}^{\theta^{\dagger}}\|^2}{2}\right). \tag{24}$$

Second, we reallocate branching budget toward high-SNR timesteps, to reduce estimator variance where it is largest. Third, despite this redistribution, we find that t_9 remains too noisy and heavily clipped to justify investment, so we deactivate it and redistribute its budget to other timesteps.

Fig. 3 shows that uniform redistribution already improves over TempFlow-GRPO, while variance-aware redistribution performs best. Our redesign improves reward more quickly while maintaining a comparable reward–KL tradeoff, suggesting that budget should be concentrated on timesteps whose guidance can be made reliable rather than on timesteps whose updates are later suppressed.

Validation Across Settings. In zeroth-order flow matching, uniform branching across timesteps is suboptimal because timestep importance is reward-dependent. For the primarily semantic GenEval [12] reward, high-SNR timesteps matter less, since semantic structure is mainly determined at low

[10] In this toy analysis, first-order curves use the full Jacobian chain to compute the exact first-order signal.

[11] Larger K_i improves the conditional estimate for a given parent latent, but under fixed budget reduces the number of distinct parent latents explored at timestep t_i. This tradeoff is not captured here, but is important in practice.

[12] As discussed in Section 3.2, this benefit comes partly from changing the underlying stochastic control problem rather than merely reducing variance for the same target.

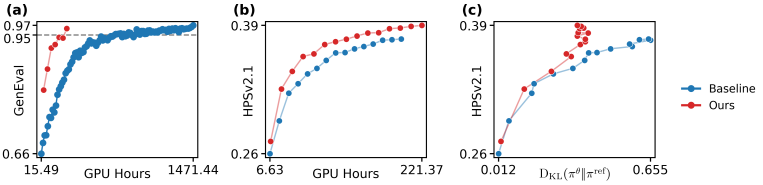

Figure 4: **Validation: Zeroth-order methods.** Principled budget allocation and temporal weighting improve performance on (a) GenEval with SD3.5-M[16] and (b, c) HPSv2.1 with SD1.5.

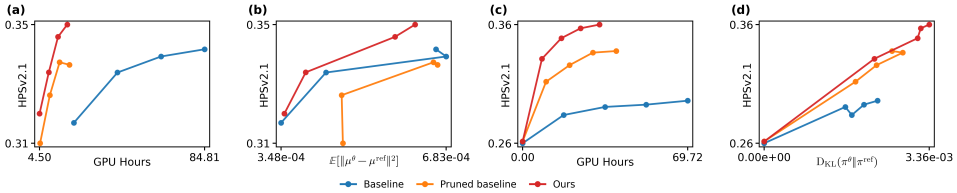

Figure 5: **Validation: First-order methods.** Improved reward guidance for low-SNR timesteps yields faster reward gains, while maintaining a competitive reward–KL tradeoff on (a, b) SD3.5-M and (c, d) SD1.5. See Appendix F.2 for more results.

SNR [58]. A redesign that concentrates branching budget on low-SNR timesteps reaches GenEval $= 0.97$ with an estimated $5\times$ speedup over the reported TempFlow-GRPO baseline (Fig. 4(a)).

In zeroth-order diffusion, the PCPO [23] baseline uses $K_i = 1$ for $\forall i$; each timestep receives a high-variance point estimate of the value gradient. Ideally, one would choose $K_i \gg 1$ at all timesteps, but doing so over the full trajectory ($N = 50$) is prohibitively expensive. As a practical compromise, we choose $K_i \gg 1$ at a sparse subset of steps and $K_i = 0$ elsewhere. This sparse-allocation design substantially improves efficency while achieving a better reward–KL tradeoff (Fig. 4(b, c)).

Existing first-order baselines [21, 29, 30] rely on Tweedie-based guidance. The severe bias at low-SNR timesteps is largely obscured by their original aesthetic-dominant evaluation settings[13] and aggressive low-SNR downweighting. When fine-tuning on HPSv2.1 with GenEval training prompts, where both aesthetic quality and semantic fidelity matter, performance drops sharply under original designs, since meaningful learning must occur at low-SNR timesteps as well. We therefore (i) replace these local gradients with terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$, effectively moving to $j = 0$, which yields a practical *reward score distillation* update[14], while (ii) reversing the inherited low-SNR downweighting. Across both flow-matching and diffusion backbones, this yields substantially faster reward improvement while remaining competitive in the reward–KL tradeoff (Fig. 5)[15].

6 Discussion

Broadening the framework. RSM covers a broad class of affine flow-based reward fine-tuning methods, including methods derived from Soft RL as well as seemingly different formulations such as VGG-Flow, DRaFT/DRaFT-K, ReFL, and Adjoint Matching. Several related objectives remain closely connected but non-identical. Soft Q-learning is also a subset of soft RL but admits slightly different gradients [42] (see Appendix B.6 for details).

DiffusionNFT [61] can be interpreted as reward-weighted regression, which is closely connected to Soft RL but does not reduce exactly to the RSM objective (see Appendix H.2.1). This connection suggests a natural first-order analogue of reward-weighted regression, obtained through a straightforward extension of DRaFT-LV (see Appendix H.2.2).

[13] Prior first-order methods are evaluated either on purely aesthetic rewards or on HPSv2.1 [54] with prompt subsets such as `hpd_photo_painting`, where semantic demands are relatively weak.

[14] Although the exact quantity $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$ would provide an unbiased first-order guidance signal, computing it requires a high-order Jacobian chain through the full denoising trajectory, which is both computationally expensive and potentially destabilizing, as noted in Poole et al. [35]. We therefore use the practical surrogate $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$; see Appendix E for details.

[15] We display *transition mean drift* for first-order flow matching, since KL divergence is undefined for the VGG-Flow baseline [29]; see Appendices E.2 and F.1.

[16] Due to resource constraints, we did not rerun the GenEval baseline; instead, we plot reverse-engineered data from Figure 3 of He et al. [13]. As a result, reward–KL tradeoff comparisons are unavailable for GenEval.

A complementary direction is to improve critic learning [19, 60], since existing auxiliary networks g_ϕ often collapse under weak supervision and may benefit from better bootstrapped or consistency-based targets.

Related ideas extend to inference-time alignment, which can also be understood through the lens of value gradients [16, 49]. Appendix H.3 shows that methods such as DNO [46] rely on closely related zeroth-order principles.

Better tuning of the design space. Rather than performing exhaustive tuning or aggressive engineering, we tested only a small number of principled modifications, yet observed substantial gains. These results suggest considerable headroom for more systematic exploration of RSM's design space. Further simplification may also be possible, for example by absorbing the trust-region term C_2 into temporal weighting, as in GRPO-Guard [52] (see Eq. (140)). In addition, joint optimization of the sampler and training objective, as in PCPO (diffusion) [23], could enable more fine-grained control over estimator variance and native temporal weights. Furthermore, for differentiable rewards, interpolating between first- and zeroth-order estimators [38] may yield a more preferable tradeoff.

Limitations. Although our redesigns improve existing reward-based fine-tuning methods, they do not eliminate reward hacking. Our auxiliary evaluations suggest that the gains are not driven by *more* aggressive reward hacking, but they do not establish emergent robustness to reward misspecification. More broadly, while RSM exactly covers a broad class of reward-based fine-tuning methods for flow-based models, its current formulation does not cover offline preference-based objectives such as Diffusion-DPO [50].

Broader Impact. Improving alignment of flow-based models is important for generating data with desired properties, thereby better serving human needs across various applications. Such methods may have harmful downstream uses, but we are not aware of any additional negative impacts specific to this work that require particular emphasis.

7 Conclusion

We introduced Reward Score Matching (RSM), a unified view of reward-based fine-tuning for flow and diffusion models. Under this view, many methods that are motivated from different perspectives reduce to the same underlying score-matching problem, with their main differences arising from how value gradients are estimated, weighted, and regularized across timesteps.

This perspective also clarifies the practical structure of the design space. In particular, it puts first-order and zeroth-order methods on common ground as different estimators of the same target, exposing shared bias–variance–compute tradeoffs that are otherwise obscured by method-specific derivations. It further shows that much of the remaining variation can be understood through three coupled design dimensions: estimator design, temporal weighting, and trust-region realization.

Guided by this framework, we identified several simplifications and principled redesigns that improve fine-tuning-based alignment across representative differentiable and black-box rewards, often at substantially lower compute cost. Taken together, these results suggest that progress depends less on introducing increasingly specialized objectives and more on understanding how to estimate, weight, and preserve value-guided updates effectively. We aspire for RSM to provide a common language for comparing existing methods, and a practical foundation for designing stronger ones.

Acknowledgement

This work was supported by the National Research Foundation of Korea under Grant RS-2024-00336454, the Advanced GPU Utilization Support Program funded by the Government of the Republic of Korea (Ministry of Science and ICT) (02-26-01-0404), and the AI Computing Infrastructure Enhancement (GPU Rental Support) User Support Program funded by the Ministry of Science and ICT (MSIT) Republic of Korea (RQT-25-120217).

References

[1] Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209): 1–80, 2025. URL http://jmlr.org/papers/v26/23-1605.html.

[2] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957.

[3] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

[4] Jaemoo Choi, Yuchen Zhu, Wei Guo, Petr Molodyk, Bo Yuan, Jinbin Bai, Yi Xin, Molei Tao, and Yongxin Chen. Rethinking the design space of reinforcement learning for diffusion models: On the importance of likelihood estimation beyond loss design, 2026. URL https://arxiv.org/abs/2602.04663.

[5] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11472–11481, June 2022.

[6] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

[7] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1vmSEVL19f.

[8] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xQBRrtQM8u.

[9] 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. *arXiv preprint arXiv:2403.03206*, 2024. URL https://arxiv.org/abs/2403.03206.

[10] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=8OTPepXzeh.

[11] Stephanie Fu, Netanel Yakir Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=DEiNSfh1k7.

[12] 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.

[13] Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. TempFlow-GRPO: When timing matters for GRPO in flow models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7mCo3R3Wyn.

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[16] Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps, 2026. URL https://arxiv.org/abs/2602.05993.

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[18] Nai-Chieh Huang, Ping-Chun Hsieh, Kuo-Hao Ho, and I-Chen Wu. PPO-Clip attains global optimality: Towards deeper understandings of clipping. In *AAAI*, pages 12600–12607, 2024.

[19] Cristian Perez Jensen, Luca Schaufelberger, Riccardo De Santi, Kjell Jorner, and Andreas Krause. Value matching: Scalable and gradient-free reward-guided flow adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7iXt44Actj.

[20] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

[21] Hyeongyu Kang, Jaewoo Lee, Woocheol Shin, Kiyoung Om, and Jinkyoo Park. Diffusion fine-tuning via reparameterized policy gradient of the soft Q-Function, 2026. URL https://openreview.net/forum?id=8zoxC9e23q.

[22] 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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[23] Jeongjae Lee and Jong Chul Ye. PCPO: Proportionate credit policy optimization for preference alignment of image generation models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=alY08iknli.

[24] Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE-SDE, 2025. URL https://arxiv.org/abs/2507.21802.

[25] Yuming Li, Yikai Wang, Yuying Zhu, Zhongyu Zhao, Ming Lu, Qi She, and Shanghang Zhang. BranchGRPO: Stable and efficient GRPO with structured branching in diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T2nP2IQasd.

[26] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[27] 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*, 2025. URL https://openreview.net/forum?id=oCBKGw5HNf.

[28] Qingming Liu, Zhen Liu, Dinghuai Zhang, and Kui Jia. Nabla-R2D3: Effective and efficient 3D diffusion alignment with 2D rewards. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Dk2qprCnu8.

[29] Zhen Liu, Tim Z. Xiao, Carles Domingo-Enrich, Weiyang Liu, and Dinghuai Zhang. Value gradient guidance for flow matching alignment. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=6MmOy2Ji8V.

[30] Zhen Liu, Tim Z. Xiao, Weiyang Liu, Yoshua Bengio, and Dinghuai Zhang. Efficient diversity-preserving diffusion alignment via gradient-informed GFlownets. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Aye5wL6TCn.

[31] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, June 2025. ISSN 2731-5398. doi: 10.1007/s11633-025-1562-4. URL http://dx.doi.org/10.1007/s11633-025-1562-4.

[32] Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70 (2):583–601, Mar 2002. doi: 10.1111/1468-0262.00296.

[33] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *CoRR*, abs/1705.07798, 2017. URL http://arxiv.org/abs/1705.07798.

[34] Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of GFlowNets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, 2023. URL https://openreview.net/forum?id=beHp3L9KXc.

[35] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FjNys5c7VyY.

[36] Jannes Quer and Enric Ribera Borrell. Connecting stochastic optimal control and reinforcement learning. *Journal of Mathematical Physics*, 65(8), 2024.

[37] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

[38] Tao Ren, Zishi Zhang, Jinyang Jiang, Zehao Li, Shentao Qin, Yi Zheng, Guanghao Li, Qianyou Sun, Yan Li, Jiafeng Liang, Xinping Li, and Yijie Peng. Half-order fine-tuning for diffusion model: A recursive likelihood ratio optimizer. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AZ6lqcvHLX.

[39] 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*, pages 10684–10695, 2022.

[40] Chrisoph Schuhmann. Laion aesthetics, Aug 2022. URL https://laion.ai/blog/laion-aesthetics/.

[41] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

[42] John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft Q-Learning, 2018. URL https://arxiv.org/abs/1704.06440.

[43] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[44] Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981.

[45] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[46] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=JpbqiD7n9r.

[47] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review, 2024. URL https://arxiv.org/abs/2407.13734.

[48] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

[49] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review, 2025. URL https://arxiv.org/abs/2501.09685.

[50] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8228–8238, June 2024.

[51] Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching, 2025. URL https://arxiv.org/abs/2509.05952.

[52] Jing Wang, Jiajun Liang, Jie Liu, Henglin Liu, Gongye Liu, Jun Zheng, Wanyuan Pang, Ao Ma, Zhenyu Xie, Xintao Wang, Meng Wang, Pengfei Wan, and Xiaodan Liang. GRPO-Guard: Mitigating implicit over-optimization in flow matching via regulated clipping, 2025. URL https://arxiv.org/abs/2510.22319.

[53] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[54] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Lai. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023. URL https://arxiv.org/abs/2306.09341.

[55] 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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15903–15935, 2023.

[56] Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models, 2025. URL https://arxiv.org/abs/2509.25050.

[57] Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing grpo on visual generation, 2025. URL https://arxiv.org/abs/2505.07818.

[58] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23174–23184, October 2023.

[59] Dinghuai Zhang, Yizhe Zhang, Jiatao Gu, Ruixiang Zhang, Joshua M. Susskind, Navdeep Jaitly, and Shuangfei Zhai. Improving GFlownets for text-to-image diffusion alignment. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=XDbY3qhM42.

[60] Hanyang Zhao, Haoxian Chen, Ji Zhang, David Yao, and Wenpin Tang. Score as action: Fine tuning diffusion generative models by continuous-time reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=V5HEX6stS2.

[61] Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VJZ477R89F.

Table 2: **Notation guide.** Main symbols used in the unified RSM framework.

Symbol	Meaning	Main role
$\boldsymbol{\Psi}^{\star}_{t_i}$	Optimal value guidance, $\frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}$	Ideal reward-guided correction to the reference score
$\hat{\boldsymbol{\Psi}}_{t_i}$	Practical estimate of $\boldsymbol{\Psi}^{\star}_{t_i}$	Guidance estimate before temporal reweighting
$\boldsymbol{\Psi}_{t_i}$	Effective guidance, $\gamma(t_i)\hat{\boldsymbol{\Psi}}_{t_i}$	Guidance actually used in the loss
$\gamma(t_i)$	Heuristic temporal weight	Controls timestep-wise guidance strength
$C_1(t_i)$	Main timestep weight in Eq. (10)	Scales the score-matching update at t_i
$C_2(t_i)$	Trust-region coefficient in Eq. (10)	Scales anchoring to the previous policy / score
$h(t_i)$	Normalized Influence Metric	Effective timestep-wise optimization strength
$\Omega(t_i)$	Score coefficient in the reverse transition kernel	Converts score guidance into mean shift
$\delta(t_i)$	Parameterization-conversion factor	Maps native outputs to score differences
j	Lookahead depth	Sets how far rollout proceeds before reward extraction
K_i	Branching width at timestep t_i	Number of descendants used for conditional estimation

A Notation Guide

Table 2 summarizes the main symbols used throughout the paper.

B Main Proofs

B.1 Generality of affine conditional flows

We show that the affine conditional flow (Eq. (4)) subsumes VP-SDE, VE-SDE, and Rectified Flow.

VP-SDE. We define the coefficients $a_t = \sqrt{\bar{\alpha}_t}$ and $b_t = \sqrt{1-\bar{\alpha}_t}$, where $\bar{\alpha}_t = \exp\left(-\int_0^t \beta(s)ds\right)$ is derived from a pre-defined noise scheduler $\beta(s)$ satisfying $\beta(0) = 0$ and strictly increasing $\beta(t)$. Consequently, the flow ODE becomes

$$d\boldsymbol{x}_t = -\frac{1}{2}\beta(t)\boldsymbol{x}_t dt - \frac{1}{2}\beta(t)\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)dt. \tag{25}$$

VE-SDE. Let $a_t = 1$ and $b_t = \sigma_t$ where σ_t denotes a pre-defined noise variance schedule satisfying $\sigma_0 = 0$ and $\sigma_t < \sigma_{t+1}$. The corresponding flow ODE is

$$d\boldsymbol{x}_t = -\dot{\sigma}_t\sigma_t\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)dt. \tag{26}$$

Rectified Flow. Let $a_t = 1 - t$ and $b_t = t$ for $t \in [0, 1]$. Then, the flow ODE becomes

$$d\boldsymbol{x}_t = \frac{\boldsymbol{x}_t - \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]}{t}dt. \tag{27}$$

B.2 Proof of discretized affine flow SDE

Proof. Consider a conditional affine flow $\boldsymbol{x}_t := \psi_t(\boldsymbol{x}_1|\boldsymbol{x}_0) = a_t\boldsymbol{x}_0 + b_t\boldsymbol{x}_1$ with $\boldsymbol{x}_1 \sim \mathcal{N}(0, \mathbf{I})$. The corresponding flow ODE is

$$\begin{aligned} d\boldsymbol{x}_t = \boldsymbol{v}_t(\boldsymbol{x}_t, t)dt &= \frac{\dot{a}_t}{a_t}\boldsymbol{x}_t dt + \left(\frac{\dot{a}_t b_t^2 - a_t\dot{b}_t b_t}{a_t}\right)\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t)dt \\ &= A(t)\boldsymbol{x}_t dt + B(t)\boldsymbol{s}(\boldsymbol{x}_t, t). \end{aligned} \tag{28}$$

From Song et al. [43], the flow SDE that matches the same marginal density across time t is

$$d\boldsymbol{x}_t = [\boldsymbol{v}_t(\boldsymbol{x}_t, t) - \frac{1}{2}g_t\boldsymbol{s}(\boldsymbol{x}_t, t)]dt + \sqrt{g_t}d\boldsymbol{w}_t \tag{29}$$

$$= [A(t)\boldsymbol{x}_t + (B(t) - \frac{1}{2}g_t)\boldsymbol{s}(\boldsymbol{x}_t, t)]dt + \sqrt{g_t}d\boldsymbol{w}_t \tag{30}$$

$$= [A(t)\boldsymbol{x}_t + \tilde{B}(t)\boldsymbol{s}(\boldsymbol{x}_t, t)]dt + \sqrt{g_t}d\boldsymbol{w}_t, \tag{31}$$

where we set $\tilde{B}(t) = B(t) - 0.5g_t$. Note that $\tilde{B}(t) = B(t)$ when $g_t = 0$ (i.e., flow ODE).

Consider an integrating factor $\Phi(t) = \exp\left(\int A(\tau)d\tau\right)$. For $\boldsymbol{y}_t = \Phi(t)^{-1}\boldsymbol{x}_t$, we apply the Itô derivative and obtain

$$d\boldsymbol{y}_t = \Phi(t)^{-1}d\boldsymbol{x}_t + \boldsymbol{x}_t d\Phi(t)^{-1} \tag{32}$$

$$= \Phi(t)^{-1}[A(t)\boldsymbol{x}_t dt + \tilde{B}(t)\boldsymbol{s}(\boldsymbol{x}_t, t)dt + \sqrt{g_t}d\boldsymbol{w}_t] - A(t)\Phi(t)^{-1}\boldsymbol{x}_t dt \tag{33}$$

$$= \Phi(t)^{-1}[\tilde{B}(t)\boldsymbol{s}(\boldsymbol{x}_t, t)dt + \sqrt{g_t}d\boldsymbol{w}_t], \tag{34}$$

where the first equality holds as $d\Phi(t)^{-1}d\boldsymbol{x}_t \approx 0$. Integrating both sides from s to t gives

$$\boldsymbol{x}_t = \frac{\Phi(t)}{\Phi(s)}\boldsymbol{x}_s + \frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\tilde{B}(r)\boldsymbol{s}(\boldsymbol{x}_r, r)dr + \frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\sqrt{g_r}d\boldsymbol{w}_r \tag{35}$$

$$\approx \frac{\Phi(t)}{\Phi(s)}\boldsymbol{x}_s + \left[\frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\tilde{B}(r)dr\right]\boldsymbol{s}(\boldsymbol{x}_s, s) + \frac{\Phi(t)}{\Phi(s)}\int_s^t \frac{\Phi(s)}{\Phi(r)}\sqrt{g_r}d\boldsymbol{w}_r, \tag{36}$$

where we assume $\boldsymbol{s}(\boldsymbol{x}_r, r) \approx \boldsymbol{s}(\boldsymbol{x}_s, s)$, which corresponds to the *first-order exponential integrator* underlying the DPM-solver. Note that higher-order DPM-solvers use higher-order approximations here.

For conditional flow models, $A(t) := \dot{a}_t/a_t$ and $\Phi(t) := a_t$. Thus, the discretized SDE simplifies to:

$$\boldsymbol{x}_t = \frac{a_t}{a_s}\boldsymbol{x}_s + \left[\frac{a_t}{a_s}\int_s^t \frac{a_s}{a_r}\tilde{B}(r)dr\right]\boldsymbol{s}(\boldsymbol{x}_s, s) + a_t\int_s^t \frac{1}{a_r}\sqrt{g_r}d\boldsymbol{w}_r, \tag{37}$$

where the first term denotes a scaled previous sample, the second term denotes a score function, and the third term denotes the Gaussian noise with zero mean and covariance $\sigma_s^2\mathbf{I}$ where $\sigma_s^2 = a_t^2\int_s^t \frac{g_r}{a_r^2}dr$. Thus, setting $s = t_i$ and $t = t_{i-1}$ yields the general transition density:

$$p(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) = \mathcal{N}(\underbrace{\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}(\boldsymbol{x}_{t_i})}_{=:\boldsymbol{\mu}_{t_i}(\boldsymbol{x}_{t_i})}, \sigma_{t_i}^2\mathbf{I}). \tag{38}$$

□

B.3 Proof of Proposition 3.1

We begin with a simple lemma establishing the zero-mean score identity.

Lemma B.1 (Zero-mean score identity under tail regularity)**.** *Fix $\boldsymbol{x} \in \mathbb{R}^d$ and let $p(\cdot \mid \boldsymbol{x})$ be a conditional density on $\mathbb{R}^d$ (with respect to Lebesgue measure) such that:*

1. *$\int_{\mathbb{R}^d} p(\boldsymbol{u} \mid \boldsymbol{x})\, d\boldsymbol{u} = 1$;*
2. *$p(\cdot \mid \boldsymbol{x})$ is differentiable in $\boldsymbol{u}$, and $\int_{\mathbb{R}^d} \left|\partial_{u_j} p(\boldsymbol{u} \mid x)\right| d\boldsymbol{u} < \infty$ for $\forall j \in \{1, \ldots, d\}$;*
3. *For $\forall j$, for almost every $\boldsymbol{u}_{-j} \in \mathbb{R}^{d-1}$, $\lim_{\boldsymbol{u}_j \to \pm\infty} p(u_j, \boldsymbol{u}_{-j} \mid \boldsymbol{x}) = 0$.*

Then the conditional score has zero mean:

$$\mathbb{E}_{\boldsymbol{u}\sim p(\cdot|\boldsymbol{x})}\left[\nabla_{\boldsymbol{u}} \log p(\boldsymbol{u} \mid \boldsymbol{x})\right] = \mathbf{0}. \tag{39}$$

Proof. For each j,

$$\mathbb{E}_{p(\cdot|\boldsymbol{x})}[\partial_{u_j} \log p(\boldsymbol{u} \mid \boldsymbol{x})] = \int_{\mathbb{R}^d} p(\boldsymbol{u} \mid \boldsymbol{x})\, \frac{\partial_{u_j} p(\boldsymbol{u} \mid \boldsymbol{x})}{p(\boldsymbol{u} \mid \boldsymbol{x})}\, d\boldsymbol{u} = \int_{\mathbb{R}^d} \partial_{u_j} p(\boldsymbol{u} \mid \boldsymbol{x})\, d\boldsymbol{u},$$

where the equality is valid wherever $p(\boldsymbol{u} \mid \boldsymbol{x}) > 0$, and condition 2 justifies the integral. Writing $\boldsymbol{u} = (u_j, \boldsymbol{u}_{-j})$ and using Fubini's theorem, we obtain

$$\int_{\mathbb{R}^d} \partial_{u_j} p(\boldsymbol{u} \mid \boldsymbol{x})\, d\boldsymbol{u} = \int_{\mathbb{R}^{d-1}} \left(\int_{\mathbb{R}} \partial_{u_j} p(u_j, \boldsymbol{u}_{-j} \mid \boldsymbol{x})\, du_j\right) d\boldsymbol{u}_{-j}.$$

For fixed $\boldsymbol{u}_{-j}$, the inner integral is a 1D integral of a derivative, hence by the fundamental theorem of calculus,

$$\int_{\mathbb{R}} \partial_{u_j} p(u_j, \boldsymbol{u}_{-j} \mid \boldsymbol{x})\, du_j = \Big[p(u_j, \boldsymbol{u}_{-j} \mid \boldsymbol{x})\Big]_{u_j=-\infty}^{u_j=+\infty}.$$

By condition 3, this boundary term is 0 for almost every $\boldsymbol{u}_{-j}$, so the outer integral is 0. Since this holds for each j, we conclude Eq. (39). □

With this identity in place, we now prove the main result for the first-order score-matching kernel.

Proposition 3.1 (Optimal value guidance)**.** *Consider a first-order solver with transitional kernel:*

$$p^{\Psi}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \mathcal{N}\Big(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\big(\boldsymbol{s}^{ref}_{t_i}(\boldsymbol{x}_{t_i}) + \boldsymbol{\Psi}_{t_i}\big),\ \sigma^2_{t_i}\mathbf{I}\Big).$$

The optimal value guidance $\boldsymbol{\Psi}_{t_i}$ that matches the optimal transitional kernel $p^\star_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$ is

$$\boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}) \triangleq \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\, \mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot\mid\boldsymbol{x}_{t_i})}\Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\Big]. \tag{9}$$

Proof. First, we will prove the first part, i.e. the current-state characterization:

$$\boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \tag{40}$$

From Eq. (7), we derive the optimal Value Guidance Vector. Take the logarithm of both sides:

$$\log p^\star_{t_i}(\boldsymbol{x}_{t_i}) = \log p^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{1}{\alpha}V_{t_i}(\boldsymbol{x}_{t_i}) - \log Z. \tag{41}$$

Then, differentiate with respect to $\boldsymbol{x}$ at $t = t_i$:

$$\begin{aligned}\nabla_{\boldsymbol{x}_{t_i}} \log p^\star_{t_i}(\boldsymbol{x}_{t_i}) &= \nabla_{\boldsymbol{x}_{t_i}} \log p^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) \\ \boldsymbol{s}^\star_{t_i}(\boldsymbol{x}_{t_i}) &= \boldsymbol{s}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}).\end{aligned} \tag{42}$$

Finally, by comparing this to the definition $\boldsymbol{s}^\star_{t_i}(\boldsymbol{x}_{t_i}) = \boldsymbol{s}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i})$, we conclude that:

$$\boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}). \tag{43}$$

Next, we will express the optimal Value Guidance Vector in relation to $\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})$. Using Eq. (5), the *reference* one-step kernel at time t_i can be represented by

$$p^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \mathcal{N}\Big(\underbrace{\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i})}_{=:\,\boldsymbol{\mu}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i})},\ \sigma^2_{t_i}\mathbf{I}\Big),$$

Using the definition of Bellman equations and optimal policies [47], we obtain:

$$p^\star_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \frac{\exp\big(V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})/\alpha\big)\ p^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})}{\exp\big(V^\star_{t_i}(\boldsymbol{x}_{t_i})/\alpha\big)}. \tag{44}$$

Substituting the Gaussian reference kernel into the preceding identity gives

$$\log p^\star_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \frac{V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})}{\alpha} - \frac{1}{2\sigma^2_{t_i}}\big(\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i})\big)^\top\big(\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i})\big),$$

up to an additive constant independent of $\boldsymbol{x}_{t_{i-1}}$. Next, we differentiate w.r.t. $\boldsymbol{x}_{t_{i-1}}$:

$$\nabla_{\boldsymbol{x}_{t_{i-1}}} \log p^\star_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_{i-1}}} V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}) - \frac{1}{\sigma^2_{t_i}}\big(\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i})\big).$$

Taking the conditional expectation under $p^\star_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$ and using Lemma B.1, we obtain

$$\mathbf{0} = \frac{1}{\alpha}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot\mid\boldsymbol{x}_{t_i})}\Big[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\Big] - \frac{1}{\sigma^2_{t_i}}\Big(\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot\mid\boldsymbol{x}_{t_i})}[\boldsymbol{x}_{t_{i-1}}] - \boldsymbol{\mu}^{\text{ref}}_{t_i}(\boldsymbol{x}_{t_i})\Big).$$

Rearranging yields the conditional mean identity

$$\mathbb{E}_{p^\star_{t_i}}[\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}] = \boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \frac{\sigma^2_{t_i}}{\alpha}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot|\boldsymbol{x}_{t_i})}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\right]. \tag{45}$$

Matching the mean of $p^{\Psi}_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$, namely $\boldsymbol{\mu}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i}) + \Omega(t_i)\boldsymbol{\Psi}_{t_i}(\boldsymbol{x}_{t_i})$, with $\mathbb{E}_{p^\star_{t_i}}[\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}]$ yields:

$$\boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\,\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot|\boldsymbol{x}_{t_i})}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\right].$$

□

B.4 Proof of Proposition 3.6

Proposition 3.6 (Conditional Consistency of First-Order Estimators)**.** *Let $j \in \{i, i-1\}$. For*

$$\nabla_{\boldsymbol{x}_{t_j}} V_{t_j}(\boldsymbol{x}_{t_j}) = \gamma_{t_i}\nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j}), \tag{16}$$

define the residual-corrected first-order estimator:

$$\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) \triangleq \left(\frac{1}{\alpha}\iota[j=i] + \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\iota[j=i-1]\right)\left(\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_j}} r(\hat{\boldsymbol{x}}_{0|t_j}) + g_\phi(\boldsymbol{x}_{t_j})\right), \tag{17}$$

where $\iota[\cdot]$ denotes the indicator function, and for $j = i-1$ we take $\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}(\cdot \mid \boldsymbol{x}_{t_i})$, while for $j = i$ we simply set $\boldsymbol{x}_{t_j} = \boldsymbol{x}_{t_i}$. Then, $\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(j) \,\middle|\, \boldsymbol{x}_{t_i}\right]$ is an unbiased estimate of the value gradient.

Proof. For $j = i$, Eq. (17) reduces to

$$\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(i) = \frac{1}{\alpha}\left(\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}) + g_\phi(\boldsymbol{x}_{t_i})\right) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}(\boldsymbol{x}_{t_i}) = \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}), \tag{46}$$

where the second equality follows from Eq. (16), and the third from Eq. (9).

For $j = i-1$, by linearity of expectation and Eq. (16),

$$\begin{aligned}\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(i-1) \,\middle|\, \boldsymbol{x}_{t_i}\right] &= \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot|\boldsymbol{x}_{t_i})}\left[\gamma_{t_i}\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}}) + g_\phi(\boldsymbol{x}_{t_{i-1}})\right] \\ &= \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1}}\sim p^\star_{t_i}(\cdot|\boldsymbol{x}_{t_i})}\left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})\right].\end{aligned} \tag{47}$$

Applying Proposition 3.1 yields

$$\mathbb{E}\left[\hat{\boldsymbol{\Psi}}^{\mathrm{FO,res}}_{t_i}(i-1) \,\middle|\, \boldsymbol{x}_{t_i}\right] = \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}), \tag{48}$$

which proves the claim. □

Proposition 3.6 formalizes the idealized condition required by residual-corrected first-order methods: if g_ϕ exactly captures the Jensen gap introduced by the Tweedie-based approximation, then both current-state and one-step lookahead first-order guidance are unbiased. However, in practice, Appendix G.1 shows that the learned residual term is negligible, suggesting that this condition is not realized empirically. Thus, the Jensen gap $V_t(\boldsymbol{x}_t) - r(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)) = \mathbb{E}[r(\boldsymbol{x}_0)] - r(\mathbb{E}[\boldsymbol{x}_0])$ is not meaningfully closed.

Under exact residual correction, Eq. (16) recovers the intended $\mathcal{L}_{\text{forward}}$ of Residual ∇-DB. Choosing $g_\phi(\cdot) = \mathbf{0}$ yields SQDF. $\gamma_t = 1, g_\phi(\cdot) = \mathbf{0}$ returns to the case where we approximate the value function with only Tweedie's formula, as in DPS [6].

B.5 Proof of Theorem 3.7

We prove that the existing implementation of the KL-regularized REINFORCE estimator (derived in Section C.2) is an unbiased estimator of the gradient of the *hard* value function. We clarify that the following derivation aligns with the practical reality that existing implementations of Soft PPO and GRPO [10, 27] treat the KL penalty as additive intermediate rewards, rather than solving the exact log-sum-exp soft Bellman equations as suggested by Schulman et al. [42].

Definition B.2 (Value Function of Diffusion MDP)**.** Let $r : \mathbb{R}^d \to \mathbb{R}$ be the terminal reward at $t = t_0$. Given a transition function $f : \mathbb{R}^d \times \mathbb{Z}_{\geq 0} \to \mathbb{R}^d$ and a noise schedule $\sigma : \mathbb{Z}_{\geq 0} \to \mathbb{R}$, the definition of the value function $V : \mathbb{R}^d \times \mathbb{Z}_{\geq 0} \to \mathbb{R}$ under the expectation over the optimal policy is stated as:

$$V_{t_i}(\boldsymbol{x}_{t_i}) := \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}} \sim p^\star_{t_{i:1}}}[r(\boldsymbol{x}_{t_0})] \tag{49}$$

Theorem 3.7 (Consistency of Policy Gradient)**.** *For the full-rollout $j = 0$, the lookahead zeroth-order estimator of the existing reward-alignment methods is an unbiased estimator of the value gradient:*

$$\hat{\boldsymbol{\Psi}}^{\mathrm{LA},0}_{t_i} \approx \frac{\sigma_{t_i}}{\alpha\Omega(t_i)} \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}} \sim p^\star_{t_{i:1}}, \boldsymbol{\epsilon}_{t_{i:1}} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}] = \mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}}[\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)} \nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})]. \tag{18}$$

Proof. When $\boldsymbol{\epsilon}_{t_i}$ denotes the Gaussian noise injected in the transition process described in Eq. 5, the value gradient estimate is given as:

$$\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}} \left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}) \right] \tag{50}$$

$$= \mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}} \left[\nabla_{\boldsymbol{x}_{t_{i-1}}} \left(\mathbb{E}_{\boldsymbol{x}_{t_{i-2:0}} \sim p^\star_{t_{i-1:1}}, \boldsymbol{\epsilon}_{t_{i-1:1}} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})} [r(\boldsymbol{x}_{t_0})] \right) \right] \tag{51}$$

$$= \mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\frac{1}{\sigma_{t_i}} \nabla_{\boldsymbol{\epsilon}_{t_i}} \left(\mathbb{E}_{\boldsymbol{x}_{t_{i-2:0}} \sim p^\star_{t_{i-1:1}}, \boldsymbol{\epsilon}_{t_{i-1:1}} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})} [r(\boldsymbol{x}_{t_0})] \right) \right] \tag{52}$$

$$= \mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}, \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\frac{1}{\sigma_{t_i}} \left(\mathbb{E}_{\boldsymbol{x}_{t_{i-2:0}} \sim p^\star_{t_{i-1:1}}, \boldsymbol{\epsilon}_{t_{i-1:1}} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})} [r(\boldsymbol{x}_{t_0})\boldsymbol{\epsilon}_{t_i}] \right) \right]. \tag{53}$$

The last line comes from a special case of Stein's identity [44], which states that for a differentiable function h,

$$\mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})}[h(\boldsymbol{\epsilon})\boldsymbol{\epsilon}] = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I})}[\nabla_{\boldsymbol{\epsilon}} h(\boldsymbol{\epsilon})]. \tag{54}$$

Multiplying Eqs. (50), (53) by $\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}$ completes the proof:

$$\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{t_i}}[\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)} \nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}})] = \frac{\sigma_{t_i}}{\alpha\Omega(t_i)} \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}} \sim p^\star_{t_{i:1}}, \boldsymbol{\epsilon}_{t_{i:1}} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}]. \tag{55}$$

□

B.6 Hard versus Soft Value Gradients in Practical Implementations

Sections B.4 and B.5 show that full-rollout estimators can provide unbiased estimates of a value gradient. The relevant point, however, is which value function is being differentiated. In the main text, we show that the existing diffusion fine-tuning methods are unified to admit $\hat{\boldsymbol{\Psi}}_{t_i}$ which denotes an estimator of the gradient of the *hard* value function, namely the expected unregularized terminal reward

$$V^{\mathrm{hard}}_{t_i}(\boldsymbol{x}_{t_i}) := \mathbb{E}_{p^\star}[r(\boldsymbol{x}_{t_0}) \mid \boldsymbol{x}_{t_i}]\,. \tag{56}$$

To get this value-gradient surrogate of many practical diffusion fine-tuning methods, the rollout estimator uses the terminal reward $r(\boldsymbol{x}_{t_0})$, while the KL regularization against the reference model is introduced as a separate timestep-wise penalty or score correction.

However, this convention differs from the fully entropy-regularized, or *soft*, value function from soft RL. For a given policy $p_{t_{i:1}}$, define the entropy-augmented return from $\boldsymbol{x}_{t_i}$ by

$$\tilde{r}_{t_i} := r(\boldsymbol{x}_{t_0}) - \alpha \sum_{k=1}^{i} D_{\mathrm{KL}}\big(p_{t_k}(\cdot \mid \boldsymbol{x}_{t_k}) \| p^{\mathrm{ref}}_{t_k}(\cdot \mid \boldsymbol{x}_{t_k})\big)\,. \tag{57}$$

The corresponding soft value function is

$$V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}) := \max_{p_{t_{i:1}}} \mathbb{E}[\tilde{r}_{t_i} \mid \boldsymbol{x}_{t_i}], \tag{58}$$

where the expectation is taken over the reverse trajectory induced by $p_{t_{i:1}}$. Equivalently, if $p^\star_{t_{i:1}}$ denotes an optimizer of the entropy-regularized objective, then

$$V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}) = \mathbb{E}_{p^\star_{t_{i:1}}}[\tilde{r}_{t_i} \mid \boldsymbol{x}_{t_i}]. \tag{59}$$

This is the diffusion analogue of the entropy-augmented value function in soft RL, where the return itself contains the KL costs rather than adding only an instantaneous KL-gradient term [42]. Under the same regularity assumptions used in Appendix B.5, applying Stein's identity to the entropy-augmented return yields the soft-value analogue of the full-rollout estimator (see Appendix B.6.1):

$$\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p^\star_{t_{i:1}},\boldsymbol{\epsilon}_{t_{i:1}}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\tilde{r}_{t_i}\boldsymbol{\epsilon}_{t_i} \mid \boldsymbol{x}_{t_i}] = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}) =: \boldsymbol{\Psi}^\star_{t_i}(\boldsymbol{x}_{t_i}). \tag{60}$$

Thus, a fully soft-value-consistent zeroth-order method would replace the terminal reward $r(\boldsymbol{x}_{t_0})$ in the rollout estimator by the entropy-augmented return $\tilde{r}_{t_i}$.

It can be shown that Policy gradient methods using the entropy-augmented reward $\tilde{r}_{t_i}$ admit an unbiased zeroth-order estimator for the soft value gradient in Eq. (60), equivalent to that of soft Q-learning methods. On the other hand, estimators that add only a local KL-gradient term, which is a widely-used surrogate of many practical diffusion fine-tuning methods, are not identical to estimators in Eq. (60) because they do not account for how earlier actions affect future KL costs[42]. More formally, the surrogate of existing implementations unified by RSM corresponds to the form used in many existing diffusion fine-tuning implementations:

$$-\frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}^{\mathrm{hard}}(\boldsymbol{x}_{t_i}) + \left(\boldsymbol{s}^\star_{t_i}(\boldsymbol{x}_{t_i}) - \boldsymbol{s}^{\mathrm{ref}}_{t_i}(\boldsymbol{x}_{t_i})\right). \tag{61}$$

This expression is closely related to, but not algebraically identical to,

$$-\frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}). \tag{62}$$

The difference is not a failure of the RSM reduction; rather, it specifies the exact surrogate that the reduction describes. Since our goal is to unify the objectives used by *existing* fine-tuning methods (see Table 1), the main text focuses on the hard-value estimator with a separately added KL correction. A fully soft-value-consistent variant can be obtained by using $\tilde{r}_{t_i}$ inside the value-gradient estimator, as in Eq. (60). We leave a systematic empirical study of when this distinction matters to future work.

B.6.1 Soft-Value Estimator

For completeness, we now formalize the soft value gradient estimator. This is a direct analogue of Theorem 3.7, with the terminal reward replaced by the entropy-augmented return.

Theorem B.3 (Soft-value analogue of the zeroth-order estimator)**.** *Assume the regularity conditions of Theorem 3.7, and assume that the optimizer $p^\star_{t_{i:1}}$ of the entropy-regularized objective satisfies the standard envelope-theorem conditions. Then*

$$\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p^\star_{t_{i:1}},\boldsymbol{\epsilon}_{t_{i:1}}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\tilde{r}_{t_i}\boldsymbol{\epsilon}_{t_i} \mid \boldsymbol{x}_{t_i}] = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}). \tag{63}$$

Proof. By definition,

$$V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}) = \max_{p_{t_{i:1}}} \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p_{t_{i:1}}}[\tilde{r}_{t_i} \mid \boldsymbol{x}_{t_i}]. \tag{64}$$

Let $p^\star_{t_{i:1}}$ be an optimizer. By the envelope theorem [32], the derivative of the optimized value with respect to $\boldsymbol{x}_{t_i}$ is obtained by differentiating the objective at $p^\star_{t_{i:1}}$, without including the derivative of the optimizer itself. Hence,

$$\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}) = \nabla_{\boldsymbol{x}_{t_i}} \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}}\sim p^\star_{t_{i:1}}}[\tilde{r}_{t_i} \mid \boldsymbol{x}_{t_i}]. \tag{65}$$

The remaining argument is identical to the derivation of Theorem 3.7 from Eq. (51) to Eq. (53), using Stein's identity as a key bridge. In particular, under the one-step Gaussian perturbation used

there, differentiating the conditional expectation with respect to $\boldsymbol{x}_{t_i}$ can be written as a Gaussian score-function identity, giving

$$\nabla_{\boldsymbol{x}_{t_i}} V_{t_i}^{\mathrm{soft}}(\boldsymbol{x}_{t_i}) = \frac{\sigma_{t_i}}{\Omega(t_i)} \mathbb{E}_{p^\star_{t_{i:1}}, \boldsymbol{\epsilon}_{t_{i:1}} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} [\tilde{r}_{t_i} \boldsymbol{\epsilon}_{t_i} \mid \boldsymbol{x}_{t_i}] \,. \tag{66}$$

Multiplying both sides by $1/\alpha$ gives the claim. □

B.7 Convergence of Policy Iteration

Section 3 and Appendix B characterize the reward-guided target and its associated optimal guidance as fixed-point identities of the optimum $p^\star$. In practice, however, RL-based fine-tuning methods sample from the current policy p^θ and update that policy using the RSM loss. We therefore rely on prior policy optimization theory for convergence guarantees. TRPO [41] was originally derived from a monotonic improvement perspective, and subsequent theory established that exact TRPO converges to the optimal policy in entropy-regularized MDPs [33]. More recent results established global convergence guarantees for PPO-Clip and its variants under neural function approximation [18]. Accordingly, our core theory specifies the fixed point that the reward score matching update is designed to target, while convergence of the resulting iterative optimization procedure is justified by prior works rather than proven in this paper.

C Proof of Theorem 3.2

Theorem 3.2 (Reward Score Matching)**.** *The methods summarized in Table 1, including Soft RL methods, VGG-Flow, DRaFT, DRaFT-K, ReFL, and Adjoint Matching, admit the common loss*

$$\mathcal{L}(\theta) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}} \Big[C_1(t_i) \Big(\| \boldsymbol{s}^\theta_{t_i} - (\boldsymbol{s}^{ref}_{t_i} + \boldsymbol{\Psi}_{t_i}) \|^2 + C_2(t_i) \| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i} \|^2 \Big) \Big] \tag{10}$$

where C_1, C_2 are weights, and $\boldsymbol{s}^\theta, \boldsymbol{s}^{\theta^\dagger}, \boldsymbol{s}^{ref}$ are the current, previous, and reference score functions [17].

Proof. The proof for each method is laid throughout Appendix C. □

Technical remark ($\alpha = 0$)**.** SQDF and policy-gradient methods are well-defined even when $\alpha = 0$. $\boldsymbol{\Psi}_{t_i}$ and $C_2(t_i)$ appear singular due to $1/\alpha$, but these quantities appear only through products $C_1(t_i)\boldsymbol{\Psi}_{t_i}$ and $C_1(t_i)C_2(t_i)$ in Eq. (12). The factor of α in $C_1(t_i)$ cancels the apparent singularity. Therefore, when $\alpha = 0$, the canonical gradient remains the correct object for interpreting optimization dynamics, although the literal decomposition from Eq. (8) no longer strictly applies.

C.1 First-order Methods

C.1.1 Residual ∇-DB

The relationship between Residual ∇-DB and Soft RL has been briefly discussed in Appendix D of Liu et al. [30]. We rewrite the objective by mapping forward and backward GFlowNet processes (P_F, P_B) to reverse sampling and forward noising, respectively.

Residual ∇-DB minimizes a weighted sum of four terms:

$$\mathcal{L}_{\text{total}}(\theta, \phi) = w_F \mathcal{L}_{\text{forward}}(\theta, \phi) + w_B \mathcal{L}_{\text{backward}}(\theta, \phi) + \mathcal{L}_{\text{terminal}}(\phi) + w_R \mathcal{L}_{\text{reg}}(\theta), \tag{67}$$

$$\mathcal{L}_{\text{forward}}(\theta, \phi) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[\left\| \nabla_{\boldsymbol{x}_{t_{i-1}}} \ell^\theta_{t_i}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{x}_{t_i}) - \mathcal{G}_{\text{GFN_F}}(\boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}) \right\|^2 \right], \tag{68}$$

$$\mathcal{L}_{\text{backward}}(\theta, \phi) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[\left\| \nabla_{\boldsymbol{x}_{t_i}} \ell^\theta_{t_i}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{x}_{t_i}) + \mathcal{G}_{\text{GFN_B}}(\boldsymbol{x}_{t_i}) \right\|^2 \right], \tag{69}$$

$$\mathcal{L}_{\text{terminal}}(\phi) = \mathbb{E}_{\boldsymbol{x}_0} [\| g_\phi(\boldsymbol{x}_0) \|^2], \tag{70}$$

$$\mathcal{L}_{\text{reg}}(\theta) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}} [\| \boldsymbol{\epsilon}^\theta_{t_i} - \boldsymbol{\epsilon}^{\theta^\dagger}_{t_i} \|^2] = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}} [(1 - \bar{\alpha}_{t_i}) \| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i} \|^2], \tag{71}$$

[17] When $C_2(t) = 0$, Eq. (10) reduces to entropy-regularized optimal control (Appendix H.1).

where ℓ, gradient fields $\mathcal{G}$ and Gaussian reparameterization are:

$$\ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{x}_{t_i}) := \log p^{\theta}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) - \log p^{\text{ref}}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}), \tag{72}$$

$$\mathcal{G}_{\text{GFN_F}}(\boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}) = \frac{1}{\alpha}\tilde{\gamma}_{t_i}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})) + g_{\phi}(\boldsymbol{x}_{t_{i-1}}), \tag{73}$$

$$\mathcal{G}_{\text{GFN_B}}(\boldsymbol{x}_{t_i}) = \frac{1}{\alpha}\tilde{\gamma}_{t_i}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i})) + g_{\phi}(\boldsymbol{x}_{t_i}), \tag{74}$$

$$\boldsymbol{x}_{t_{i-1}} = \boldsymbol{\mu}^{\theta}_{t_i} + \sigma_{t_i}\boldsymbol{\epsilon}_{t_i}, \quad \boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Pruning $\mathcal{L}_{\text{backward}}$ and $\mathcal{L}_{\text{terminal}}$. $\mathcal{L}_{\text{backward}}$ requires the gradient $\nabla_{\boldsymbol{x}_{t_i}} \ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{x}_{t_i})$. Under the Gaussian reparameterization, $\boldsymbol{x}_{t_{i-1}}$ depends on $\boldsymbol{x}_{t_i}$ through the model output, which introduces Jacobian terms $\partial\boldsymbol{\epsilon}^{\theta}_{t_i}/\partial\boldsymbol{x}_{t_i}$ into $\nabla_{\boldsymbol{x}_{t_i}}\ell^{\theta}_{t_i}$. Consequently, $\nabla_{\theta}\mathcal{L}_{\text{backward}}$ involves higher-order derivatives, including mixed θ-$\boldsymbol{x}$ terms, rendering optimization numerically unstable and empirically ineffective. This is consistent with our ablation results in Appendix G.1; removing $\mathcal{L}_{\text{backward}}$ does not degrade performance, whereas removing $\mathcal{L}_{\text{forward}}$ causes training to fail.

$\mathcal{L}_{\text{terminal}}$ (Eq. (70)) is not parameterized by θ and therefore contributes no gradient to the canonical loss term in Theorem 3.2. Moreover, as discussed in Appendix G.1, the outputs of g_{ϕ} are negligible in magnitude and do not affect training dynamics.

Reduction to Reward Score Matching. $\nabla_{\boldsymbol{x}_{t_{i-1}}}\ell^{\theta}_{t_i}$ can be expressed as a scaled score difference:

$$\nabla_{\boldsymbol{x}_{t_{i-1}}}\ell^{\theta}_{t_i}(\boldsymbol{x}_{t_{i-1}}, \boldsymbol{x}_{t_i}) = \frac{1}{\sigma^2_{t_i}}(\boldsymbol{\mu}^{\theta}_{t_i} - \boldsymbol{\mu}^{\text{ref}}_{t_i}) = \frac{\Omega(t_i)}{\sigma^2_{t_i}}(\boldsymbol{s}^{\theta}_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}). \tag{75}$$

$$\therefore \mathcal{L}_{\text{forward}}(\theta, \phi) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}}\left[\frac{\Omega(t_i)^2}{\sigma^4_{t_i}}\left\|\boldsymbol{s}^{\theta}_{t_i} - \left(\boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\sigma^2_{t_i}}{\Omega(t_i)}\,\mathcal{G}_{\text{GFN_F}}(\boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i})\right)\right\|^2\right]. \tag{76}$$

Thus, $w_F\mathcal{L}_{\text{forward}} + w_R\mathcal{L}_{\text{reg}}$ is a special case of the canonical loss, with

$$\boldsymbol{\Psi}_{t_i} = \tilde{\gamma}_{t_i}\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\,\text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})), \quad C_1(t_i) = \frac{1}{d}\frac{\Omega(t_i)^2}{\sigma^4_{t_i}}, \quad C_2(t_i) = \frac{(1-\bar{\alpha}_{t_i})\sigma^4_{t_i}}{\Omega(t_i)^2}\frac{w_R}{w_F}, \tag{77}$$

where the factor $\frac{1}{d}$ arises implicitly from a `torch.mean()` operation over tensor dimensions.

C.1.2 SQDF

SQDF is motivated by the following objective, given in Eq. (11) of Kang et al. [21]:

$$\mathcal{L}(\theta) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}}\big[-r(\hat{\boldsymbol{x}}_{0|t_{i-1}}) + \alpha\mathcal{D}_{\text{KL}}(p^{\theta}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}) \| p^{\text{ref}}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i}))\big]. \tag{78}$$

In the actual implementation, however, the transition mean $\boldsymbol{\mu}^{\theta}_{t_{i-1}}$ is reparameterized, and a stopgrad operation is applied to the Tweedie estimate inside the reward term. In addition, an attenuation factor $\tilde{\gamma}_{t_i} = \tilde{\gamma}^{Nt_i}_{\text{SQDF}}$ is introduced in the reward term for temporal credit assignment. This yields:

$$\mathcal{L}(\theta) = \mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}}\left[-\tilde{\gamma}_{t_i}\ \text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}}))\cdot\boldsymbol{\mu}^{\theta}_{t_{i-1}} + \frac{\alpha}{2\sigma^2_{t_i}}\left\|\boldsymbol{\mu}^{\theta}_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}}\right\|^2\right]. \tag{79}$$

Substituting $\boldsymbol{\mu}^{\theta}_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}} = \Omega(t_i)(\boldsymbol{s}^{\theta}_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i})$ yields

$$\mathbb{E}\left[-\tilde{\gamma}_{t_i}\ \text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}}))\cdot\left(\kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^{\theta}_{t_i}\right) + \frac{\alpha\Omega(t_i)^2}{2\sigma^2_{t_i}}\left\|\boldsymbol{s}^{\theta}_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}\right\|^2\right]. \tag{80}$$

Up to additive terms that do not contribute gradients, this is equivalent to:

$$\mathbb{E}_{t_i, \boldsymbol{x}_{t_i}, \boldsymbol{\epsilon}_{t_i}}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma^2_{t_i}}\left\|\boldsymbol{s}^{\theta}_{t_i} - \left(\boldsymbol{s}^{\text{ref}}_{t_i} + \tilde{\gamma}_{t_i}\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\ \text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}}))\right)\right\|^2\right]. \tag{81}$$

Therefore, SQDF is a special case of the canonical loss, with

$$\boldsymbol{\Psi}_{t_i} = \tilde{\gamma}_{t_i}\frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\ \text{sg}(\nabla_{\boldsymbol{x}_{t_{i-1}}} r(\hat{\boldsymbol{x}}_{0|t_{i-1}})), \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma^2_{t_i}}, \quad C_2(t_i) = 0. \tag{82}$$

C.1.3 VGG-Flow

We rewrite VGG-Flow objectives, simplified using $G^\phi(\boldsymbol{x}_{t_i}) \triangleq -\tilde{\gamma}_{t_i}\,\mathrm{sg}\big(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i})\big) - g^\phi(\boldsymbol{x}_{t_i})$.

$$\mathcal{L}_{\text{consistency}}(\phi) = \mathbb{E}_{t_i,\boldsymbol{x}_{t_i}}\Big\| \frac{\partial}{\partial t} G^\phi(\boldsymbol{x}_t)\Big|_{t=t_i} + [\nabla G^\phi(\boldsymbol{x}_{t_i})]^\top (\boldsymbol{v}^{\text{ref}}_{t_i} + \frac{1}{\lambda} G^\phi(\boldsymbol{x}_{t_i})) + [\nabla_{\boldsymbol{x}_{t_i}} \boldsymbol{v}^{\text{ref}}_{t_i}]^\top G^\phi(\boldsymbol{x}_{t_i})\Big\|^2, \tag{83}$$

$$\mathcal{L}_{\text{boundary}}(\phi) = \mathbb{E}_{\boldsymbol{x}_0}\|g^\phi(\boldsymbol{x}_0)\|^2, \tag{84}$$

$$\mathcal{L}_{\text{matching}}(\theta) = \mathbb{E}_{\boldsymbol{x}_{t_i}}\Big[\|\boldsymbol{v}^\theta_{t_i} - \boldsymbol{v}^{\text{ref}}_{t_i} - \frac{1}{\alpha}[-\tilde{\gamma}_{t_i}\,\mathrm{sg}((\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i})) - g^\phi(\boldsymbol{x}_{t_i})]\|^2\Big]. \tag{85}$$

Here, $\mathcal{L}_{\text{total}}(\theta,\phi) = \mathcal{L}_{\text{matching}}(\theta) + \mathcal{L}_{\text{consistency}}(\phi) + w_b\mathcal{L}_{\text{boundary}}(\phi)$.[18]

As with Residual ∇-DB, and as discussed in the main text, $\mathcal{L}_{\text{boundary}}$ effectively enforces $g^\phi(\boldsymbol{x}_{t_i}) \approx \mathbf{0}$ for all t_i. After dropping g^ϕ, we analyze the dominant term, $\mathcal{L}_{\text{matching}}(\theta)$, by mapping the velocity field $\boldsymbol{v}_{t_i}$ to our unified score parameterization $\boldsymbol{s}_{t_i}$ under the general SDE formulation, specifically using the $\boldsymbol{s}_{t_i}$ form without $\Omega(t_i)$.

Using $\delta(t_i)$ for $\boldsymbol{v}$-parameterized Rectified Flow (Eq. (165)), $\mathcal{L}_{\text{matching}}$ can be written as:

$$\begin{aligned}\mathcal{L}_{\text{matching}}(\theta) &= \mathbb{E}_{\boldsymbol{x}_{t_i}}\Big[\Big\|\frac{1}{\delta(t_i)}(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}) - \frac{1}{\alpha}\tilde{\gamma}_{t_i}\,\mathrm{sg}(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i}))\Big\|^2\Big] \\ &= \mathbb{E}_{\boldsymbol{x}_{t_i}}\Big[\frac{1}{\delta(t_i)^2}\Big\|\boldsymbol{s}^\theta_{t_i} - \Big(\boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\delta(t_i)}{\alpha}\tilde{\gamma}_{t_i}\,\mathrm{sg}(\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i}))\Big)\Big\|^2\Big].\end{aligned} \tag{86}$$

Therefore, VGG-Flow is a special case of the canonical loss, with

$$\boldsymbol{\Psi}_{t_i} = \frac{\tilde{\gamma}_{t_i}}{\alpha}\delta(t_i)\;\mathrm{sg}((\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}^\theta_{0|t_i})), \quad C_1(t_i) = \frac{1}{d}\frac{1}{\delta(t_i)^2}, \quad C_2(t_i) = 0, \tag{87}$$

where the factor $\frac{1}{d}$ arises implicitly from a `torch.mean()` operation over tensor dimensions, rather than from an explicit scalar normalization.

C.1.4 DRaFT, DRaFT-K

DRaFT [7] directly differentiates the terminal reward through the denoising trajectory. Consider its objective with the optional additive KL regularization described in Appendix B.8 of Clark et al. [7]. Using the canonical-gradient representation above, we have

$$\mathcal{G}(\boldsymbol{x}_{t_i}) = \underbrace{\frac{\alpha}{\delta(t_i)^2}}_{C_1(t_i)}\left(\underbrace{-\frac{\delta(t_i)^2}{2\alpha}\nabla_{\boldsymbol{s}^\theta_{t_i}} r(\boldsymbol{x}_0)}_{\boldsymbol{\Psi}_{t_i}} + (\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i})\right), \tag{88}$$

with $C_2(t_i) = 0$.

Define the reward gradient propagated through the remaining denoising trajectory as

$$\lambda_{t_{i-1}} \triangleq \nabla_{\boldsymbol{x}_{t_{i-1}}} r(\boldsymbol{x}_0). \tag{89}$$

Since the unified transition parameterization gives

$$\frac{\partial \boldsymbol{x}_{t_{i-1}}}{\partial \boldsymbol{s}^\theta_{t_i}} = \Omega(t_i)\mathbf{I}, \tag{90}$$

we obtain

$$\nabla_{\boldsymbol{s}^\theta_{t_i}} r(\boldsymbol{x}_0) = \Omega(t_i)\lambda_{t_{i-1}}, \tag{91}$$

and hence

$$\boldsymbol{\Psi}_{t_i} = \frac{\delta(t_i)^2\Omega(t_i)}{2\alpha}\lambda_{t_{i-1}}. \tag{92}$$

[18] The original paper and our manuscript use opposite time conventions ($0 \to 1$ vs. $1 \to 0$). To match our notation, we flip the signs highlighted in red relative to the original formulation.

DRaFT differentiates through every subsequent denoising step, so $\lambda_{t_{i-1}}$ is the full-lookahead reward gradient. Using Definition 3.5,

$$\hat{\mathbf{\Psi}}^{\mathrm{LA},1}_{t_i}(0,1) = \frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)}\lambda_{t_{i-1}}, \tag{93}$$

and Eq. (92) can therefore be written as

$$\mathbf{\Psi}_{t_i} = \underbrace{\frac{\delta(t_i)^2\Omega(t_i)^2}{2\sigma_{t_i}^2}}_{\gamma(t_i)}\hat{\mathbf{\Psi}}^{\mathrm{LA},1}_{t_i}(0,1). \tag{94}$$

Thus, DRaFT is a special case of RSM with

$$\hat{\mathbf{\Psi}}_{t_i} = \hat{\mathbf{\Psi}}^{\mathrm{LA},1}_{t_i}(0,1), \qquad \gamma(t_i) = \frac{\delta(t_i)^2\Omega(t_i)^2}{2\sigma_{t_i}^2}, \qquad C_1(t_i) = \frac{\alpha}{\delta(t_i)^2}, \qquad C_2(t_i) = 0. \tag{95}$$

For deterministic DDIM sampling, $\sigma_{t_i} = 0$, so $\gamma(t_i)$ and $\hat{\mathbf{\Psi}}^{\mathrm{LA},1}_{t_i}(0,1)$ should not be interpreted separately; their product in Eq. (92) remains well-defined. This is analogous to the $\alpha = 0$ interpretation discussed at the beginning of Appendix C.

DRaFT-K truncates the Jacobian chain by applying $\boldsymbol{x}_{t_K} \leftarrow \mathrm{stopgrad}(\boldsymbol{x}_{t_K})$ and differentiating only through the final K denoising steps. Under our time convention, it therefore coincides with DRaFT for $1 \le i \le K$ and has zero reward guidance for $i > K$:

$$\mathbf{\Psi}^{\text{DRaFT-}K}_{t_i} = \mathbf{1}[i \le K]\,\gamma(t_i)\hat{\mathbf{\Psi}}^{\mathrm{LA},1}_{t_i}(0,1). \tag{96}$$

Here, K denotes the backpropagation horizon of DRaFT-K and is distinct from the branching width K_i in Definition 3.5. DRaFT provides an unbiased full-lookahead first-order estimator, but requires backpropagation through the full Jacobian chain and consequently incurs prohibitive memory cost at scale; DRaFT-K trades this consistency for reduced memory usage.

C.1.5 ReFL

ReFL [55] directly differentiates the reward through a current-state Tweedie prediction. For VP-SDE, ReFL samples an intermediate timestep t_i and computes

$$\hat{\boldsymbol{x}}_{0|t_i} = \frac{\mathrm{stopgrad}(\boldsymbol{x}_{t_i}) - \frac{1}{\delta(t_i)}\boldsymbol{\epsilon}^\theta_{t_i}}{a_{t_i}} = \frac{\mathrm{stopgrad}(\boldsymbol{x}_{t_i}) + \frac{1}{\delta(t_i)^2}\boldsymbol{s}^\theta_{t_i}}{a_{t_i}}, \qquad a_{t_i} = \sqrt{\bar{\alpha}_{t_i}}. \tag{97}$$

The stop-gradient operation removes the dependence of the reward on $\boldsymbol{x}_{t_i}$, yielding

$$\frac{\partial\hat{\boldsymbol{x}}_{0|t_i}}{\partial\boldsymbol{s}^\theta_{t_i}} = \frac{1}{a_{t_i}\delta(t_i)^2}\mathbf{I}. \tag{98}$$

Therefore,

$$\nabla_{\boldsymbol{s}^\theta_{t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}) = \frac{1}{a_{t_i}\delta(t_i)^2}\nabla_{\hat{\boldsymbol{x}}_{0|t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}). \tag{99}$$

Substituting Eq. (99) into the canonical gradient gives

$$\mathcal{G}(\boldsymbol{x}_{t_i}) = \underbrace{\frac{\alpha}{\delta(t_i)^2}}_{C_1(t_i)}\left(-\frac{1}{2\alpha a_{t_i}}\nabla_{\hat{\boldsymbol{x}}_{0|t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}) + (\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\mathrm{ref}}_{t_i})\right). \tag{100}$$

Using the current-state pullback notation of Definition 3.4,

$$\hat{\mathbf{\Psi}}^{\mathrm{CS},1}_{t_i}[\mathbf{I}] = \frac{1}{\alpha}\nabla_{\hat{\boldsymbol{x}}_{0|t_i}} r(\hat{\boldsymbol{x}}_{0|t_i}), \tag{101}$$

so the effective guidance in Eq. (100) is

$$\mathbf{\Psi}_{t_i} = \underbrace{\frac{1}{2a_{t_i}}}_{\gamma(t_i)}\hat{\mathbf{\Psi}}^{\mathrm{CS},1}_{t_i}[\mathbf{I}]. \tag{102}$$

Thus, ReFL is a special case of RSM with

$$\hat{\mathbf{\Psi}}_{t_i} = \hat{\mathbf{\Psi}}^{\mathrm{CS},1}_{t_i}[\mathbf{I}], \qquad \gamma(t_i) = \frac{1}{2\sqrt{\bar{\alpha}_{t_i}}}, \qquad C_1(t_i) = \frac{\alpha}{\delta(t_i)^2}, \qquad C_2(t_i) = 0. \tag{103}$$

ReFL therefore belongs to the same current-state first-order family as VGG-Flow, while differing in its local pullback: VGG-Flow uses the full Tweedie Jacobian $\mathbf{J}_{\mathrm{Tw}}$, whereas ReFL uses the identity pullback induced by its stop-gradient construction, with the remaining scalar Jacobian factor absorbed into $\gamma(t_i)$.

C.1.6 Adjoint Matching

Adjoint Matching [8] performs reward fine-tuning of diffusion and flow-matching models with entropy-regularized stochastic optimal control, where the optimal control term is obtained by regressing onto an adjoint state computed from a backward ODE. Specifically, given the following reverse-time SDE formulation of the reference model's generation process,

$$d\boldsymbol{x}_t = b^{\mathrm{ref}}_t(\boldsymbol{x}_t)dt + \tilde{\sigma}_t dw_t, \quad \boldsymbol{x}_1 \sim p_1, \quad t: 1 \to 0, \tag{104}$$

Adjoint Matching aims to find the optimal control term $u_t(\boldsymbol{x}_t, t)$ that minimizes the cost functional of the following SOC problem.

$$\min_u \mathbb{E}\left[\int_0^1 \left(\frac{\alpha}{2}\left\|u_t(\boldsymbol{x}_t)\right\|^2 - \rho_t(\boldsymbol{x}_t)\right) dt - r(\boldsymbol{x}_0)\right] \tag{105}$$

$$\text{s.t.} \quad d\boldsymbol{x}_t = b^{\theta}_t(\boldsymbol{x}_t)dt + \tilde{\sigma}_t dw_t := \left(b^{\mathrm{ref}}_t(\boldsymbol{x}_t) - \tilde{\sigma}_t u_t(\boldsymbol{x}_t)\right) dt + \tilde{\sigma}_t dw_t, \quad \boldsymbol{x}_1 \sim p_1, \quad t: 1 \to 0$$

Here, $\rho_t(\boldsymbol{x}_t)$ and $r(\boldsymbol{x}_0)$ denote intermediate and terminal reward, respectively.

For a trajectory τ sampled from SDE controlled by u, adjoint state $\boldsymbol{a}_t(\tau, u)$ is defined as the pathwise derivative of the realized future cost with respect to the state $\boldsymbol{x}_t$. It can be computed backward along the generative trajectory through the following adjoint ODE:

$$\frac{d\boldsymbol{a}_t}{dt} = -\boldsymbol{a}_t^{\top}\nabla_{\boldsymbol{x}_t}\left(b^{\mathrm{ref}}_t(\boldsymbol{x}_t) - \tilde{\sigma}_t u_t(\boldsymbol{x}_t)\right) + \nabla_{\boldsymbol{x}_t}\left(-\rho_t(\boldsymbol{x}_t) + \frac{\alpha}{2}\left\|u_t(\boldsymbol{x}_t)\right\|^2\right) \tag{106}$$

$$\boldsymbol{a}_0 = -\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0). \tag{107}$$

Adjoint Matching describes how $\boldsymbol{a}_t$ obtained from different trajectories on $\boldsymbol{x}_t$ can be a regression target for the difference between the reference and fine-tuned model on $\boldsymbol{x}_t$. Based on the Hamilton-Jacobi-Bellman equation, Eq. (34) of [8] states the following objective;

$$\mathcal{L}_{\mathrm{AM}}(\theta) = \mathbb{E}_{\tau\sim p^u,t}\left[\left\|u_t(\boldsymbol{x}_t) + \frac{1}{\alpha}\tilde{\sigma}_t \boldsymbol{a}_t(\tau, \mathrm{sg}(u))\right\|^2\right]. \tag{108}$$

Substituting the linear correlation between $u_t(\boldsymbol{x}_t)$, $b_t(\boldsymbol{x}_t)$, and $\boldsymbol{s}^{\theta}_t(\boldsymbol{x}_t)$ into Eq. (108) yields

$$\mathcal{L}_{\mathrm{AM}}(\theta) = \mathbb{E}_{\tau\sim p^u,t}\left[\left\|-\frac{1}{\tilde{\sigma}_t}(b^{\theta}_t(\boldsymbol{x}_t) - b^{\mathrm{ref}}_t(\boldsymbol{x}_t)) + \frac{1}{\alpha}\tilde{\sigma}_t\boldsymbol{a}_t\right\|^2\right] \tag{109}$$

$$= \mathbb{E}_{\tau\sim p^u,t}\left[\left\|\left(\frac{\tilde{\sigma}_t}{2} + \frac{1}{\tilde{\sigma}_t\delta(t)}\right)(\boldsymbol{s}^{\theta}_t(\boldsymbol{x}_t) - \boldsymbol{s}^{\mathrm{ref}}_t(\boldsymbol{x}_t)) + \frac{1}{\alpha}\tilde{\sigma}_t\boldsymbol{a}_t\right\|^2\right] \tag{110}$$

$$= \mathbb{E}_{\tau\sim p^u,t}\left[\left(\frac{\tilde{\sigma}_t}{2} + \frac{1}{\tilde{\sigma}_t\delta(t)}\right)^2\left\|\boldsymbol{s}^{\theta}_t(\boldsymbol{x}_t) - \left(\boldsymbol{s}^{\mathrm{ref}}_t(\boldsymbol{x}_t) - \frac{1}{\alpha}\frac{2\tilde{\sigma}_t^2\delta(t)}{\tilde{\sigma}_t^2\delta(t)+2}\boldsymbol{a}_t\right)\right\|^2\right], \tag{111}$$

where $\delta(t)$ represents the coefficient relating the score difference to the velocity difference, as defined in Appendix D.2. Thus, Adjoint Matching is an instance of the RSM objective in Eq. (10), with

$$\mathbf{\Psi}_t = -\frac{1}{\alpha}\frac{2\tilde{\sigma}_t^2\delta(t)}{\tilde{\sigma}_t^2\delta(t)+2}\boldsymbol{a}_t, \quad \gamma(t) = 1, \quad C_1(t) = \left(\frac{\tilde{\sigma}_t}{2} + \frac{1}{\tilde{\sigma}_t\delta(t)}\right)^2, \quad C_2(t) = 0. \tag{112}$$

Meanwhile, when we define cost-to-go functional $J(u; \boldsymbol{x}_t, t)$ of $\boldsymbol{x}_t$ as the expected future cost under a fixed control u,

$$J(u; \boldsymbol{x}_t, t) = \mathbb{E}_{\tau \sim p^u}\left[\int_0^t \left(\frac{\alpha}{2}\left\|u_s(\boldsymbol{x}_s)\right\|^2 - \rho_s(\boldsymbol{x}_s)\right) ds - r(\boldsymbol{x}_0)\middle| \boldsymbol{x}_t\right], \tag{113}$$

it is known that the adjoint state of the controlled trajectory from $\boldsymbol{x}_t$ is an unbiased estimator of the gradient of the cost-to-go functional for any fixed control u;

$$\mathbb{E}_{\tau \sim p^u}\left[\boldsymbol{a}_t(\tau, u)|\boldsymbol{x}_t\right] = \nabla_{\boldsymbol{x}_t} J(u; \boldsymbol{x}_t, t). \tag{114}$$

Under the optimal control $u^\star$, the cost-to-go functional $J(u^\star; \boldsymbol{x}_t, t)$ becomes the value function $V_t^{\text{cost}}(\boldsymbol{x}_t)$ in terms of cost, which is the negated value function for the reward. Therefore,

$$\mathbb{E}_{\tau \sim p^{u^\star}}\left[-\frac{1}{\alpha}\boldsymbol{a}_t(\tau, u^\star)|\boldsymbol{x}_t\right] = -\frac{1}{\alpha}\nabla_{\boldsymbol{x}_t} V_t^{\text{cost}}(\boldsymbol{x}_t) = \frac{1}{\alpha}\nabla_{\boldsymbol{x}_t} V_t(\boldsymbol{x}_t) = \boldsymbol{\Psi}_t^\star(\boldsymbol{x}_t) \tag{115}$$

Compared to the unbiased value gradient estimator in Eq. (115), $\boldsymbol{\Psi}_t$ in Eq. (112) has an additional coefficient attached to the adjoint value $\boldsymbol{a}_t$. This means that Adjoint Matching only yields the optimal tilted marginal distribution in Eq. (7) when $\frac{2\tilde{\sigma}_t^2\delta(t)}{\tilde{\sigma}_t^2\delta(t)+2} = 1$, which only happens when $\tilde{\sigma}_t$ is equal to $\sqrt{2/\delta(t)}$, also known as *memoryless* noise schedule. In this specific setting of stochasiticity, the corresponding RSM coefficients for RSM in Eq. (112) can be further simplified as

$$\boldsymbol{\Psi}_t = -\frac{1}{\alpha}\boldsymbol{a}_t, \quad \gamma(t) = 1, \quad C_1(t) = \frac{2}{\delta(t)}, \quad C_2(t) = 0. \tag{116}$$

Adjoint Matching propagates the terminal reward gradient to intermediate states through a backward ODE to utilize an unbiased value gradient estimator for a first-order method. However, as mentioned in the main manuscript, this requires Jacobian propagation throughout timesteps, making it computationally heavy and time-consuming. Moreover, as a trade-off of introducing a full rollout, the estimator can have increased variance, which had to be addressed by variance reduction techniques such as the lean adjoint regression.

Note that since the adjoint state is an unbiased value gradient estimator only when the control is already optimal, the guarantee of convergence to the optimal control and the training dynamics have to be discussed similarly to other methods in Appendix B.7, which were further addressed by the authors [8].

C.2 Zeroth-order Methods

We now turn to policy-gradient methods, beginning with KL-regularized REINFORCE as the canonical strictly on-policy zeroth-order base case. This yields the fundamental zeroth-order effective value guidance, from which other variants arise as straightforward modifications within the RSM design space.

C.2.1 KL-regularized REINFORCE

We formulate REINFORCE [45, 53] in the KL-regularized form used by Algorithm 1 (Soft PPO) of Uehara et al. [47]. Using our general parameterization, the objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^\theta, \boldsymbol{\epsilon}_{t_{i:1}}}\left[-r(\boldsymbol{x}_0) + \alpha \sum_{i=1}^{N} \mathcal{D}_{\text{KL}}(p^\theta(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) \| p^{\text{ref}}(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}))\right]. \tag{117}$$

For Gaussian transitions with fixed covariance $\sigma_{t_i}^2 \mathbf{I}$, each KL term reduces to

$$\frac{\alpha}{2}\left\|\frac{1}{\sigma_{t_i}}(\boldsymbol{\mu}_{t_{i-1}}^\theta(\boldsymbol{x}_{t_i}) - \boldsymbol{\mu}_{t_{i-1}}^{\text{ref}}(\boldsymbol{x}_{t_i}))\right\|^2. \tag{118}$$

The REINFORCE gradient of the reward term is

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^\theta, \boldsymbol{\epsilon}_{t_{i:1}}}\left[-r(\boldsymbol{x}_0)\right] = \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^\theta, \boldsymbol{\epsilon}_{t_{i:1}}}\left[-r(\boldsymbol{x}_0)\nabla_\theta \log p_\theta(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i})\right]. \tag{119}$$

Since

$$\nabla_\theta \log p_\theta(\boldsymbol{x}_{t_{i-1}}|\boldsymbol{x}_{t_i}) = (\boldsymbol{x}_{t_{i-1}} - \boldsymbol{\mu}^\theta_{t_{i-1}})^\top \frac{1}{\sigma^2_{t_i}} \nabla_\theta \boldsymbol{\mu}^\theta_{t_{i-1}} = (\frac{1}{\sigma_{t_i}} \boldsymbol{\epsilon}_{t_i})^\top \nabla_\theta \boldsymbol{\mu}^\theta_{t_{i-1}}, \tag{120}$$

combining the reward and KL terms yields

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_\tau \left[\Big(- r(\boldsymbol{x}_0) \frac{1}{\sigma_{t_i}} \boldsymbol{\epsilon}_{t_i} + \alpha \frac{1}{\sigma^2_{t_i}} (\boldsymbol{\mu}^\theta_{t_{i-1}} - \boldsymbol{\mu}^{\text{ref}}_{t_{i-1}}) \Big)^\top \nabla_\theta \boldsymbol{\mu}^\theta_{t_{i-1}} \right], \tag{121}$$

where $\tau = \{\boldsymbol{x}_{t_i}\}_{i=0}^N$ denotes the trajectory induced by θ.

Substituting the unified parameterization $\boldsymbol{\mu}^\theta_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^\theta_{t_i}$ from Proposition 3.1, we obtain

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_\tau \left[\Big(- r(\boldsymbol{x}_0) \frac{1}{\sigma_{t_i}} \boldsymbol{\epsilon}_{t_i} + \frac{\alpha \Omega(t_i)}{\sigma^2_{t_i}} (\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i}) \Big)^\top \Omega(t_i) \nabla_\theta \boldsymbol{s}^\theta_{t_i} \right] \tag{122}$$

$$= \mathbb{E}_\tau \left[\frac{\alpha \Omega(t_i)^2}{\sigma^2_{t_i}} \Big(\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} - \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i} \Big)^\top \nabla_\theta \boldsymbol{s}^\theta_{t_i} \right]. \tag{123}$$

This is precisely the gradient of the weighted squared loss

$$\mathcal{L}(\theta) = \mathbb{E}_\tau \left[\frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \| (\boldsymbol{s}^\theta_{t_i} - (\boldsymbol{s}^{\text{ref}}_{t_i} + \boldsymbol{\Psi}_{t_i})) \|^2 \right], \tag{124}$$

with

$$\boldsymbol{\Psi}_{t_i} = \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2} \frac{\Omega(t_i)^2}{\sigma^2_{t_i}}, \quad C_2(t_i) = 0. \tag{125}$$

In relation to Proposition 3.1, REINFORCE corresponds to the zeroth-order estimation of the lookahead value gradient,

$$\mathbb{E}_{\boldsymbol{x}_{t_{i-1}} \sim p^\star_{\boldsymbol{x}_{t_i}}} \left[\nabla_{\boldsymbol{x}_{t_{i-1}}} V^\star_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}) \right] = \frac{\mathbb{E}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}]}{\sigma_{t_i}}, \tag{126}$$

using a point estimation for the expectation.

C.2.2 Clipped log-ratio surrogate

We next consider the clipped log-ratio surrogate loss used by PCPO-base [23]. Denoting the importance sampling ratio by

$$\rho_{t_i} = \frac{p^\theta_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})}{p^{\theta^\dagger}_{t_i}(\boldsymbol{x}_{t_{i-1}} \mid \boldsymbol{x}_{t_i})}, \tag{127}$$

the clipped objective is

$$\mathcal{L}_{\text{clip-log}}(\theta) = \begin{cases} \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^{\theta^\dagger}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[\frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} \right\|^2 \right] & \text{if } \big(\log \rho_{t_i} < -\xi \wedge r(\boldsymbol{x}_0) \le 0 \big) \\ & \quad \text{or } \big(\log \rho_{t_i} > \xi \wedge r(\boldsymbol{x}_0) \ge 0 \big), \\ \mathbb{E}_{\boldsymbol{x}_{t_i} \sim p^{\theta^\dagger}_{t_i}, \boldsymbol{\epsilon}_{t_i}} \left[-r(\boldsymbol{x}_0) \log \rho_{t_i} + \frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} \right\|^2 \right] & \text{otherwise,} \end{cases} \tag{128}$$

where

$$-\log \rho_{t_i} = -\frac{\Omega(t_i)}{\sigma_{t_i}} (\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i})^\top \boldsymbol{\epsilon}_{t_i} + \frac{\Omega(t_i)^2}{2\sigma^2_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i} \right\|^2. \tag{129}$$

We group terms by $\boldsymbol{s}^\theta_{t_i}$, and perform a square completion equivalent to the previous section:

$$\begin{aligned} & r(\boldsymbol{x}_0)(-\log \rho_{t_i}) + \frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \left\| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} \right\|^2 \\ &= -\frac{r(\boldsymbol{x}_0)\Omega(t_i)}{\sigma_{t_i}} (\boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} + \boldsymbol{s}^{\text{ref}}_{t_i})^\top \boldsymbol{\epsilon}_{t_i} + \frac{r(\boldsymbol{x}_0)}{2} \frac{\Omega(t_i)^2}{\sigma^2_{t_i}} \| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i} \|^2 + \frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\text{ref}}_{t_i} \|^2 \\ &= \frac{\alpha \Omega(t_i)^2}{2\sigma^2_{t_i}} \| \boldsymbol{s}^\theta_{t_i} - \big(\boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\sigma_{t_i}}{\alpha \Omega(t_i)} r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i} \big) \|^2 + \frac{r(\boldsymbol{x}_0)}{2} \frac{\Omega(t_i)^2}{\sigma^2_{t_i}} \| \boldsymbol{s}^\theta_{t_i} - \boldsymbol{s}^{\theta^\dagger}_{t_i} \|^2 + C \end{aligned} \tag{130}$$

where C is a non-gradient-producing constant w.r.t. θ. Thus,

$$\mathcal{L}(\theta) = \begin{cases} \mathbb{E}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right\|^2\right] & \text{if } \big(\log\rho_{t_i} < -\xi \wedge r(\boldsymbol{x}_0) \le 0\big) \\ & \quad \text{or } \big(\log\rho_{t_i} > \xi \wedge r(\boldsymbol{x}_0) \ge 0\big), \\ \mathbb{E}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \big(\boldsymbol{s}_{t_i}^{\text{ref}} + \boldsymbol{\Psi}_{t_i}\big)\right\|^2 + \frac{r(\boldsymbol{x}_0)}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\theta^\dagger}\right\|^2\right] & \text{otherwise.} \end{cases} \tag{131}$$

Therefore, the clipped log-ratio surrogate loss admits the canonical form with

$$\boldsymbol{\Psi}_{t_i} = \frac{\sigma_{t_i}}{\alpha\Omega(t_i)} r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{132}$$

C.2.3 Variants

EPG. EPG [4] is a variant of REINFORCE, which extends it to the *loose* on-policy setting, and uses a control variate for variance reduction. Its loss can be written as

$$\mathcal{L}_{\text{EPG}}(\theta) = \mathbb{E}_{\tau\sim p^{\theta^\dagger}}\left[\text{sg}(\rho_{t_i})\,\frac{\alpha\Omega(t_i)^2}{2\sigma(t_i)^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \big(\boldsymbol{s}_{t_i}^{\text{ref}} + \boldsymbol{\Psi}_{t_i}\big)\right\|^2\right], \tag{133}$$

with the same $\boldsymbol{\Psi}_{t_i}$ as KL-regularized REINFORCE. Compared to REINFORCE, the only modification is the multiplicative stop-gradient importance ratio, which does not produce gradients itself. Note that the practical calculation of ρ_{t_i} requires exponentiation, which is numerically unstable and exhibits worse training dynamics compared to the log-ratio surrogate [23].

PPO, GRPO. Huang et al. [18] proved that the KL-regularized PPO loss

$$\mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^{\theta^\dagger},\boldsymbol{\epsilon}_{t_i}}\left[\max\big(-\rho_{t_i}r(\boldsymbol{x}_0), -\text{clip}_\xi(\rho_{t_i})r(\boldsymbol{x}_0)\big) + \frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right\|^2\right] \tag{134}$$

is equivalent to

$$\mathcal{L}_{\text{clip}}(\theta) = \begin{cases} \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^{\theta^\dagger},\boldsymbol{\epsilon}_{t_i}}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right\|^2\right] & \text{if } \big(\rho_{t_i} < 1-\xi \wedge r(\boldsymbol{x}_0) \le 0\big) \\ & \quad \text{or } \big(\rho_{t_i} > 1+\xi \wedge r(\boldsymbol{x}_0) \ge 0\big), \\ \mathbb{E}_{\boldsymbol{x}_{t_i}\sim p^{\theta^\dagger},\boldsymbol{\epsilon}_{t_i}}\left[-r(\boldsymbol{x}_0)(\rho_{t_i}-1) + \frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right\|^2\right] & \text{otherwise.} \end{cases} \tag{135}$$

up to constants independent of θ. Therefore, PPO simply replaces the log-ratio term in Eq. (128) with $\rho_{t_i} - 1$. For Gaussian transitions,

$$\rho_{t_i} - 1 = \exp\left(-\frac{\Omega(t_i)}{\sigma_{t_i}}(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\theta^\dagger})^\top\boldsymbol{\epsilon}_{t_i} - \frac{\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\theta^\dagger}\right\|^2\right) - 1. \tag{136}$$

Directly optimizing this exponential is numerically unstable. In practice, diffusion/flow-based PPO and GRPO implementations [3, 10, 27, 57] use an extremely tight clipping threshold (e.g., $\xi = 10^{-4}$). Under such tight clipping, the first-order approximation $\exp(x) - 1 \approx x$ incurs error of order $\xi^2/2$ for the interior regime $1-\xi \le \rho_{t_i} \le 1+\xi$, as well as for clipped cases where the loss is already fixed at the clipping boundary[19]. Under this necessary approximation, PPO / GRPO reduce to the clipped log-ratio surrogate in Eq. (131), and therefore have the same unified terms as Eq. (132).

PCPO-reweight. For flow-matching models, PCPO *explicitly* reweights the policy loss by

$$\gamma_{t_i} = \frac{w_{\text{new}}(t_i)}{w(t_i)}, \qquad w_{\text{new}}(t_i) = \zeta\Delta t_i, \qquad \zeta = \sum_{i=1}^{N} w(t_i). \tag{137}$$

See Appendix D.3 for more details. This modifies only the effective guidance and anchor strength:

$$\boldsymbol{\Psi}_{t_i} = \gamma_{t_i}\frac{\sigma_{t_i}}{\alpha\Omega(t_i)} r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \gamma_{t_i}\frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{138}$$

[19]The remaining non-clipped sign-consistent regimes, $(\rho_{t_i} < 1-\xi \wedge r(\boldsymbol{x}_0) \ge 0)$ and $(\rho_{t_i} > 1+\xi \wedge r(\boldsymbol{x}_0) \le 0)$, admit a looser bound. Empirically, however, Lee and Ye [23] observe that $|\rho_{t_i} - 1| < 0.01$ throughout training due to a low learning rate, so the approximation error remains small in practice.

For diffusion models, PCPO *implicitly* reweights $h(t_i)$ by modifying the variance schedule σ_{t_i} to σ'_{t_i}. Its effect is summarized in Table 1.

GRPO-Guard. GRPO-Guard observes that $\log \rho_{t_i}$ is negatively biased by the KL term between the current and old policies, which empirically shifts its mean below zero. To correct this, it centers and rescales the log-ratio as

$$\sigma_{t_i}\big(\log \rho_{t_i} - \mathbb{E}[\log \rho_{t_i}]\big),$$

and additionally reweights the policy term by $1/\Delta t_i$. Under the clipped log-ratio objective, this yields

$$\mathcal{L}(\theta) = \begin{cases} \mathbb{E}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right\|^2\right] & \text{if } \big(\sigma_{t_i}(\log \rho_{t_i} - \mathbb{E}[\log \rho_{t_i}]) < -\xi \wedge r(\boldsymbol{x}_0) \le 0\big) \\ & \quad \text{or } \big(\sigma_{t_i}(\log \rho_{t_i} - \mathbb{E}[\log \rho_{t_i}]) > \xi \wedge r(\boldsymbol{x}_0) \ge 0\big), \\ \mathbb{E}\left[\frac{\alpha\Omega(t_i)^2}{2\sigma_{t_i}^2}\left\|\boldsymbol{s}_{t_i}^{\theta} - \big(\boldsymbol{s}_{t_i}^{\text{ref}} + \boldsymbol{\Psi}_{t_i}\big)\right\|^2\right] & \text{otherwise,} \end{cases} \tag{139}$$

with

$$\boldsymbol{\Psi}_{t_i} = \gamma_{t_i}\frac{\sigma_{t_i}}{\alpha\Omega(t_i)} r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}, \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = 0, \tag{140}$$

where $\gamma_{t_i} = \frac{\sigma_{t_i}\Omega(t_i)}{\Delta t_i}$. Thus, GRPO-Guard explicitly removes the old-policy anchor term.

TempFlow-GRPO. TempFlow-GRPO introduces *trajectory branching*: starting from a deterministic ODE trajectory, it revisits each intermediate latent $\boldsymbol{x}_{t_i}$, injects one-step SDE noise to generate multiple descendants, and then deterministically denoises each descendant to the terminal state for reward evaluation. This is closely related to *vine sampling* used to reduce estimator variance in TRPO [41], and is precisely a multi-particle lookahead estimator in our framework. Note that TempFlow-GRPO does not explicitly average value-guidance vectors; instead, it applies an L_2 regression loss to multiple reward-weighted point estimates. However, because the gradient of a sum of squared errors is centered at the sample mean, this optimization is functionally equivalent to regressing toward the average of those particle-wise targets. Hence, under RSM, TempFlow-GRPO implicitly realizes a zeroth-order lookahead value-guidance estimator with branching width $K_i > 1$, together with heuristic temporal reweighting $\gamma(t_i) = \frac{9}{4}\sigma_{t_i}$. This yields

$$\boldsymbol{\Psi}_{t_i} = \gamma_{t_i}\frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\Big(\frac{1}{K_i}\sum_{k=1}^{K_i} r(\boldsymbol{x}_0^{(k)})\boldsymbol{\epsilon}_{t_i}^{(k)}\Big), \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \gamma_{t_i}\frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{141}$$

BranchGRPO. BranchGRPO likewise uses multiple descendants to estimate a zeroth-order lookahead guidance term, but differs from TempFlow-GRPO in that its branching is *recursive* rather than one-step from each revisited parent latent. Under the same RSM view, it therefore realizes a multi-particle estimator form:

$$\boldsymbol{\Psi}_{t_i} = \frac{\sigma_{t_i}}{\alpha\Omega(t_i)}\Big(\frac{1}{K_i}\sum_{k=1}^{K_i} r(\boldsymbol{x}_0^{(k)})\boldsymbol{\epsilon}_{t_i}^{(k)}\Big), \quad C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}, \quad C_2(t_i) = \frac{r(\boldsymbol{x}_0)}{\alpha}. \tag{142}$$

C.3 Remark on parameters.

To address the discrepancies in naming conventions across Soft RL and GFlowNet literature, we have unified the notation for components that serve identical functions. Table 3 maps the various notations found in prior works to the single, unified notation used in this text.

Entropy Regularization (α): While often denoted as β (inverse temperature) or $1/\beta$ in Soft RL contexts, we adopt α to strictly control the degree of entropy regularization[20] .

Downweighting Factor ($\tilde{\gamma}_t$): We rename VGG-Flow's one-step reward estimation factor from η_t to $\tilde{\gamma}_t$ to align with the conventions established by Residual ∇-DB and SQDF.

Reward Value ($r(\cdot)$): In diffusion fine-tuning GFlowNet literature, the reward is often interpreted as the log of the GFlowNet reward. We use $r(\cdot)$ directly to refer to this value.

[20] SQDF [21] apply mean reduction over the latent dimension d for *only* entropy regularization, and omit the $\frac{1}{2}$ coefficient for the KL divergence. Consequently, their reported α corresponds to an effective regularization weight of $2\alpha/d$.

Table 3: **Unified Notation for Prior Works.**

Parameter Description	Notation in Literature	Unified
Entropy Regularization	α [47] $\frac{2\alpha}{d}$ [21] β [23, 27] $\frac{1}{\beta}$ [29, 30]	α
Downweighting Factor	η_t [29] γ_t [21, 30]	$\tilde{\gamma}_t$
Residual Parameterization	g_ϕ [30] $-\nu_\phi$ [29]	g_ϕ
Reward Representation	$\log R(\cdot)$ [30, 59] $r(\cdot)$ [21, 23, 27, 29, 47]	$r(\cdot)$

D Parameterizations

D.1 Ω Derivations for Popular Samplers

In this section, we provide step-by-step derivations of the scaling factor $\Omega(t_i)$ for two widely used sampling algorithms: DDIM applied to VP-SDE (e.g., Stable Diffusion 1.5) and Euler Discrete applied to Rectified Flow (e.g., Stable Diffusion 3).

From Proposition 3.1, the relationship between the optimal score $\boldsymbol{s}^\star_{t_i}$, the reference score $\boldsymbol{s}^{\text{ref}}_{t_i}$, and the value gradient is established via the conditional transition mean $\boldsymbol{\mu}_{t_{i-1}}$. When $\boldsymbol{\mu}_{t_{i-1}}$ can be expressed in the form $\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}_{t_i}$, the guidance term simplifies to:

$$\boldsymbol{s}^\star_{t_i} = \boldsymbol{s}^{\text{ref}}_{t_i} + \frac{\sigma^2_{t_i}}{\alpha\Omega(t_i)}\nabla_{\boldsymbol{x}_{t_{i-1}}} V_{t_{i-1}}(\boldsymbol{x}_{t_{i-1}}). \tag{143}$$

The specific value of $\Omega(t_i)$ depends on the choice of noise schedule and numerical solver.

D.1.1 DDIM Sampler (VP-SDE)

For the DDIM sampler applied to a VP-SDE, the transition from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_{i-1}}$ is defined as:

$$\boldsymbol{x}_{t_{i-1}} = \sqrt{\bar{\alpha}_{t_{i-1}}}\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) + \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i} + \sigma_{t_i}\boldsymbol{\epsilon}_{t_i} \tag{144}$$

where $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) = \frac{\boldsymbol{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i}}{\sqrt{\bar{\alpha}_{t_i}}}$ is the Tweedie estimate of the clean data, and $\boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the injected noise. The conditional transition mean $\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i})$ can be rewritten as:

$$\begin{aligned}
\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) &= \sqrt{\bar{\alpha}_{t_{i-1}}}\left(\frac{\boldsymbol{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i}}{\sqrt{\bar{\alpha}_{t_i}}}\right) + \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\boldsymbol{\epsilon}^\theta_{t_i} \\
&= \sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\boldsymbol{x}_{t_i} - \left(\sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\sqrt{1-\bar{\alpha}_{t_i}} - \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\right)\boldsymbol{\epsilon}^\theta_{t_i} \\
&= \sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\boldsymbol{x}_{t_i} + \underbrace{\left(\sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}}\sqrt{1-\bar{\alpha}_{t_i}} - \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma^2_{t_i}}\right)\sqrt{1-\bar{\alpha}_{t_i}}}_{\Omega(t_i)}\boldsymbol{s}^\theta_{t_i}.
\end{aligned} \tag{145}$$

D.1.2 SDE-DPM-Solver++ (VP-SDE)

To further demonstrate the generality of our framework, we derive $\Omega(t_i)$ for SDE-DPM-Solver++ [31], a higher-order solver supported by Residual ∇-DB.

Let $a_t = \sqrt{\bar{\alpha}_t}$ and $b^2_t = 1 - a^2_t$ for the brevity. Let $\lambda_t = \log(a_t/b_t)$ be log-SNR and $h = \lambda_{t_{i-1}} - \lambda_{t_i}$. For SDE-DPM-Solver++ 1 sampler applied to a VP-SDE, the transition from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_{i-1}}$ is defined

as:

$$\boldsymbol{x}_{t_{i-1}} = \frac{b_{t_{i-1}}}{b_{t_i}} e^{-h} \boldsymbol{x}_{t_i} + a_{t_{i-1}}(1 - e^{-2h})\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) + b_{t_{i-1}}\sqrt{1 - e^{-2h}}\boldsymbol{\epsilon}_{t_i}, \tag{146}$$

where $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) = \frac{\boldsymbol{x}_{t_i} - b_{t_i}\boldsymbol{\epsilon}^\theta_{t_i}}{a_{t_i}} = \frac{\boldsymbol{x}_{t_i} + b^2_{t_i}\boldsymbol{s}^\theta_{t_i}}{a_{t_i}}$ denotes the Tweedie estimate of the clean data, and $\boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(0, \mathbf{I})$ is the sampled random noise. Thus,

$$\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) = \left(\frac{b_{t_{i-1}}}{b_{t_i}} e^{-h} + \frac{a_{t_{i-1}}}{a_{t_i}}(1 - e^{-2h})\right)\boldsymbol{x}_{t_i} + \frac{a_{t_{i-1}}}{a_{t_i}}(1 - e^{-2h}) b^2_{t_i}\boldsymbol{s}^\theta_{t_i}. \tag{147}$$

The covariance for this step is $\boldsymbol{\Sigma}_{t_i} = b^2_{t_{i-1}}(1 - e^{-2h})\mathbf{I} := \sigma_{t_i}\mathbf{I}$. By comparing this result with the form $\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$, we have

$$\Omega(t_i) = \frac{a_{t_{i-1}}}{a_{t_i}}\left(1 - e^{-2h}\right) b^2_{t_i} = \frac{a_{t_{i-1}}}{a_{t_i}} \frac{b^2_{t_i}}{b^2_{t_{i-1}}}\left(1 - e^{-2h}\right) b^2_{t_{i-1}} = \sqrt{\frac{\bar{\alpha}_{t_{i-1}}}{\bar{\alpha}_{t_i}}} \frac{1 - \bar{\alpha}_{t_i}}{1 - \bar{\alpha}_{t_{i-1}}}\sigma_{t_i}. \tag{148}$$

D.1.3 Euler Discrete Sampler (Flow-SDE)

For the Euler Discrete sampler applied to Rectified Flow (e.g., Flow-GRPO [27]), we consider the reverse SDE that shares the same marginal densities as the probability flow ODE. The reverse SDE step from t_i to t_{i-1} (where $t_{i-1} < t_i$, and we define $\Delta t_i = t_i - t_{i-1}$) takes the form:

$$\boldsymbol{x}_{t_{i-1}} = \underbrace{\boldsymbol{x}_{t_i} - \Delta t_i \boldsymbol{v}^\theta_{t_i}(\boldsymbol{x}_{t_i}) + \frac{\sigma^2_{t_i}}{2}\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})}_{\boldsymbol{\mu}_{t_{i-1}}} + \sigma_{t_i}\boldsymbol{\epsilon}_{t_i}, \tag{149}$$

where $\boldsymbol{v}^\theta_{t_i}(\boldsymbol{x}_{t_i})$ is the velocity field prediction, $\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$ is the score prediction, and $\boldsymbol{\epsilon}_{t_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the injected noise[21]. In Rectified Flow, the velocity field and the marginal score are related by:

$$\boldsymbol{v}_{t_i}(\boldsymbol{x}_{t_i}) = -\frac{1}{1 - t_i}\boldsymbol{x}_{t_i} - \frac{t_i}{1 - t_i}\boldsymbol{s}_{t_i}(\boldsymbol{x}_{t_i}), \tag{150}$$

which is consistent with the Tweedie estimate $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{t_i}) = \frac{\boldsymbol{x}_{t_i} + \boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})t_i^2}{1 - t_i} = \boldsymbol{x}_{t_i} - \boldsymbol{v}^\theta_{t_i}(\boldsymbol{x}_{t_i})t_i$. Substituting Eq. (150) for $\boldsymbol{v}^\theta_{t_i}$ into the conditional transition mean $\boldsymbol{\mu}_{t_{i-1}}$, we get:

$$\begin{aligned}
\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) &= \boldsymbol{x}_{t_i} - \Delta t_i\left(-\frac{1}{1 - t_i}\boldsymbol{x}_{t_i} - \frac{t_i}{1 - t_i}\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})\right) + \frac{\sigma^2_{t_i}}{2}\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i}) \\
&= \left(1 + \frac{\Delta t_i}{1 - t_i}\right)\boldsymbol{x}_{t_i} + \left(\frac{t_i}{1 - t_i}\Delta t_i + \frac{\sigma^2_{t_i}}{2}\right)\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i}).
\end{aligned} \tag{151}$$

Next, our aim is to write the guidance term via the transition mean $\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$. Equating the coefficients of $\boldsymbol{s}^\theta_{t_i}(\boldsymbol{x}_{t_i})$, we obtain:

$$\Omega(t_i) = \frac{t_i}{1 - t_i}\Delta t_i + \frac{\sigma^2_{t_i}}{2}. \tag{152}$$

Note that $\sigma_{t_i} = \tilde{\sigma}_{t_i}\sqrt{\Delta t_i}$, where $\tilde{\sigma}_{t_i} = a\sqrt{\frac{t_i}{1 - t_i}}$ for Flow-GRPO SDE and $\tilde{\sigma}_{t_i} = a$ for Dance-GRPO SDE.

This derivation clarifies how the intrinsic parameters of the flow model (t_i) and the chosen noise scale (σ_{t_i}) jointly determine the magnitude of value guidance $\Omega(t_i)$ required to navigate the reverse SDE towards the optimal policy.

[21] Here, σ_t follows the definition in Proposition 3.1. Note that $\sigma_t = \tilde{\sigma}_t(\Delta t)$.

D.1.4 CPS Sampler

CPS [51] uses a modified sampler for flow-matching models with a coefficient-preserving stochastic interpolation. Its transition mean is defined using

$$\tilde{\sigma}_{t_i} = t_{i-1} \sin\left(\frac{\eta\pi}{2}\right), \tag{153}$$

and

$$\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) = (1 - t_{i-1})\hat{\boldsymbol{x}}_0 + \sqrt{t_{i-1}^2 - \tilde{\sigma}_{t_i}^2}\, \hat{\boldsymbol{x}}_1, \tag{154}$$

where

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_{t_i} - t_i \boldsymbol{v}_{t_i}, \qquad \hat{\boldsymbol{x}}_1 = \boldsymbol{x}_{t_i} + (1 - t_i)\boldsymbol{v}_{t_i}. \tag{155}$$

Since

$$\sqrt{t_{i-1}^2 - \tilde{\sigma}_{t_i}^2} = t_{i-1} \cos\left(\frac{\eta\pi}{2}\right), \tag{156}$$

the transition mean becomes

$$\begin{aligned}\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) = &\left(1 - t_{i-1} + t_{i-1} \cos\left(\frac{\eta\pi}{2}\right)\right) \boldsymbol{x}_{t_i} \\ &+ \left(-t_i(1 - t_{i-1}) + (1 - t_i)t_{i-1} \cos\left(\frac{\eta\pi}{2}\right)\right) \boldsymbol{v}_{t_i}.\end{aligned} \tag{157}$$

For Rectified Flow,

$$\boldsymbol{v}_{t_i} = -\frac{1}{1 - t_i}\boldsymbol{x}_{t_i} - \frac{t_i}{1 - t_i}\boldsymbol{s}_{t_i}, \tag{158}$$

so substituting the velocity–score relation yields

$$\begin{aligned}\boldsymbol{\mu}_{t_{i-1}}(\boldsymbol{x}_{t_i}) = &\left(1 - t_{i-1} + t_{i-1} \cos\left(\frac{\eta\pi}{2}\right) + \frac{t_i(1 - t_{i-1})}{1 - t_i} - t_{i-1} \cos\left(\frac{\eta\pi}{2}\right)\right) \boldsymbol{x}_{t_i} \\ &+ \underbrace{\left(\frac{t_i^2(1 - t_{i-1})}{1 - t_i} - t_i t_{i-1} \cos\left(\frac{\eta\pi}{2}\right)\right)}_{\Omega(t_i)} \boldsymbol{s}_{t_i}.\end{aligned} \tag{159}$$

Therefore, CPS admits the unified transition parameterization

$$\boldsymbol{\mu}_{t_{i-1}} = \kappa(t_i)\boldsymbol{x}_{t_i} + \Omega(t_i)\boldsymbol{s}_{t_i}, \tag{160}$$

with

$$\boxed{\Omega(t_i) = \frac{t_i^2(1 - t_{i-1})}{1 - t_i} - t_i t_{i-1} \cos\left(\frac{\eta\pi}{2}\right)}. \tag{161}$$

The corresponding stochastic transition coefficient is

$$\boxed{\sigma_{t_i} = t_{i-1} \sin\left(\frac{\eta\pi}{2}\right) \sqrt{t_i - t_{i-1}}}. \tag{162}$$

D.2 δ Derivations for Common Parameterizations

We derive $\delta(t_i)$, the signed scalar relating the parameterization difference to the score difference, for cases used in practice in this paper.

$\boldsymbol{\epsilon}$-prediction, VP-SDE. Under the forward marginal, the score satisfies

$$\boldsymbol{s}_{t_i} = -\frac{1}{\sqrt{1 - \bar{\alpha}_{t_i}}}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} = -\sqrt{1 - \bar{\alpha}_{t_i}}\, \boldsymbol{s}_{t_i}. \tag{163}$$

Therefore,

$$\boldsymbol{\epsilon}_{t_i}^{\theta} - \boldsymbol{\epsilon}_{t_i}^{\text{ref}} = -\sqrt{1 - \bar{\alpha}_{t_i}}\big(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\big).$$

Comparing this with

$$\boldsymbol{\epsilon}_{t_i}^{\theta} - \boldsymbol{\epsilon}_{t_i}^{\text{ref}} = -\frac{1}{\delta(t_i)}\big(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\big),$$

we obtain, for VP-SDE diffusion models,

$$\delta(t_i) = \frac{1}{\sqrt{1-\bar{\alpha}_{t_i}}}. \tag{164}$$

v-prediction, Rectified Flow. Consider the general forward marginal $p_t(\boldsymbol{x}_{t_i} \mid \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t_i}; a_{t_i}\boldsymbol{x}_0, b_{t_i}^2\mathbf{I})$, which induces the trajectory $\boldsymbol{x}_{t_i} = a_{t_i}\boldsymbol{x}_0 + b_{t_i}\boldsymbol{\epsilon}$. Differentiating with respect to time gives $\boldsymbol{v}_{t_i} \triangleq \frac{d\boldsymbol{x}_{t_i}}{dt} = \dot{a}_{t_i}\boldsymbol{x}_0 + \dot{b}_{t_i}\boldsymbol{\epsilon}$. Substituting $\boldsymbol{x}_0 = \frac{1}{a_{t_i}}(\boldsymbol{x}_{t_i} - b_{t_i}\boldsymbol{\epsilon})$ to eliminate the clean data yields

$$\boldsymbol{v}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\dot{a}_{t_i}}{a_{t_i}}\boldsymbol{x}_{t_i} + \left(\dot{b}_{t_i} - \frac{\dot{a}_{t_i}b_{t_i}}{a_{t_i}}\right)\boldsymbol{\epsilon}.$$

Using the score identity (Eq. (163)), this becomes

$$\boldsymbol{v}_{t_i}(\boldsymbol{x}_{t_i}) = \frac{\dot{a}_{t_i}}{a_{t_i}}\boldsymbol{x}_{t_i} + \left(\frac{\dot{a}_{t_i}b_{t_i}^2 - a_{t_i}\dot{b}_{t_i}b_{t_i}}{a_{t_i}}\right)\boldsymbol{s}_{t_i}(\boldsymbol{x}_{t_i}).$$

The coefficient of $\boldsymbol{x}_{t_i}$ depends only on the fixed forward schedule and therefore contains no learned parameters. Hence, when taking the difference between the learned policy and the reference policy, the $\boldsymbol{x}_{t_i}$ terms cancel exactly, giving

$$\boldsymbol{v}_{t_i}^{\theta} - \boldsymbol{v}_{t_i}^{\text{ref}} = \left(\frac{\dot{a}_{t_i}b_{t_i}^2 - a_{t_i}\dot{b}_{t_i}b_{t_i}}{a_{t_i}}\right)\left(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right).$$

Combining this with

$$\boldsymbol{v}_{t_i}^{\theta} - \boldsymbol{v}_{t_i}^{\text{ref}} = -\frac{1}{\delta(t_i)}\left(\boldsymbol{s}_{t_i}^{\theta} - \boldsymbol{s}_{t_i}^{\text{ref}}\right),$$

we obtain

$$\delta(t_i) = -\frac{a_{t_i}}{\dot{a}_{t_i}b_{t_i}^2 - a_{t_i}\dot{b}_{t_i}b_{t_i}}.$$

For Rectified Flow, where $a_{t_i} = 1 - t_i$ and $b_{t_i} = t_i$, this reduces to

$$\delta(t_i) = \frac{1-t_i}{t_i}. \tag{165}$$

D.3 Sampler-dependent weights w

Table 1 shows $h(t_i)$, which summarizes timestep-wise optimization strength. For lookahead methods, a convenient bridge to this quantity is the sampler-dependent weight

$$w(t_i) := \frac{\Omega(t_i)\delta(t_i)}{\sigma_{t_i}} \quad \Longleftrightarrow \quad \Omega(t_i)\delta(t_i) = w(t_i)\sigma_{t_i}. \tag{166}$$

Substituting Eq. (166) into the lookahead form of $h(t_i)$ yields, for $C_1(t_i) = \frac{\alpha}{2}\frac{\Omega(t_i)^2}{\sigma_{t_i}^2}$,[22]

$$h(t_i) = \delta(t_i)C_1(t_i)\gamma(t_i)\frac{\sigma_{t_i}^2}{\alpha\Omega(t_i)} = \frac{\gamma(t_i)}{2}\delta(t_i)\Omega(t_i) = \frac{\gamma(t_i)}{2}w(t_i)\sigma_{t_i} \propto w(t_i)\sigma_{t_i}. \tag{167}$$

Thus, $w(t_i)$ provides a convenient sampler-level lens for understanding how different design choices reschedule effective optimization strength across timesteps. See Figure 2 of Lee and Ye [23] for the shapes of $w(t_i)$ under different samplers.

We now use PCPO-reweight as a case study. Its key modification is to replace the native schedule induced by $w(t_i)$ with a flatter alternative $w'(t_i)$, thereby reshaping $h(t_i)$.

[22] Residual ∇-DB is slightly different: since $C_1(t_i) = \frac{1}{d}\frac{\Omega(t_i)^2}{\sigma_{t_i}^4}$, the same substitution yields $h(t_i) \propto w(t_i)\sigma_{t_i}^{-1}$ rather than $w(t_i)\sigma_{t_i}$. The role of $w(t_i)$ as a useful sampler-level lens remains analogous.

Table 4: **Summary of experiment settings for Section 5.2.**

Experiment setting	Baseline	Base model	Reward model	Prompts	Max epoch
Mechanistic Analysis	TempFlow-GRPO	SD3.5-M	Aesthetic Score	Pick-a-Pic	150
Validation: ZO, Flow	TempFlow-GRPO	SD3.5-M	GenEval	GenEval	420
Validation: ZO, Diffusion	PCPO	SD1.5	HPSv2.1	HPDv2	300
Validation: FO, Flow	VGG-Flow	SD3.5-M	HPSv2.1	GenEval	200
Validation: FO, Diffusion	Residual ∇-DB	SD1.5	HPSv2.1	GenEval	100

DDIM sampler, ϵ-prediction (SD1.5). PCPO defines

$$w(t_i) = \frac{\frac{\sqrt{1-\bar{\alpha}_{t_i}}}{\sqrt{\alpha_{t_i}}} - \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma_{t_i}^2}}{\sigma_{t_i}}, \tag{168}$$

which satisfies Eq. (166) together with Ω from Eq. (145) and δ from Eq. (164). PCPO then modifies the noise schedule from σ_{t_i} to σ'_{t_i} so that the resulting weights become approximately constant, $w'(t_i) \approx w_{\text{const}}$. Accordingly, the induced schedule changes from $\frac{1}{2}\sigma_{t_i} w(t_i)$ to $\frac{1}{2}\sigma'_{t_i} w'(t_i)$. To avoid confounding this reweighting with a global change in loss scale, PCPO normalizes the modified schedule so that $\text{avg}(w') = \text{avg}(w)$.

Euler discrete sampler, v-prediction (SD3). Here PCPO keeps the sampler noise schedule fixed and instead reweights the native timestep weights directly. Writing $\sigma_{t_i} = \tilde{\sigma}_{t_i}\sqrt{\Delta t_i}$, where $\tilde{\sigma}_{t_i}$ is the instantaneous diffusion coefficient in the Itô SDE, the standard weight is

$$w(t_i) = \frac{\sqrt{\Delta t_i}}{\tilde{\sigma}_{t_i}} \left(1 + \frac{(1-t_i)\tilde{\sigma}_{t_i}^2}{2t_i}\right)^2, \tag{169}$$

which again satisfies Eq. (166) with Ω from Eq. (152) and δ from Eq. (165). PCPO replaces this native schedule by $w'(t_i) = w_{\text{const}}\Delta t_i$, so the induced weighting changes from $\frac{1}{2}\sigma_{t_i} w(t_i)$ to $\frac{1}{2}\sigma_{t_i} w'(t_i)$. As in the DDIM case, PCPO matches the average scale by enforcing $\text{avg}(w') = \text{avg}(w)$.

E Experiment Details

E.1 Toy experiments.

To conduct the experiments in Section 5.1, we designed a base data distribution on 2 dimensional space $p_0^{\text{ref}}(\boldsymbol{x}_0)$ as mixture of three Gaussian nodes with unit covariance:

$$p_0^{\text{ref}}(\boldsymbol{x}_0) := \frac{1}{3}\left\{\mathcal{N}\left(\begin{bmatrix}3\\-\sqrt{3}\end{bmatrix}, \mathbf{I}\right) + \mathcal{N}\left(\begin{bmatrix}-3\\-\sqrt{3}\end{bmatrix}, \mathbf{I}\right) + \mathcal{N}\left(\begin{bmatrix}0\\2\sqrt{3}\end{bmatrix}, \mathbf{I}\right)\right\} \tag{170}$$

We defined our reward $r(\boldsymbol{x}_0) := \boldsymbol{x}_0[0]/2 + 3$ as a linear function, which allows the reward-tilted distribution $p_0^\star(\boldsymbol{x}_0) \propto p_0^{\text{ref}}(\boldsymbol{x}_0)e^{r(\boldsymbol{x}_0)}$ to be another mixture of Gaussians. The noised marginal distributions $p_t^{\text{ref}}(\boldsymbol{x}_t)$ and $p_t^\star(\boldsymbol{x}_t)$ can be obtained in a closed form as an interpolation between data Gaussians and noise prior $\mathcal{N}(0, \mathbf{I})$. Their score function difference is used as a ground truth for the value gradient according to Eq. (8).

We trained diffusion models for $p_0^{\text{ref}}(\boldsymbol{x}_0)$ and $p_0^\star(\boldsymbol{x}_0)$ with ϵ-prediction on 500 timesteps. The training was done with batch size 4096 for 2000 iterations, using Adam optimizer with learning rate 10^{-3}. To simulate the value gradient estimator for different reward alignment training methods, DDIM sampling was done with timestep interval 10 with Markovian stochasticity.

E.2 End-to-end Design.

Table 4 summarizes experiment settings used in Section 5.2. Below we record the method-specific choices omitted from the main text. Unless otherwise specified, we use the default configurations of the corresponding baseline. Although we used H200 GPUs for some experiments to reduce wall-clock time, all experiments are reproducible on GPUs with 24GB VRAM except for the non-pruned VGG-Flow baseline.

Mechanistic analysis (ZO, flow). We follow TempFlow-GRPO on SD3.5-M [9] with Aesthetic Score [40] reward and Pick-a-Pic [22] prompts. Each epoch uses 16 prompts with 4 seeds per prompt. For entropy regularization, we use $\alpha = 0.01$ and follow the heuristic in the Flow-GRPO repository, which temporally weights the entropy term by Δt_i. The baseline branching profile is $K = [6, 6, 6, 6, 6, 6, 6, 6, 6]$. For the redesigned variants, the Uniform Reallocation profile is $K = [7, 7, 7, 7, 7, 7, 6, 6, 0]$, and the Equal Variance Reallocation profile is $K = [4, 5, 5, 6, 7, 8, 9, 10, 0]$. Following the policy-gradient setting, we use the group-normalized advantage $\hat{A}$ in place of raw reward r; see Appendix G.3. Evaluation was performed every 30th epoch on a fixed set of prompts. Wall-clock time is reported in H200 hours; 2 H200 GPUs were used for experiments.

Validation: ZO, flow. We follow the batch setting of TempFlow-GRPO, using 48 prompts per epoch with 4 seeds per prompt. For entropy regularization, we use $\alpha = 0.004$ and do *not* apply the Flow-GRPO Δt_i heuristic. Because GenEval is primarily semantic, our redesign does not invest branching budget at high-SNR timesteps; instead, it concentrates compute on the first few low-SNR timesteps, where semantic guidance is most useful. Concretely, we use the branching profile $K = [6, 6, 8, 10, 0, 0, 0, 0, 0]$, together with GRPO-Guard temporal reweighting and the fair clipping rule in Eq. (24). For non-branching timesteps, we add a KL penalty anchoring to $\boldsymbol{s}^{\text{ref}}_{t_i}$ using ODE trajectories, which avoids additional sampling cost for anchor trajectories. We use the same α for these ODE anchors. As above, we use the group-normalized advantage $\hat{A}$ in place of reward r. Wall-clock time is measured excluding GenEval reward computation. For the baseline, wall-clock time is estimated from the per-step time required under the TempFlow-GRPO setting. Our method was evaluated every 60th epoch, while the TempFlow-GRPO baseline reported evaluations every 20th epoch. Evaluation was performed on the GenEval test prompt dataset, separate from the training set. Wall-clock time is reported in H200 hours; 4 H200 GPUs were used for experiments.

Validation: ZO, diffusion. The baseline PCPO can be viewed as using the uniform profile $K_i \equiv 1$, i.e., a single sampled continuation at every reverse step. Ideally, one would branch at every reverse step, but doing so over the full trajectory ($N = 50$) is prohibitively expensive. We therefore branch at only 10% of reverse steps, leaving the remaining steps unbranched. Specifically, for a roughly matched per-epoch compute budget, we use 8 prompts per epoch with 5 seeds per prompt, and set $K_i = 4$ at every 10th timestep. For non-branching timesteps, we add a small $\|\boldsymbol{s}^\theta - \boldsymbol{s}^{\text{ref}}\|^2$ regularizer on 10% of ODE rows to prevent uncontrolled drift from the reference model, with $\alpha = 8 \times 10^{-5}$. We fine-tune SD1.5 [39] with LoRA [17] rank 16. Evaluation was performed every 10th epoch on a fixed set of prompts. As above, we use the group-normalized advantage $\hat{A}$ in place of reward r. Wall-clock time is reported in RTX4090 hours; 4 RTX4090 GPUs were used for experiments.

Validation: FO, flow. We follow the batch setting of VGG-Flow, using 32 generation trajectories and one update per epoch. The baseline relies on local Tweedie-based reward gradients, whereas our redesign uses terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$, which effectively correspond to using $j = 0$ reward information. The exact unbiased quantity would be $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$, but computing it requires an expensive high-order Jacobian chain. Following score distillation approaches, we therefore use $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ as a practical surrogate. To counteract inherited downweighting of low-SNR timesteps, we replace the baseline schedule $\tilde{\gamma}_{t_i} = (1 - t_i)^2$ with $\tilde{\gamma}_{t_i} = \frac{1-t_i}{2}$. We plot *transition mean drift*, i.e. $\mathbb{E}[\|\boldsymbol{\mu}^\theta - \boldsymbol{\mu}^{\text{ref}}\|^2]$, which is analogous to the KL divergence defined for lookahead methods, i.e. $\mathbb{E}[\|\boldsymbol{\mu}^\theta - \boldsymbol{\mu}^{\text{ref}}\|^2 / 2\sigma^2]$. Evaluation was performed every 50th epoch on the GenEval test dataset. As we observed more variability across random seeds in this setting, we report results averaged over three runs with different seeds. Wall-clock time is reported in H200 hours; 4 H200 GPUs were used for experiments.

Validation: FO, diffusion. Both the baseline and our method collect 256 generation trajectories. For training, we subsample 20% of transitions from each collected trajectory by uniformly splitting the trajectory into timestep intervals and sampling one transition from each interval, while always including the final transition, following Liu et al. [30]. The baseline again uses local Tweedie-based reward gradients, while our redesign uses terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ as a practical surrogate for the exact quantity $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$. This effectively corresponds to using $j = 0$ reward information without differentiating through the full Jacobian chain. To reverse the inherited low-SNR downweighting, we multiply $\ell^\theta_{t_i}$ by $3\sigma_{t_i}$ and set $\tilde{\gamma}_{t_i} = \frac{3}{2}\sigma_{t_i}$. Evaluation was perfomed every 50th epoch on the GenEval test dataset. Wall-clock time is reported in RTX4090 hours; 4 RTX4090 GPUs were used for experiments.

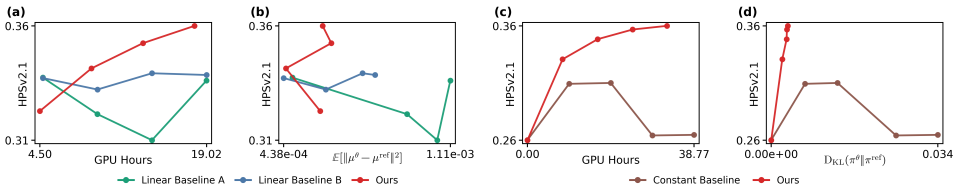

Figure 6: **Ablating the first-order estimator.** Replacing $\nabla_{\boldsymbol{x}_t} r(\hat{\boldsymbol{x}}_0)$ with $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ improves both reward efficiency and the reward–KL tradeoff. In flow matching, we compare against two linearized baselines that keep the original local Tweedie-based estimator but adopt milder temporal weighting: (a) reward vs. GPU hours; (b) reward vs. KL. In diffusion, we compare against the corresponding baseline with the original estimator under the modified weighting: (c) reward vs. GPU hours; (d) reward vs. KL.

Remark on Reward Score Distillation. The first-order validation experiments use terminal-image reward gradients $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ as a practical surrogate for the exact full-lookahead signal $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$. This choice should be understood as a *reward score distillation* approximation, analogous in spirit to Score Distillation Sampling (SDS) [35]: rather than backpropagating through the full denoising trajectory, we use the terminal reward gradient as a tractable direction for improving the intermediate score update. The exact quantity $\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0)$ would be unbiased, but requires differentiating through the full Jacobian chain, which is prohibitively memory-intensive in practical settings and prevents the method from retaining the memory footprint of the original first-order baselines. The recently proposed half-order fine-tuning [38] provides a more principled alternative by constructing unbiased estimators with lower variance than zeroth-order policy-gradient estimators, at the cost of maintaining a local backward computation graph. Such estimators may offer the best overall bias–variance tradeoff when the additional memory overhead is acceptable. Our goal in this validation experiment is more conservative: to show that a Pareto-improving redesign is possible under essentially the same memory budget as the baseline. Compared with local Tweedie-based guidance, which is severely biased at low-SNR timesteps, replacing the local gradient $\nabla_{\boldsymbol{x}_{t_i}} r(\hat{\boldsymbol{x}}_0 \mid t_i)$ with terminal-image guidance $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ reduces the dominant low-SNR bias while preserving the low variance and low memory cost of the original first-order update. Thus, the experiment should not be interpreted as claiming that reward score distillation is the statistically optimal estimator, but rather as demonstrating that a lightweight SDS-style surrogate can substantially improve the practical first-order design without introducing additional rollout or memory overhead.

E.3 Additional Experiments

Figures 8–11 were obtained using the original codebases of Residual ∇-DB [30] and VGG-Flow [29], with Aesthetic Score as the reward.

F Additional Results

F.1 Ablations for First-Order Experiments

To isolate the contribution of the modified value-guidance estimator $\boldsymbol{\Psi}_{t_i}$, we compare our full method against ablated variants that retain the baseline first-order estimator based on $\nabla_{\boldsymbol{x}_t} r(\hat{\boldsymbol{x}}_0)$ rather than $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$. In flow matching, our redesign changes both the estimator and the temporal weighting: it replaces the local Tweedie-based gradient with terminal-image reward gradients, and replaces the baseline weighting $\tilde{\gamma}(t_i) = (1 - t_i)^2$ with a milder low-SNR schedule. Since the original VGG-Flow paper already observed that a linear schedule can outperform $(1 - t_i)^2$ in some settings, Fig. 6(a, b) further compares against two linearized baselines to test whether our gains come solely from the weighting change. *Linear Baseline A* keeps the baseline estimator and uses $\tilde{\gamma}(t_i) = (1 - t_i)$, while *Linear Baseline B* keeps the baseline estimator and uses our scaled linear schedule. Our full method still performs substantially better, showing that the improvement is not explained by temporal weighting alone, but critically depends on the estimator change itself. In diffusion, Fig. 6(c, d) shows the analogous comparison against a baseline that keeps the original local Tweedie-based estimator under the same modified weighting. Overall, replacing $\nabla_{\boldsymbol{x}_t} r(\hat{\boldsymbol{x}}_0)$ with $\nabla_{\boldsymbol{x}_0} r(\boldsymbol{x}_0)$ consistently improves both reward efficiency and the reward–KL tradeoff in flow-matching and diffusion settings.

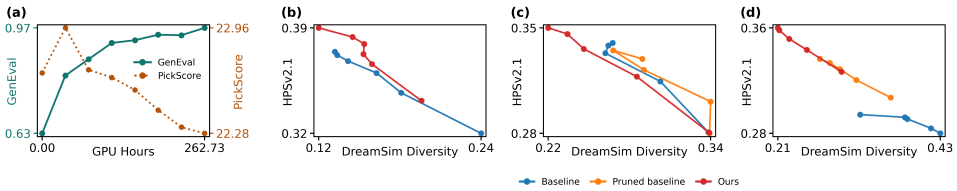

Figure 7: **Auxiliary metrics suggest no obvious reward hacking.** (a) PickScore during GenEval zeroth-order flow-matching fine-tuning. It improves initially and declines only modestly at later stages, rather than collapsing catastrophically. (b–d) DreamSim diversity on HPSv2.1 for zeroth-order diffusion, first-order flow matching, and first-order diffusion, respectively.

Table 5: **GenEval subtask breakdown.** Our method improves performance broadly across subtasks, rather than only within a narrow subset of categories. †: Results for SD3.5-M and TempFlow-GRPO are taken from Table 1 of He et al. [13].

Model	Overall	Single Obj.	Two Obj.	Counting	Colors	Position	Attr. Binding
SD3.5-M†	0.63	0.98	0.78	0.50	0.81	0.24	0.52
TempFlow-GRPO†	0.97	1.00	1.00	0.96	0.95	0.99	0.91
Ours	0.97	1.00	0.99	0.98	0.93	0.98	0.87

Table 6: **Cross-reward evaluation of GenEval fine-tuning.** Performance of the SD3.5-M base model and our final GenEval-fine-tuned checkpoint on the GenEval test prompts, measured using GenEval, CLIPScore, and PickScore.

Model	GenEval	CLIPScore	PickScore
SD3.5-M	0.63	0.289	22.67
Ours	0.97	0.316	22.28

F.2 Reward Hacking

We next examine whether the redesigns introduced in Section 5.2 improve the target reward at the cost of substantially degraded sample quality or diversity. Overall, we do not observe evidence that our improvements are driven by excessive reward hacking relative to the corresponding baselines.

For the GenEval zeroth-order flow-matching experiments, direct qualitative comparison to the baseline is unavailable because the original baseline checkpoints were not retained. We report GenEval subtask results in Table 5, which show that our method does not substantially underperform on any individual subtask. We also report validation on unseen rewards (Table 6, Figure 7(a)) as evidence against catastrophic model collapse. The increase in CLIPScore indicates that GenEval improvement is not attributable to reward hacking. PickScore exhibits a modest degradation at late stages of fine-tuning, consistent with the fact that PickScore captures aesthetics and general human preference in addition to prompt alignment, whereas GenEval favors simple and unambiguous images that are readily recognized by its external detector.

For other settings, we evaluate the tradeoff between reward and diversity by plotting reward–DreamSim Diversity [11] Pareto frontiers. Fig. 7(b)–(d) show that our method is not worse than the baseline on this frontier, indicating that the reward gains do not come at the cost of a systematically worse diversity profile. Taken together, these results suggest that the improvements from our redesigns reflect better optimization of the intended objective rather than merely exploiting weaknesses of the reward model.

Remark on Fig. 5(b). Although the non-pruned VGG-Flow baseline attains a slightly better reward–transition-drift Pareto frontier in Fig. 5(b), we do not view this as evidence that the auxiliary network g_ϕ is essential. First, g_ϕ may induce nontrivial interactions in optimization dynamics that make the unpruned model slightly easier to stabilize, but this modest benefit comes at a substantial systems cost: the non-pruned baseline requires roughly twice the VRAM and significantly longer wall-clock time, making it a poor tradeoff in practice. In our view, comparable gains would be better pursued through simpler knobs such as a smaller learning rate or a larger effective batch size in the pruned

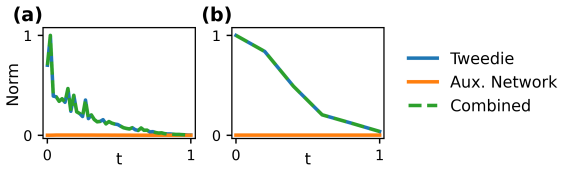

Figure 8: $\|g_\phi\|$ **is negligible.** The learned refinement term g_ϕ is negligible compared to the analytic reward gradient throughout the entire generation process for both (a) Residual ∇-DB, (b) VGG-Flow.

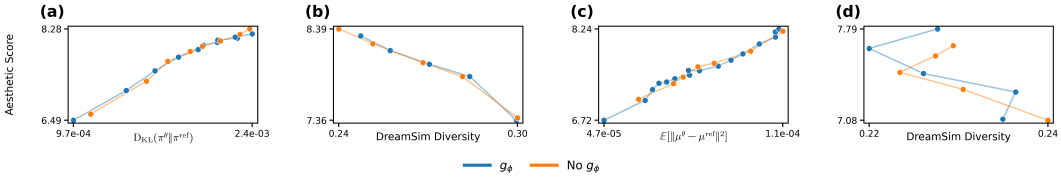

Figure 9: g_ϕ **is redundant for training.** Removing g_ϕ reduces wall-clock time, while maintaining optimality on the tradeoff between reward and prior/diversity preservation. Averaged across three consecutive random seeds. (a, b) Residual ∇-DB, (c, d) VGG-Flow[23].

baseline, rather than through an additional auxiliary network. Second, among all baselines, VGG-Flow exhibited the largest seed-to-seed variability, so advantages at earlier stages of training should be interpreted cautiously. Third, auxiliary evaluations based on DreamSim Diversity in Fig. 7(d) show that the pruned and unpruned baselines are comparable, with the pruned baseline slightly better if anything. Taken together, these observations support our main claim: g_ϕ is not a core ingredient of effective optimization, but an auxiliary component that adds substantial complexity for at most modest and inconsistent gains.

G Auxiliary Designs of Prior Work

G.1 Redundancy of Auxiliary Components

Our unified framework posits that the core optimization signal is fully captured by Eq. (10). This implies that auxiliary components common in prior work—specifically refinement networks and backward losses—are theoretically redundant or even harmful. We empirically validate that stripping these components simplifies the algorithm and reduces computational overhead without compromising performance; note that our findings are consistent with Liu et al. [28].

Redundancy of Refinement Networks (g_ϕ). Residual ∇-DB and VGG-Flow employ a learnable network g_ϕ to compensate for Tweedie approximation errors (i.e., by learning the residual $\nabla V - \nabla r(\hat{\boldsymbol{x}}_0)$). However, since g_ϕ is trained primarily via a terminal boundary loss ($g_\phi(\boldsymbol{x}_0) \to 0$) without intermediate supervision, it collapses toward zero due to the spectral bias of neural networks [37]. Empirical measurements in Fig. 8 confirm that the learned signal is negligible relative to the Tweedie gradient: $\|g_\phi\| \ll \frac{\tilde{\gamma}_t}{\alpha}\|\nabla r(\hat{\boldsymbol{x}}_0)\|$. Consequently, ablations in Fig. 9 demonstrate that removing g_ϕ maintains the Pareto frontier between reward maximization and prior preservation while strictly reducing training time. This confirms that g_ϕ contributes computational overhead without algorithmic benefit.

Instability of Backward Loss ($\mathcal{L}_{\text{backward}}$). Residual ∇-DB incorporates a backward loss $\mathcal{L}_{\text{backward}}$ derived from detailed balance conditions. As detailed in Appendix C.1.1, this term introduces high-order Jacobian dependencies that are analytically prone to variance explosion. Results in Fig. 10 confirm that $\mathcal{L}_{\text{backward}}$ induces training instability when its weight is non-negligible, whereas the forward gradient alone is sufficient for stable convergence. This suggests that the backward consistency loss is practically unnecessary, adding volatility rather than value.

[23] Due to missing checkpoints, panels (c) and (d) are reported from different runs.

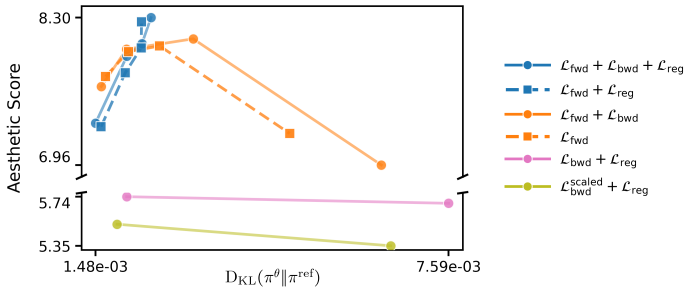

Figure 10: $\mathcal{L}_{\text{backward}}$ **of Residual ∇-DB does not contribute to effective training.**

G.2 Revisiting Existing Justifications for Attenuation Coefficients $\tilde{\gamma}_t$

Beyond the limitations of value function approximation—which can collapse when the terminal loss drives $g_\phi \approx \mathbf{0}$ (Section G.1)—the specific attenuation schedules $\tilde{\gamma}_t$ used in prior work remain only partially justified.

Residual ∇-DB. Motivated by GFlowNet theory, this method uses the forward-looking (FL) trick [34] to decompose $\nabla_{x_t} \log \tilde{F}(x_t)$ into a single-step Tweedie estimate and a residual term g_ϕ. However, this derivation relies on assumptions that are difficult to satisfy for complex image-generation rewards, such as computable intermediate energy or additive energy structure. Even if one interprets the "energy" more broadly as expected future reward, the resulting $\tilde{\gamma}_t$ schedule still appears heuristic. A simpler interpretation is our RSM perspective; the method attenuates the score-matching objective where the Tweedie estimate is less reliable.

VGG-Flow. VGG-Flow likewise introduces $\tilde{\gamma}_t$ and decomposes the gradient into a Tweedie term and a learnable correction g_ϕ, but lacks theoretical justification for the weighting itself. At the same time, the paper explicitly draws inspiration from score distillation [35], which is compatible with our score-matching interpretation.

SQDF. SQDF, motivated by Soft RL, interprets $\tilde{\gamma}_t$ as a credit-assignment coefficient, assigning little weight to high-noise regimes under the intuition that these states have lower signal-to-noise ratio and therefore weaker influence on the final sample. However, this intuition is not well aligned with the standard understanding of diffusion models, where low-SNR stages are often responsible for global semantic structure, and early denoising decisions can strongly affect the final reward [5]. As such, monotonically decaying credit assignment may be suboptimal for semantics-aware rewards such as GenEval [12]. Empirically, their intuition is also inconsistent with the strong performance of Flow-GRPO-Fast [27], which trains only on low-to-mid-SNR timesteps, yet outperforms its full-trajectory counterpart.

Overall, these observations suggest that interpreting $\tilde{\gamma}_t$ primarily as a damping factor for approximation error is more consistent with both prior empirical results and the known behavior of diffusion models.

G.3 Additional Design Considerations

We adopt specific simplifications to isolate the effects of our proposed components.

Reward Normalization. $\tilde{\gamma}(t)$ effectively controls the scale of the estimated reward gradient. Residual ∇-DB and VGG-Flow tune this quantity indirectly, through reward-specific choices of α or C_2, to improve stability. Motivated by the canonical gradient form (Eq. (12)), we suggest that first-order methods may reduce this hyperparameter sensitivity by explicitly scaling $\tilde{\gamma}(t)$ with $\frac{1}{\|\nabla r\|}$.

For zeroth-order methods, the reward-gradient estimator takes the form $\frac{1}{\sigma_{t_i}} \mathbb{E}[r(\boldsymbol{x}_0)\boldsymbol{\epsilon}_{t_i}]$. This perspective helps clarify why reward normalization can substantially improve optimization. First, subtracting the group mean acts as a control variate. Replacing r with $r - \hat{\mu}_G$ reduces estimator variance, while also changing the optimization geometry: instead of merely inducing stronger or weaker attraction toward high- and low-reward regions, it produces push–pull dynamics relative to the group average, which is often easier to optimize. Second, dividing by a scale term helps make the magnitude of the estimator more comparable across reward functions. In particular, replacing r with $\hat{A} = \frac{r - \hat{\mu}_G}{\hat{\sigma}_G}$ makes

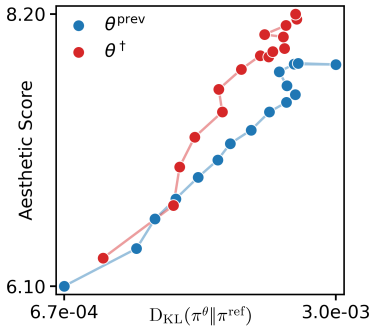

Figure 11: **Online samples suffice.** Including past rollouts (offline buffer) does not improve the Pareto frontier for Residual ∇-DB.

the scale of $\mathbb{E}[\hat{A}\boldsymbol{\epsilon}_{t_i}]$ roughly comparable across different rewards, and therefore eases hyperparameter tuning. This same scale-normalization principle also helps explain the empirical success of the Flow-GRPO implementation, which normalizes rewards to be approximately in $[0, 1]$. More broadly, the results of Choi et al. [4] suggest that dividing by the average magnitude $\|r\|$, or by the sample mean $\|\hat{\mu}_G\|$ as a practical surrogate, can further stabilize training.

Trust Regions. Trust-region mechanisms, including the quadratic penalty C_2 and clipping, should be viewed as heuristics rather than essential components.

For example, GRPO-Guard effectively sets $C_2(t_i) = 0$ while remaining stable. This suggests that the RSM objective can sometimes be simplified further without sacrificing performance, although doing so appears to require nontrivial engineering.

A similar picture emerges for clipping. EPG reports that, in some settings, removing clipping does not hurt performance. In contrast, we find that in certain Flow-GRPO settings, naively increasing the clip range ξ destabilizes training. Moreover, PCPO observes that standard PPO-style clipping can reduce the effective batch size and thereby harm performance [23]. This issue is particularly severe in Flow-SDEs, where the singular behavior near $t \approx 0$ causes nearly all samples to be clipped, as noted by Wang et al. [52].

Overall, these observations suggest that trust-region design remains an open heuristic space. Further simplification is possible in some regimes, but robust performance across settings likely depends on better-engineered heuristics.

Dataset $\mathcal{D}$: Focus on Online Samples. While prior work augments training with offline replay buffers [21, 30], empirical evidence suggests this offers minimal benefit. SQDF reports negligible difference in performance, and our ablations with Residual ∇-DB (Fig. 11) show that including past rollouts can actually degrade the Pareto frontier. Thus, we train exclusively on online samples.

Incorporation of CFG in Training Loss. Although we use CFG in all experiments, it is not a necessary component. Since CFG [14] amplifies the variance of score estimates, it can destabilize training [4, 27, 57]. In some settings, it may therefore be preferable to exclude CFG from the training loss.

H Extension of RSM Framework

H.1 Connection with Entropy-regularized Optimal Control Perspective

We provide a summary of the main results from prior work [36, 48]. Consider the soft RL objective defined over path measures:

$$V(\tau) = \max_{u,\nu} \left[\mathbb{E}_{\mathbb{P}_{u,\nu}}[r(\boldsymbol{x}_0)] - \alpha \mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{u,\nu}(\tau) \| \mathbb{P}_{\mathrm{ref}}(\tau)) \right]. \tag{171}$$

Here, $\mathbb{P}_{\mathrm{ref}}$ denotes the path measure induced by the pretrained diffusion model, while $\mathbb{P}_{u,\nu}$ denotes the path measure induced by a fine-tuned diffusion model and initial distribution ν.

Note that fine-tuning a diffusion model can be interpreted as modifying the reference dynamics through a drift perturbation. Such a perturbation admits a natural control interpretation, which enables reformulation of the soft RL objective as an entropy-regularized optimal control problem in continuous time.

To formalize this view, consider a reverse-time SDE with control $u : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$,

$$d\boldsymbol{x}_t = [f(\boldsymbol{x}_t, t) + u(\boldsymbol{x}_t, t)]dt + g(t)dw_t, \tag{172}$$

where $f(\boldsymbol{x}_t, t)$ denotes the drift coefficient including score function and $g(t)$ is an invertible diffusion coefficient. Our goal is to characterize the optimal control u that solves the aforementioned entropy-regularized problem. To obtain an explicit expression of this path-space KL divergence, we invoke Girsanov's theorem.

Girsanov's theorem. Consider the diffusion process with path measure $\mathbb{P}_{\text{ref}}$,

$$dX_t = f(X_t, t)dt + g(t)dW_t, \tag{173}$$

where $f(X_t, t)$ denotes the drift coefficient, $g(t)$ is an invertible diffusion coefficient, and W_t denotes the Brownian motion under $\mathbb{P}_{\text{ref}}$. Let $u(X_t, t)$ be an adapted process satisfying Novikov's condition. Define the stochastic exponential

$$Y_T = \exp\left(\int_0^T (g(t)^{-1}u(X_t, t))^\top dW_t - \frac{1}{2}\int_0^T \|g(t)^{-1}u(X_t, t)\|^2 dt\right). \tag{174}$$

As Y_T is a positive martingale with unit expectation, it defines a new probability measure $\mathbb{P}_u$ via

$$\frac{d\mathbb{P}_u}{d\mathbb{P}_{\text{ref}}} = Y_T. \tag{175}$$

Then, the process

$$W_t^u = W_t - \int_0^t g(s)^{-1}u(X_s, s)ds \tag{176}$$

is a Brownian motion under $\mathbb{P}_u$, and the dynamics of X_t become

$$dX_t = [f(X_t, t) + u(X_t, t)]dt + g(t)dW_t^u \tag{177}$$

Now, from the definition of path-space KL divergence, we get

$$\mathcal{D}_{\text{KL}}(\mathbb{P}_{u,\nu}||\mathbb{P}_{\text{ref}}) = \mathbb{E}_{\mathbb{P}_{u,\nu}}\left[\log\frac{\mathbb{P}_{u,\nu}}{\mathbb{P}_{\text{ref}}}\right] = \mathbb{E}_{\mathbb{P}_{u,\nu}}\left[\log\frac{\nu}{\nu_{\text{ref}}} + \log\frac{\mathbb{P}_{u,\nu}(\cdot|\boldsymbol{x}_T)}{\mathbb{P}_{\text{ref}}(\cdot|\boldsymbol{x}_T)}\right] \tag{178}$$

where the second equality follows from the factorization of the path measure into the initial distribution $\nu_{\text{ref}} = p_T^{\text{ref}}$ and the conditional dynamics. The first term corresponds to the discrepancy in the initial distributions, and the second term measures the deviation of the controlled dynamics from the reference dynamics conditioned on the initial state. With fixed initial sample $\boldsymbol{x}_T$, by the Girsanov theorem, we obtain

$$\begin{aligned}
\mathbb{E}_{u,\nu}\left[\log\frac{\mathbb{P}_{u,\nu}(\cdot|\boldsymbol{x}_T)}{\mathbb{P}_{\text{ref}}(\cdot|\boldsymbol{x}_T)}\right] &= \mathbb{E}_{u,\nu}\left[\int_0^T \frac{u(\boldsymbol{x}_t, t)^\top}{g(t)}d\boldsymbol{w}_t - \frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right] \\
&= \mathbb{E}_{u,\nu}\left[\int_0^T \frac{u(\boldsymbol{x}_t, t)^\top}{g(t)}\left(d\boldsymbol{w}^u + \frac{u(\boldsymbol{x}_t, t)}{g(t)}dt\right) - \frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right] \\
&= \mathbb{E}_{u,\nu}\left[\frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right],
\end{aligned} \tag{179}$$

where the second equality holds due to Eq. (176) and the third equality holds as $\boldsymbol{w}_t^u$ is $\mathbb{P}_{u,\nu}$-martingale.

In consequence, the soft RL objective function is rewritten as

$$\max_{u,\nu} \mathbb{E}_{\mathbb{P}_{u,\nu}}[r(\boldsymbol{x}_0)] - \alpha\mathbb{E}_{\mathbb{P}_{u,\nu}}\left[\frac{1}{2}\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt + \log\frac{\nu}{\nu_{\text{ref}}}\right]. \tag{180}$$

Under the perspective of optimal control problem, this is equivalent to

$$\min_{u,\nu} \frac{\alpha}{2}\mathbb{E}_{\mathbb{P}_{u,\nu}}\underbrace{\left[\int_0^T \|\frac{u(\boldsymbol{x}_t, t)}{g(t)}\|^2 dt\right]}_{\text{running cost}} + \mathbb{E}_{\mathbb{P}_{u,\nu}}\underbrace{[-r(\boldsymbol{x}_0)]}_{\text{terminal cost}} + \mathbb{E}_{\mathbb{P}_{u,\nu}}\underbrace{\left[\log\frac{\nu}{\nu_{\text{ref}}}\right]}_{\text{initial cost}}. \tag{181}$$

The initial cost appears because the initial distribution is treated as an optimization variable, which is not standard in classical stochastic optimal control formulation.

Value function and Optimal control. Uehara et al. [48] has shown that the marginal density induced by Eq. (172) with optimal control and optimal initial distribution obtained from Eq. (181) is

$$p_t^\star(\boldsymbol{x}_t) = \frac{1}{Z_{\text{tar}}} p_t^{\text{ref}}(\boldsymbol{x}_t) \exp\left(\frac{V_t^\star(\boldsymbol{x}_t)}{\alpha}\right) \tag{182}$$

where the optimal value function satisfies

$$\exp\left(\frac{V_t^\star(\boldsymbol{x}_t)}{\alpha}\right) = \mathbb{E}_{\boldsymbol{x}_0 \sim p^{\text{ref}}(\cdot|\boldsymbol{x}_t)}\left[\exp\left(\frac{r(\boldsymbol{x}_0)}{\alpha}\right)\right] \tag{183}$$

and the optimal control is defined by

$$u^\star(\boldsymbol{x}_t, t) = \frac{g^2(t)}{\alpha} \nabla_{\boldsymbol{x}} V_t^\star(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}_t} = g^2(t) \nabla_{\boldsymbol{x}} \log \mathbb{E}_{\boldsymbol{x}_0 \sim p^{\text{ref}}(\cdot|\boldsymbol{x})}\left[\exp\left(\frac{r(\boldsymbol{x}_0)}{\alpha}\right)\right]\Big|_{\boldsymbol{x}=\boldsymbol{x}_t} \propto \boldsymbol{\Psi}_t^\star(\boldsymbol{x}_t). \tag{184}$$

We emphasize that fine-tuning a diffusion model can be interpreted as perturbing the drift term via a control input. In this view, the fine-tuning model effectively learns to predict $\boldsymbol{s}_t^{\text{ref}} + u^\star$, which corresponds to the target score function in the RSM framework. The optimal control perspective further provides an alternative route to estimating the value function using classical tools from optimal control theory, such as adjoint sampling.

H.2 Reward-weighted Regression

Reward-weighted regression (RWR) provides a closely related regression-based perspective on online reward fine-tuning. Rather than regressing toward the value-guided optimal score in Eq. (8), RWR reweights the regression target of the pretraining loss according to the reward of generated samples. This structure closely resembles reward-weighted maximum likelihood estimation, a classical formulation of Soft RL, but is not exactly equivalent; hence, it is neither an exact Soft RL objective nor an exact instance of RSM.

This perspective encompasses seemingly different online fine-tuning constructions. DiffusionNFT [61] can be decomposed into an advantage-weighted regression term toward the forward velocity target together with an old-policy anchor. Meanwhile, an extension of DRaFT-LV [7] yields a first-order form of online RWR by directly differentiating the reward-weighted regression objective.

In particular, the first-order formulation (DRaFT-LV) makes explicit the Jacobian that propagates the reward-weighted regression target through the generative model. As shown in the following subsections, DiffusionNFT can be viewed as a simplified realization of this structure that locally approximates the Jacobian by a scaled identity matrix, clarifying both its connection to first-order reward fine-tuning and its distinction from exact RSM reductions.

H.2.1 DiffusionNFT

We next examine DiffusionNFT [61] through the lens of our unified formulation. Although its objective is written as a preference-weighted combination of two squared velocity-matching losses, its effective structure is easier to interpret after rewriting it as a regression objective with an explicit old-policy anchor. This reveals that DiffusionNFT simultaneously pulls the current velocity field toward the forward target $\boldsymbol{v}_t^{\text{fwd}}$ according to the clipped advantage, while regularizing it toward the reference velocity field $\boldsymbol{v}_t^{\theta^\dagger}$.

$$\begin{aligned}\mathcal{L}_{\text{NFT}}(\theta) = \mathbb{E}_{\boldsymbol{x}_0 \sim p^{\theta^\dagger},\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[&\frac{1+c}{2}\left\|(1-\beta)\boldsymbol{v}_t^{\theta^\dagger} + \beta \boldsymbol{v}_t^{\theta} - \boldsymbol{v}_t^{\text{fwd}}\right\|^2 \\ &+ \frac{1-c}{2}\left\|(1+\beta)\boldsymbol{v}_t^{\theta^\dagger} - \beta \boldsymbol{v}_t^{\theta} - \boldsymbol{v}_t^{\text{fwd}}\right\|^2],\end{aligned} \tag{185}$$

where $c = \text{clip}_1(\hat{A}(\boldsymbol{x}_0)) \in [-1, 1]$ and $\boldsymbol{v}_t^{\text{fwd}} = \boldsymbol{\epsilon} - \boldsymbol{x}_0$. Here, $\theta^\dagger$ denotes the old policy used to sample $\boldsymbol{x}_0$, while θ denotes the current policy being optimized.

Differentiating Eq. (185) w.r.t. θ and collecting terms yields

$$\nabla_\theta \mathcal{L}_{\mathrm{NFT}}(\theta) = \mathbb{E}\Big[2\beta\Big(\beta \boldsymbol{v}_t^\theta + (c-\beta)\boldsymbol{v}_t^{\theta^\dagger} - c\,\boldsymbol{v}_t^{\mathrm{fwd}}\Big)\cdot \nabla_\theta \boldsymbol{v}_t^\theta\Big]. \tag{186}$$

Therefore, up to θ-independent constants, Eq. (185) is equivalent to

$$\mathcal{L}(\theta) = \mathbb{E}\left[\beta^2 \left\| \boldsymbol{v}_t^\theta - \left(\boldsymbol{v}_t^{\theta^\dagger} + \frac{c}{\beta}(\boldsymbol{v}_t^{\mathrm{fwd}} - \boldsymbol{v}_t^{\theta^\dagger})\right)\right\|^2\right]. \tag{187}$$

This form shows that DiffusionNFT trains $\boldsymbol{v}_t^\theta$ toward an advantage-dependent interpolation between the old-policy velocity $\boldsymbol{v}_t^{\theta^\dagger}$ and the forward velocity target $\boldsymbol{v}_t^{\mathrm{fwd}}$. To make the two components explicit, we use the identity

$$\|y - ((1-\lambda)a + \lambda b)\|^2 = \lambda\|y-b\|^2 + (1-\lambda)\|y-a\|^2 + C \tag{188}$$

where C is independent of y. Applying the identity to Eq. (187) with $\lambda = \frac{c}{\beta}$, we see that up to θ-independent constants,

$$\mathcal{L}_{\mathrm{NFT}}(\theta) \;\equiv\; \mathbb{E}\Big[\beta c \big\|\boldsymbol{v}_t^\theta - \boldsymbol{v}_t^{\mathrm{fwd}}\big\|^2 + \beta(\beta-c)\big\|\boldsymbol{v}_t^\theta - \boldsymbol{v}_t^{\theta^\dagger}\big\|^2\Big]. \tag{189}$$

Thus, DiffusionNFT decomposes into a reward-weighted regression term toward the forward target $\boldsymbol{v}_t^{\mathrm{fwd}} = \boldsymbol{\epsilon} - \boldsymbol{x}_0$ and an old-policy anchor toward $\boldsymbol{v}_t^{\theta^\dagger}$. The relative strength of these two terms is controlled by the clipped advantage c and the coefficient β.

Connection to Reward-weighted MLE. Eq. (189) shows that DiffusionNFT consists of a reward-weighted regression term toward the forward target and an old-policy anchor. In parallel, Soft RL admits a reward-weighted MLE formulation, whose objective reduces to a reward-weighted denoising regression loss. This suggests viewing RWR as a practical approximation to reward-weighted MLE [47]. From this perspective, DiffusionNFT can be viewed as a practical approximation to Soft RL, and is therefore closely related to RSM. We hypothesize that this approximation may also be the source of its weaker optimization stability. This may explain why DiffusionNFT relies on EMA policy updates in practice, since EMA updates serve as a well-established remedies with favorable convergence guarantees in RL literature [20, 41].

H.2.2 DRaFT-LV

We next show that DRaFT-LV [7] suggests a first-order analogue of online reward-weighted regression. The contrast between DiffusionNFT and DRaFT-LV parallels the distinction between policy gradients and DRaFT.

DRaFT-LV is introduced as a low-variance extension of DRaFT-1. After generating an image $\boldsymbol{x}_0$, it forward-noises $\boldsymbol{x}_0$ multiple times and differentiates the reward through a one-step denoising prediction. In the notation of Clark et al. [7], DRaFT-LV fixes $K = 1$, so the reward gradient takes the form

$$\nabla_\theta r(\hat{\boldsymbol{x}}_0) = (\nabla_\theta \hat{\boldsymbol{x}}_0)^\top \nabla_{\hat{\boldsymbol{x}}_0} r(\hat{\boldsymbol{x}}_0) = -\frac{\sigma_1}{\alpha_1}(\nabla_\theta \boldsymbol{\epsilon}_1^\theta)^\top \nabla_{\hat{\boldsymbol{x}}_0} r(\hat{\boldsymbol{x}}_0). \tag{190}$$

Multiple independently noised copies of the same generated $\boldsymbol{x}_0$ are used to average this gradient and reduce its variance.

However, this original construction is specialized to the final denoising step. In particular, it evaluates the reward through a one-step Tweedie prediction and contains no explicit reference-model regularization. We consider a natural extension that retains DRaFT-LV's forward-noising construction while allowing arbitrary timesteps:

$$\boldsymbol{x}_{t_i} = a_{t_i}\boldsymbol{x}_0 + b_{t_i}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad t_i \sim p(t), \tag{191}$$

where, importantly, $\boldsymbol{x}_0$ is *not* detached from the computation graph. Consequently, the terminal reward gradient can be propagated to the forward-noised state,

$$\nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0), \tag{192}$$

rather than evaluating the reward gradient at a local Tweedie prediction. This avoids the target-point approximation underlying local first-order estimators.

Combining this first-order reward direction with reference-model regularization yields the generalized update

$$g = 2(\nabla_\theta \boldsymbol{v}_{t_i}^\theta)^\top \left[(\boldsymbol{v}_{t_i}^\theta - \boldsymbol{v}_{t_i}^{\mathrm{ref}}) + \frac{1}{\alpha_{\mathrm{KL}}(t_i)} \nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0) \right], \tag{193}$$

where $\alpha_{\mathrm{KL}}(t_i)$ is a timestep-dependent coefficient controlling the relative strength of reward guidance and reference regularization. We treat this coefficient as a design choice rather than directly inheriting the potentially ill-conditioned scaling induced by the forward-noising schedule.

Eq. (193) is the first-order counterpart of the generalized online-RWR gradient. Online RWR uses the reward-weighted forward-regression direction

$$r(\boldsymbol{x}_0)\big(\boldsymbol{v}_{t_i}^\theta - \boldsymbol{v}_{t_i}^{\mathrm{fwd}}\big), \tag{194}$$

whereas the first-order construction replaces this zeroth-order direction with

$$-\frac{1}{\alpha_{\mathrm{KL}}(t_i)} \nabla_{\boldsymbol{x}_{t_i}} r(\boldsymbol{x}_0). \tag{195}$$

Thus, the zeroth-order comparison Flow-GRPO versus online RWR has a direct first-order analogue: DRaFT estimates reward guidance by differentiating through the reverse sampling trajectory, while the generalized DRaFT-LV construction obtains the reward direction through forward noising of on-policy samples.

This perspective also clarifies the relationship to the common Jacobian approximation used in score-distillation-style methods. Replacing the exact pullback Jacobian relating $\boldsymbol{x}_{t_i}$ and $\boldsymbol{x}_0$ by an identity-like approximation, $\mathbf{J} \approx \mathbf{I}$, converts the exact first-order reward direction into a local terminal-image reward-gradient surrogate. This Jacobian approximation is therefore the first-order analogue of the forward-noising simplification underlying online RWR methods.

H.3 Inference-Time Alignment

While the Reward Score Matching framework primarily addresses weight updates during training, its core principles are isomorphic to inference-time alignment strategies. DNO [46] steers diffusion models without parameter fine-tuning by directly optimizing the intermediate noise variables $\boldsymbol{\epsilon}_t$ to maximize a reward. To handle non-differentiable rewards, DNO employs the *Hybrid2* method, estimating the gradient of the reward with respect to the noise via zeroth-order perturbations.

We demonstrate that this approach is functionally identical to applying the zeroth-order value guidance estimator to the noise space. Recall that zeroth-order methods update model parameters θ using the Monte Carlo estimator:

$$\frac{\sigma_{t_i}}{\alpha\Omega(t_i)} \mathbb{E}_{\boldsymbol{x}_{t_{i-1:0}} \sim p^\star_{t_{i:1}}, \boldsymbol{\epsilon}_{t_{i:1}} \overset{\mathrm{iid}}{\sim} \mathcal{N}(\mathbf{0},\mathbf{I})} \big[r(\boldsymbol{x}_0) \boldsymbol{\epsilon}_{t_i} \big]. \tag{196}$$

In parallel, DNO updates the latent variable $\boldsymbol{z}$ using the estimator (Eqs. (20, 21) in Tang et al. [46]):

$$\mathbb{E}_{\boldsymbol{\epsilon}} \Big[\frac{1}{\sigma} \big(r(D_\theta(\boldsymbol{z} + \sigma\boldsymbol{\epsilon})) - r(D_\theta(\boldsymbol{z})) \big) \big(D_\theta(\boldsymbol{z} + \sigma\boldsymbol{\epsilon}) - D_\theta(\boldsymbol{z}) \big) \Big] \cdot \nabla_{\boldsymbol{z}} D_\theta(\boldsymbol{z}), \tag{197}$$

where D_θ denotes the denoising network and σ denotes the degree of perturbation (not to be confused with variance schedule σ_{t_i}). Note that the update above is approximately equal to (Eqs. (18, 19) in Tang et al. [46]):

$$\mathbb{E}_{\boldsymbol{\epsilon}} \Big[\frac{1}{\sigma} \big(r(\boldsymbol{x} + \sigma\boldsymbol{\epsilon}) - r(\boldsymbol{x}) \big) \boldsymbol{\epsilon} \Big] \cdot \nabla_{\boldsymbol{z}} D_\theta(\boldsymbol{z}) \tag{198}$$

up to a scaling factor. This simplified form reveals a striking structural equivalence: both methods rely on a reward-weighted noise term ($\approx r \cdot \boldsymbol{\epsilon}$) to estimate the descent direction. This implies that training techniques, such as branching and signal-aware scheduling, can be repurposed to improve inference-time alignment.

H.4 AWM

Advantage Weighted Matching (AWM) [56] also connects diffusion alignment to score matching. However, its objective is fundamentally different from ours. AWM proposes a new advantage-weighted matching objective that brings RL post-training closer to the pretraining loss. By contrast,

our goal is not to reinterpret RL fine-tuning as pretraining, but to show that a broad class of *existing* reward-based fine-tuning methods already admit a common Reward Score Matching (RSM) form.

This distinction is central. AWM introduces a new training direction, whereas our contribution is a unifying analysis of prior methods. In particular, we show that methods derived from seemingly different perspectives reduce to a shared RSM structure, and that their primary differences are algorithmic—most notably, how value guidance is estimated and incorporated. To the best of our knowledge, this is the first work to provide such a unification of existing reward-based fine-tuning methods. Accordingly, AWM is best understood as an orthogonal and complementary direction, not a precursor to the perspective developed in this paper.

I Qualitative Results

This section presents qualitative comparisons between baseline methods and our improved variants from the **Validation Across Settings** experiments (Section 5.2). The baselines—Residual ∇-DB, VGG-Flow, PCPO, and TempFlow-GRPO—correspond to Figs. 12–15, respectively. Additional experimental details are provided in Appendix E.2 and summarized in Table 4.

Examples are not cherry-picked to highlight severe reward hacking in the baselines, nor do we claim that our methods exhibit emergent safeguards against reward hacking. Instead, these results serve to illustrate that the observed reward improvements are not achieved through more aggressive exploitation of the reward function. For Fig. 15, we report only our method's outputs, as the corresponding baseline checkpoints trained on GenEval are not publicly available and were not rerun.

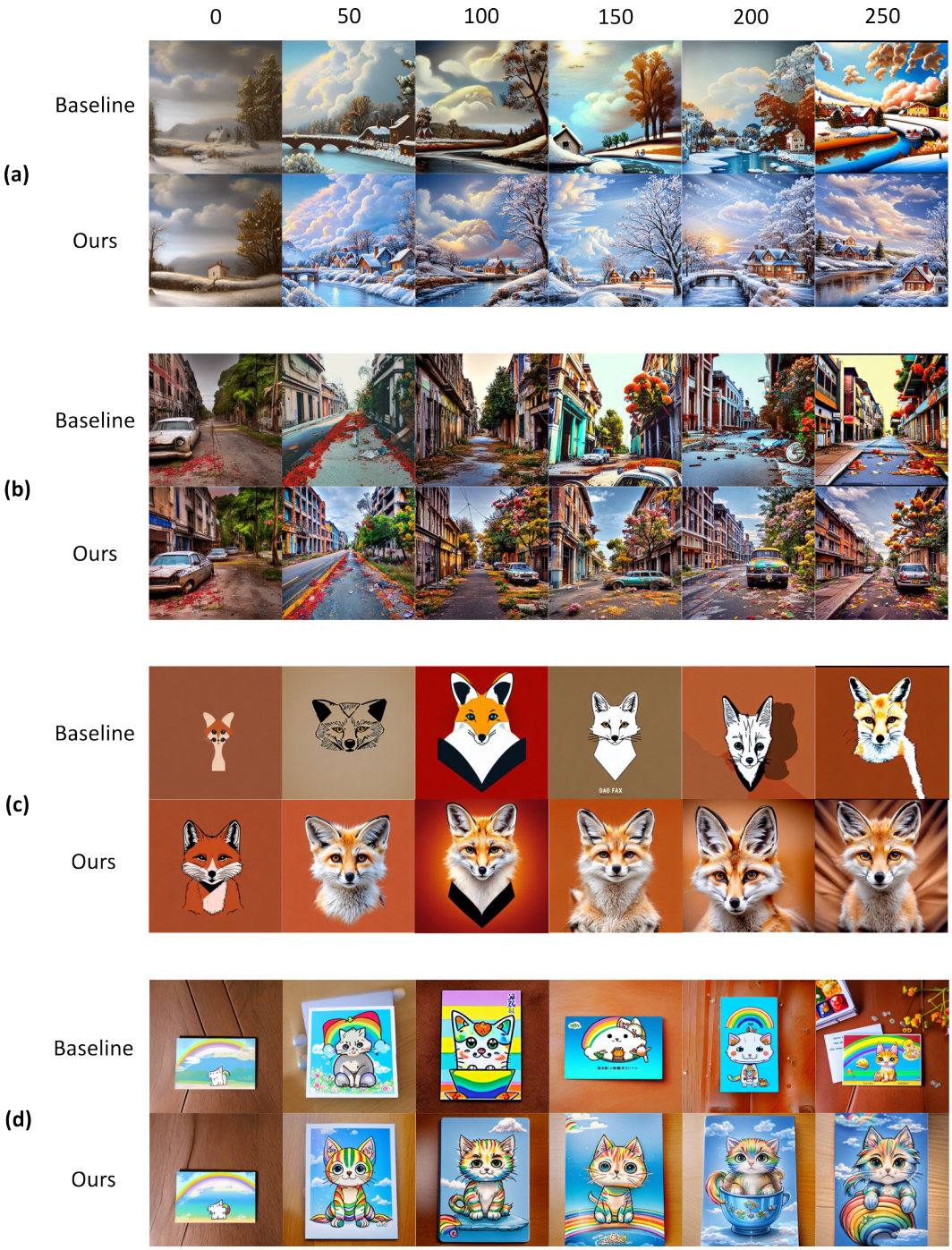

Figure 12: **Qualitative comparisons on the First-order, SD1.5 Validation Setting.** Images are shown at checkpoints after 0, 50, 100, 150, 200, 250 training epochs. Prompts: *(a) A painting depicting a snowy winter scene featuring a river, a small house on a hill, and a dreamy cloudy sky; (b) abandoned city with ruined buildings, long deserted streets, cars aged by time, trees, flowers, scattered leaves, empty street, vibrant colors; (c) portrait of a desert fox, professional, sleek, modern, minimalist, graphic, line art, simple background; (d) A postcard with cute rainbow kitten in front of blue sky in the chibi-style of Studio Ghibli.*

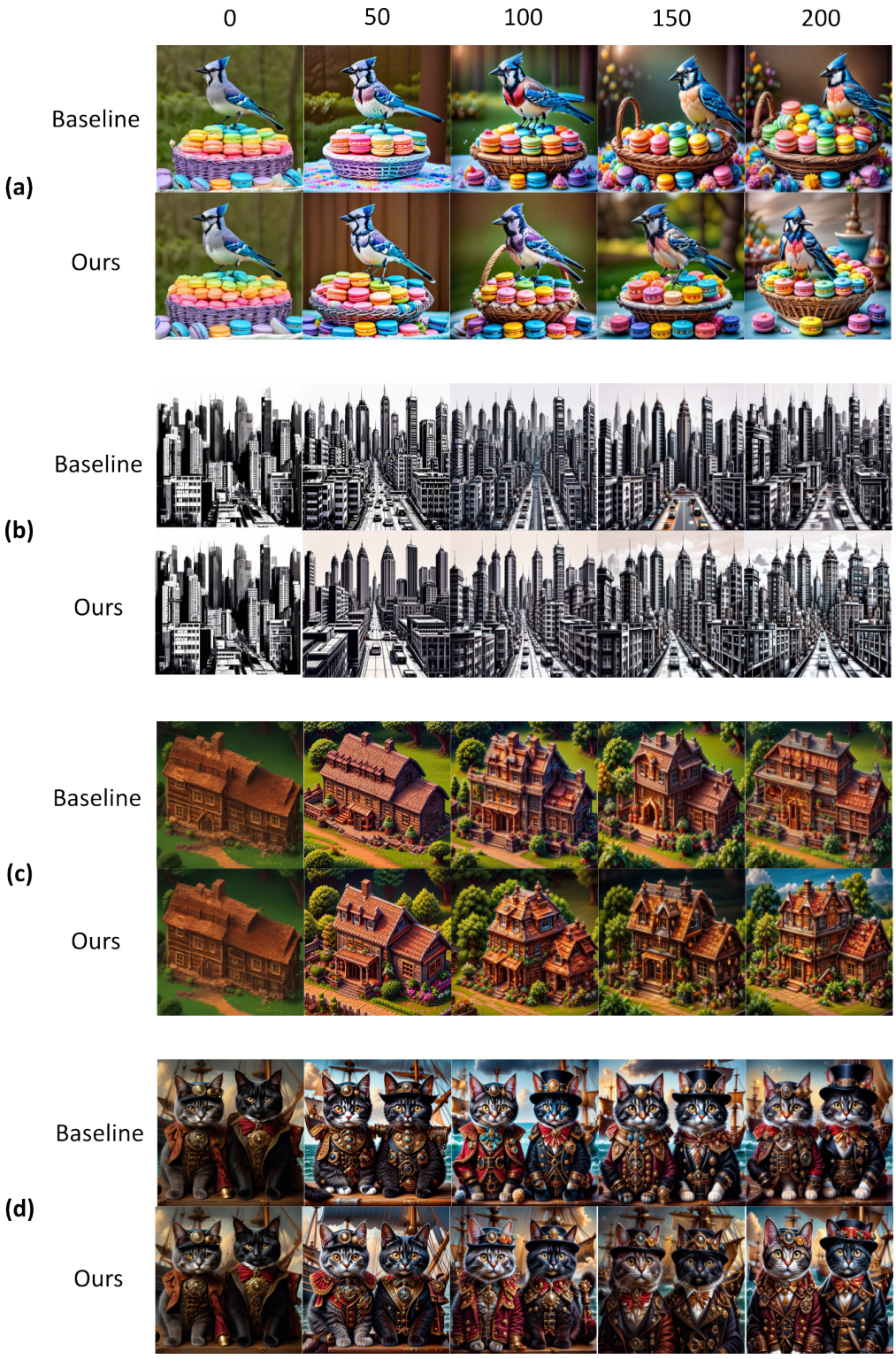

Figure 13: **Qualitative comparisons on the First-order, SD3.5-M Validation Setting.** Images are shown at checkpoints after 0, 50, 100, 150, 200 training epochs. Prompts: *(a) A blue jay standing on a large basket of rainbow macarons; (b) an illustration of monochrome cityscape vector graphic;(c) isometric style farmhouse from RPG game, unreal engine, vibrant, beautiful, crisp detailed, ultradetailed, intricate; (d) Two cats, one grey and one black are wearing steampunk attire and standing in front of a ship in a heavily detailed painting.*



Figure 14: **Qualitative comparisons on the Zeroth-order, SD1.5 Validation Setting.** Images are shown at checkpoints after 0, 50, 100, 150, 200, 250 training epochs. Prompts: *(a) A photograph of a giant diamond gem in the ocean, featuring vibrant colors and detailed textures; (b) logo of mountain, hike, modern, colorful, rounded, 2d concept; (c) A colorful tin toy robot runs a steam engine on a path near a beautiful flower meadow in the Swiss Alps with a mountain panorama in the background; (d) A mechanical planet amidst a space with exploding stars.*



Figure 15: **Qualitative results on the Zeroth-order, SD3.5-M Validation Setting.** Images are shown at checkpoints after 0, 60, 120, 180, 240, 300, 360, 420 training epochs. Prompts: *(a) a photo of a brown bed and a pink cell phone; (b) a photo of a cat below a backpack; (c) a photo of a green couch and an orange umbrella; (d) a photo of a refrigerator above a baseball bat; (e) a photo of three donuts.*

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The paper unifies the theory of reward-based fine-tuning of flow-based models, and demonstrates empirical improvements.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in Section 6

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Proofs are in Appendices.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We dedicate a section in the Appendix to all experiment details required to reproduce results.

   Guidelines:

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

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

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

Answer: [No]

Justification: Code and reproduction scripts are not included at submission time. We intend to release code and detailed instructions at camera-ready.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We dedicate a section in the Appendix to all experiment details required to reproduce results. If not specified, they use the same settings as the baseline papers cited.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Following common practice in existing literature, we run experiments on a wide suite of settings; encompassing multiple architectures (SD3.5-M, SD1.5), reward functions (Aesthetic Score, GenEval, HPSv2.1), and baselines (Residual ∇-DB, VGG-Flow, PCPO, TempFlow-GRPO), rather than reporting error bars on a small set of settings.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We specify the number and type of GPUs, and GPU hours for each experiment.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research conforms with the Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Yes, in Section 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Not applicable for our case.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We credit and cite owners of original assets.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [No]

    Justification: Code and reproduction scripts are not included at submission time. We intend to release well-documented code with detailed instructions at camera-ready.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: We do not crowdsource or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This research does not require IRB approvals or equivalents.

    Guidelines:

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

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were only used for writing, editing, formatting, and proofreading.

    Guidelines:

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