Reinforce Adjoint Matching

Scaling RL Post-Training of Diffusion and Flow-Matching Models

Andreas Bergmeister[1], **Stefanie Jegelka**[1,2], **Nikolas Nüsken**[3]
Carles Domingo-Enrich[4,†], **Jakiw Pidstrigach**[5,†]

[1]TU Munich, MCML, [2]MIT CSAIL, [3]King's College London, [4]Microsoft Research New England, [5]University of Oxford [†]Joint last authors

Diffusion and flow-matching models scale because pretraining is supervised regression: a clean sample is noised analytically, and a model regresses against a closed-form target. RL post-training aligns the model with a reward. In image generation, this makes samples compose objects correctly, render text legibly, and match human preferences. Existing methods rely on costly SDE rollouts, reward gradients, or surrogate losses, sacrificing pretraining's regression structure. We show that the structure extends to RL post-training. Under KL-regularized reward maximization, the optimal generative process tilts the clean-endpoint distribution towards samples with higher reward and leaves the noising law unchanged. Combining this with the adjoint-matching optimality condition and a REINFORCE identity, we derive Reinforce Adjoint Matching (RAM): a consistency loss that corrects the pretraining target with the reward. At each step, we draw a clean endpoint from the current model, evaluate its reward, noise it as in pretraining, and regress. No SDE rollouts, backward adjoint sweeps, or reward gradients are required. Like the pretraining objective, RAM is simple and scales. On Stable Diffusion 3.5M, RAM achieves the highest reward on composability, text rendering, and human preference, reaching Flow-GRPO's peak reward in up to 50× fewer training steps.

Code: https://github.com/AndreasBergmeister/ram

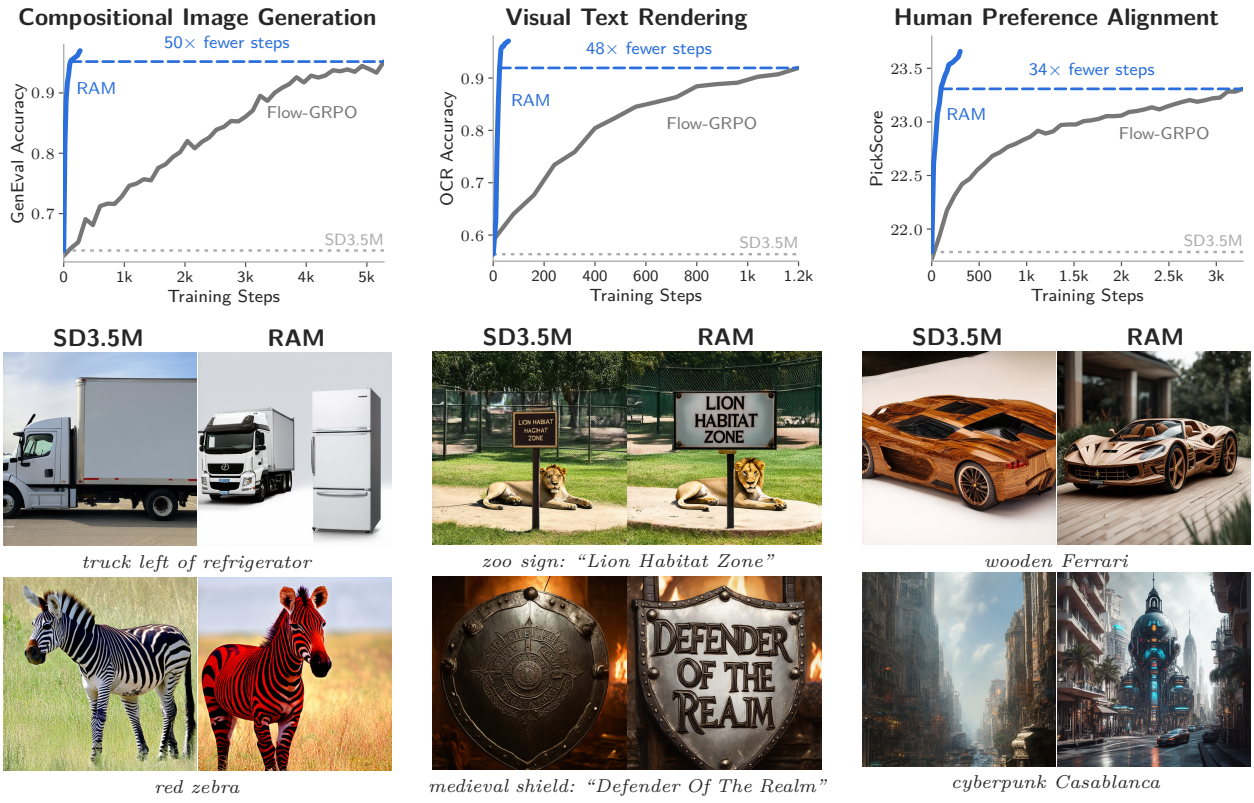

Figure 1 *Top: RAM reaches Flow-GRPO's peak training reward in up to 50× fewer training steps (with even slightly lower per-step compute). Bottom: RAM improves SD3.5M generations on composability, text rendering, and human preference.*

1 Introduction

Diffusion and flow-matching models dominate high-fidelity generation in continuous domains (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Lipman et al., 2023; Liu et al., 2023). They scale because pretraining is supervised regression. A clean sample from the dataset is corrupted by Gaussian noise, and a model regresses against a closed-form target. That is the entire training procedure.

We often want to optimize generation towards a reward function, rather than just matching the data distribution. In image generation, this is what makes a model follow complex prompts (Ghosh et al., 2023), render text legibly, or align with human preferences (Xu et al., 2023). In language modeling, reinforcement learning (RL) post-training is what produces today's reasoning systems.

Given a pretrained model and a reward function, the canonical post-training objective is to maximize expected reward while staying close to the pretrained model in a KL sense. For models with tractable likelihoods, such as language models, policy gradients optimize this objective using the log-likelihood of sampled outputs. In diffusion and flow-matching models, sample likelihoods are intractable, so post-training is more challenging.

Existing methods pay a different price to work around this. Policy-gradient methods make the denoising process stochastic and use Gaussian transition densities to optimize a variational lower bound (Black et al., 2024; Fan et al., 2023; Liu et al., 2025a). Stochastic optimal control methods target the KL-regularized objective exactly, but current estimators require reward gradients backpropagated through the sampling process via an adjoint ODE (Uehara et al., 2024; Domingo-Enrich et al., 2025). Both approaches sample by SDE rollout, which requires many steps, and for flow-matching models the noise schedule diverges near the noisy end. To sidestep this, recent methods rely on surrogate objectives: Xue et al. (2025) replace the intractable likelihood with a flow-matching ELBO, and Zheng et al. (2026) distill an improvement direction by contrasting positive and negative endpoints.

We show that the regression structure of pretraining extends to RL post-training. Casting post-training as a stochastic optimal-control problem gives a fixed-point condition for the optimal control (Domingo-Enrich et al., 2025). From this condition and a REINFORCE identity, we derive a reward-corrected regression target for the model. To scale this target, we use a key structural property of the optimum: it changes which clean samples are likely, but not how a fixed clean sample is noised. The rule that generated training pairs at pretraining is the same at the optimum. This lets us sample an endpoint from the current model using any off-the-shelf sampler, evaluate its reward, and noise it analytically as in pretraining. We reuse each endpoint for multiple noisy samples, amortizing the cost of sampling and reward evaluation. Our method, Reinforce Adjoint Matching (RAM), uses no SDE rollouts, no backward adjoint sweeps, and no reward gradients.

We post-train Stable Diffusion 3.5M on compositional generation (GenEval), visual text rendering (OCR), and human-preference alignment (PickScore). RAM achieves the highest reward on each task without reward hacking or visible quality degradation. It matches Flow-GRPO's peak reward in 50×, 48×, and 34× fewer training steps, at slightly lower per-step compute cost. The simple regression that scales pretraining now scales post-training too.

2 Diffusion and Flow-Matching Models

Diffusion and flow-matching models share a common structure: a *forward* process that corrupts data to noise from time t=0 to t=1, and a *backward* process that reverses it. Diffusion models construct both processes explicitly. Flow-matching models specify only the noising kernel and generate by integrating the velocity field (the probability-flow ODE). The associated forward and backward SDEs are nonetheless well-defined.

Forward noising. We linearly interpolate between a data distribution p_0 on $\mathbb{R}^d$ at t=0 and the Gaussian prior $\mathcal{N}(0, I)$ at t=1 (Liu et al., 2023; Lipman et al., 2023; Esser et al., 2024). Non-linear interpolations recover other classical schedules such as DDPM (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2021), and all results in this paper generalize. The interpolation defines a *noising kernel* that sends a clean sample $X_0 \sim p_0$ to a noisy version at time t,

$$X_t \mid X_0 \;\sim\; \mathcal{N}\big((1-t)\,X_0,\, t^2 I\big). \tag{1}$$

The kernel is realized by the linear forward SDE

$$\mathrm{d}X_t = \kappa_t\, X_t\, \mathrm{d}t + \sigma_t\, \mathrm{d}B_t, \qquad X_0 \sim p_0, \tag{2}$$

with $\kappa_t = -1/(1-t)$, $\sigma_t^2 = 2t/(1-t)$, and B_t a standard Brownian motion on $\mathbb{R}^d$. We write p_t for the marginal density of X_t.

Velocity matching. The standard choice in large-scale generative modeling (Liu et al., 2023; Esser et al., 2024) is to learn a model v^θ of the *velocity field*

$$v_t(x) = \mathbb{E}[\epsilon - X_0 \mid X_t = x]\,, \qquad X_0 \sim p_0, \quad \epsilon \sim \mathcal{N}(0, I). \tag{3}$$

By the tower property, regressing v_t^θ against the random target inside (3) recovers the conditional expectation. Because X_t is an affine function of X_0 and ϵ, each training pair is cheap to construct from a data sample and independent noise. The flow-matching loss is

$$\begin{aligned} \mathcal{L}_{\mathrm{FM}}(\theta) &= \mathbb{E}_t\Big[\big\|v_t^\theta(X_t) - (\epsilon - X_0)\big\|^2\Big]\,, \\ \text{where } X_0 \sim p_0, &\quad \epsilon \sim \mathcal{N}(0, I), \quad X_t = (1-t)\, X_0 + t\epsilon. \end{aligned} \tag{4}$$

Alternative parametrizations that predict the clean data or the noise instead are equivalent: given X_t, any two of the velocity, clean data, and noise determine the third.

Backward generation. The time reversal of the forward SDE (2) (Anderson, 1982) is

$$\mathrm{d}X_t = \big(\kappa_t X_t - \sigma_t^2 \nabla_x \log p_t(X_t)\big)\mathrm{d}t + \sigma_t\, \mathrm{d}B_t, \qquad X_1 \sim \mathcal{N}(0, I), \tag{5}$$

which we integrate from $t{=}1$ to $t{=}0$ to sample from p_0. The score is related to the velocity by

$$\nabla_x \log p_t(x) = \frac{2}{\sigma_t^2}\big(\kappa_t x - v_t(x)\big), \tag{6}$$

so a learned velocity model v^θ suffices to sample. More generally, replacing the noise coefficient in (5) by any other schedule $\tilde{\sigma}_t \geq 0$ (with correspondingly adjusted drift) preserves the marginals p_t (Albergo et al., 2025; Song et al., 2021). The null choice $\tilde{\sigma} \equiv 0$ recovers the probability-flow ODE $\mathrm{d}X_t = v_t(X_t)\,\mathrm{d}t$, which is the default sampler for flow-matching models.

3 RL Post-Training as Optimal Control

Given a pretrained generative model with endpoint distribution p_0^{ref} and a scalar reward function $r : \mathbb{R}^d \to \mathbb{R}$, the canonical KL-regularized target is the *tilted distribution*

$$p_0^* = \arg\max_p \big\{\mathbb{E}_{x\sim p}[r(x)] - D_{\mathrm{KL}}\big(p \| p_0^{\mathrm{ref}}\big)\big\} \;\propto\; p_0^{\mathrm{ref}}(x)\exp\big(r(x)\big). \tag{7}$$

The KL term keeps the post-trained model close to the pretrained reference and within its support. The reward shifts mass toward desirable samples. The regularization strength is controlled by scaling the reward, which we leave implicit for notational simplicity.

Stochastic optimal control. Sampling directly from p_0^* is intractable. We instead steer the pretrained backward SDE toward it with a drift correction u_t,

$$\mathrm{d}X_t^u = \big(b_t^{\mathrm{ref}}(X_t^u) + \sigma_t u_t(X_t^u)\big)\mathrm{d}t + \sigma_t\, \mathrm{d}B_t, \qquad X_1^u \sim \mathcal{N}(0, I), \tag{8}$$

where $b_t^{\mathrm{ref}}(x) = \kappa_t x - \sigma_t^2 \nabla_x \log p_t^{\mathrm{ref}}(x)$ is the drift of the reference backward process (5). We choose u to maximize the terminal reward minus a quadratic control cost,

$$\max_u\; \mathbb{E}\left[r(X_0^u) - \frac{1}{2}\int_0^1 \|u_\tau(X_\tau^u)\|^2\, \mathrm{d}\tau\right]. \tag{9}$$

By Girsanov's theorem, this stochastic optimal control problem is equivalent to KL-regularized reward maximization in path space (Section A.1).

Structure of the optimal process. The next theorem is a standard consequence of stochastic optimal control theory (Pham, 2009): optimality is equivalent to tilting the clean-endpoint distribution to p_0^* while leaving the conditional law of noisy states given that clean endpoint unchanged. This recovers the memoryless-schedule result of Domingo-Enrich et al. (2025) as the marginal-time statement of a full path-space characterization.

Theorem 3.1 (Optimal controlled process). *Under standard regularity assumptions (Section A.3), a controlled process X^u solves (9) if and only if*

$$X_0^u \sim p_0^* \quad \text{and} \quad \mathrm{Law}\big((X_t^u)_{t\in[0,1]} \,\big|\, X_0^u = x_0\big) = \mathrm{Law}\big((X_t)_{t\in[0,1]} \,\big|\, X_0 = x_0\big) \tag{10}$$

for p_0^*-almost every x_0. Equivalently, the optimal controlled process is the time reversal of the forward noising (2) started from p_0^*.

The theorem is the precise sense in which RL post-training changes *which* clean samples are preferred, but not the noising rule that connects a fixed clean sample to its noisy versions. The reward tilt acts only on X_0, and the conditional trajectory law given X_0 depends only on the interpolation schedule, not on the data distribution.

Two consequences follow. First, the conditional law of a noisy state given the clean endpoint is the same analytic Gaussian kernel as in pretraining (1). Only the endpoint distribution changes, from p_0 to p_0^*. Second, the optimal marginal p_t^*, velocity v_t^*, and score $\nabla_x \log p_t^*$ take the same functional forms as (3) and (6), with p_0^* in place of p_0.

Adjoint matching. To reach this fixed point in practice, we introduce the Bellman value function

$$V_t^u(x) = \mathbb{E}\left[r(X_0^u) - \frac{1}{2}\int_0^t \|u_\tau(X_\tau^u)\|^2 \,\mathrm{d}\tau \;\middle|\; X_t^u = x \right], \tag{11}$$

and its spatial gradient $A_t^u(x) \coloneqq \nabla_x V_t^u(x)$, which we call the *adjoint*. The following verification result turns optimal control into a self-consistency condition on u.

Theorem 3.2 (Adjoint matching). *Under standard regularity assumptions (Section A.5), a control u is optimal if and only if*

$$u_t(x) = -\sigma_t A_t^u(x) \qquad \text{for all } (t,x) \in [0,1]\times\mathbb{R}^d. \tag{12}$$

Following Domingo-Enrich et al. (2025), we turn this fixed-point condition into on-policy training: estimate the adjoint A_t^u under the current control, and regress u_t against the resulting target. In practice we parametrize the velocity field rather than u directly, so we derive a regression target for v_t^θ via the control–velocity relation (16).

4 Reinforce Adjoint Matching

Reinforce Adjoint Matching (RAM) is an on-policy consistency loss for the velocity field that requires no reward gradients. By linearity of expectation, the value function (11) splits into an endpoint-reward term and an integrated running-cost term. Differentiating the endpoint term in x naively produces a reward gradient ∇r, which we avoid since r may be non-differentiable. Applying the log-derivative (REINFORCE) identity replaces this gradient with the backward bridge score weighted by the reward, yielding the exact decomposition

$$A_t^u(x) = \underbrace{\mathbb{E}\Big[r(X_0^u)\nabla_x \log p_{0|t}^u(X_0^u \mid x) \;\Big|\; X_t^u = x \Big]}_{\text{reward term}} - \frac{1}{2}\underbrace{\mathbb{E}\left[\nabla_x \int_0^t \|u_\tau(X_\tau^u)\|^2 \,\mathrm{d}\tau \;\middle|\; X_t^u = x \right]}_{\text{path-cost correction}}. \tag{13}$$

The reward term admits a closed-form Monte Carlo estimator, which we develop below. For the path-cost correction, Section 5 presents several estimators trading off exactness, variance, and computational cost. At

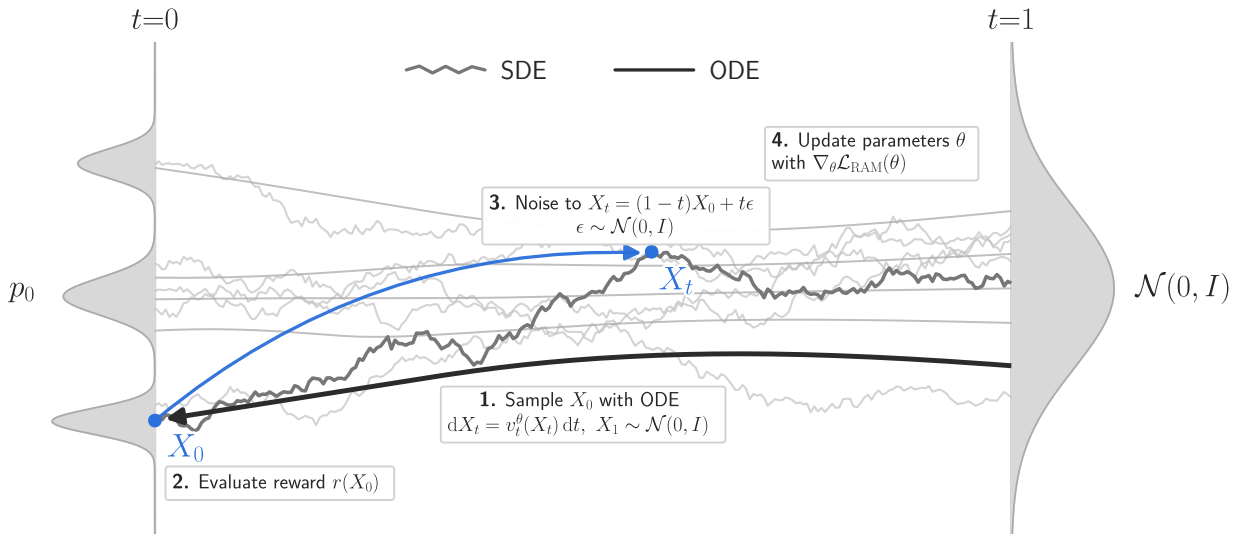

Figure 2 RAM training. *We draw a clean endpoint with an ODE sampler, evaluate its reward, and noise it as in pretraining to obtain training states. This avoids SDE rollouts, and reusing each endpoint across many noise draws amortizes the cost of sampling and reward evaluation.*

image scale, either variance is too high or the estimator requires a backward adjoint sweep along a stored SDE rollout, bringing numerical instability and substantial per-step compute. We therefore use the approximation

$$A_t^u(x) \approx \mathbb{E}\Big[r(X_0^u)\nabla_x \log p^u_{0|t}(X_0^u \mid x) \;\Big|\; X_t^u = x\Big]. \tag{14}$$

This approximation coincides with the full adjoint at initialization, where $u{\equiv}0$ makes the path cost vanish, and exactly for a Gaussian reference under a linear reward (Section B.4). Section 4.1 relates the resulting RAM fixed point to the KL-regularized optimum more generally. The pretrained reference remains an explicit anchor in the regression target (17), penalizing deviation from v^{ref} throughout training. In practice, this prevents reward hacking despite RAM achieving the highest task reward among all baselines (Section 6).

Endpoint sampling and analytic noising. Equation (14) requires joint samples (X_0^u, X_t^u). In principle these come from simulating the controlled SDE (8), which requires many steps and is numerically unstable for flow-matching models. Theorem 3.1 opens up a much cheaper route: at the optimum, the bridge from clean endpoint to noisy state is the pretraining kernel (1). We therefore draw an on-policy endpoint X_0 with any off-the-shelf ODE sampler and noise it: $X_t = (1-t)X_0 + t\epsilon$.

This recovers the controlled-SDE joint (X_0^u, X_t^u) exactly at initialization and at the optimum. Since the training objective is a fixed-point condition, exactness is formally needed only at the optimum. In practice, endpoint sampling works well throughout training. Each endpoint then yields K independent training states at the cost of one model sample and one reward query (Figure 2 and Algorithm 1). Where existing SDE-based methods produce correlated states from a single rollout, RAM's K states are conditionally independent given the endpoint. This translates into more gradient signal per step (Section 6).

Bayes bridge score. The reward term in (14) requires the backward bridge score. By Bayes' rule, and noting that $p_0(x_0)$ is constant in x_t, $\nabla_{x_t}\log p_{0|t}(x_0 \mid x_t) = \nabla_{x_t}\log p_{t|0}(x_t \mid x_0) - \nabla_{x_t}\log p_t(x_t)$. The forward-bridge score is analytic from (1) and the marginal score follows from (6), giving a closed-form expression in v_t.

Proposition 4.1 (Bayes bridge score). *For the noising kernel (1), velocity field (3), $0 < t \le 1$, and $\epsilon := (x_t - (1-t)x_0)/t$,*

$$\nabla_{x_t}\log p_{0|t}(x_0 \mid x_t) = \frac{1-t}{t}\big(v_t(x_t) - (\epsilon - x_0)\big). \tag{15}$$

Algorithm 1 RAM

Inputs: parameters θ initialized from the pretrained reference field v^{ref}, reward r, targets per endpoint K

while not converged **do**
  sample x_0 from p_0^θ ▷ any sampler
  $r_0 \leftarrow r(x_0)$
  for $k = 1, \dots, K$ **do in parallel**
    sample time t_k and noise $\epsilon_k \sim \mathcal{N}(0, I)$
    $x_k \leftarrow (1 - t_k)x_0 + t_k\epsilon_k$
    $\hat{v}_k \leftarrow v_{t_k}^{\mathrm{ref}}(x_k) + r_0\big((\epsilon_k - x_0) - v_{t_k}^\theta(x_k)\big)$
  end for
  update θ with $\nabla_\theta \frac{1}{K} \sum_{k=1}^K \| v_{t_k}^\theta(x_k) - \mathrm{sg}(\hat{v}_k) \|^2$ ▷ sg stops gradients
end while

The full algebra is given in Section B.1. The identity is exact for the uncontrolled reference process at initialization, and by Theorem 3.1 it also holds for the optimal process. Away from those points we substitute v_t^θ for v_t as a plug-in approximation.

RAM objective. We train the velocity field v^θ rather than parametrizing u_t directly, initializing $v^\theta = v^{\mathrm{ref}}$. The two are related by

$$\sigma_t u_t = 2\big(v_t^\theta - v_t^{\mathrm{ref}}\big), \tag{16}$$

matching the true control–velocity relation at the optimum (Section A.4). Substituting (14) and (15) into the adjoint-matching condition (12) gives a Monte Carlo target for the velocity field. The time-dependent prefactors cancel (algebra in Section B.2), yielding the RAM loss

$$\mathcal{L}_{\mathrm{RAM}}(\theta) = \mathbb{E}_t\Big[\big\| v_t^\theta(X_t) - \mathrm{sg}\big(v_t^{\mathrm{ref}}(X_t) + r(X_0)\big((\epsilon - X_0) - v_t^\theta(X_t)\big)\big)\big\|^2\Big], \quad \text{where } X_0 \sim p_0^\theta, \quad \epsilon \sim \mathcal{N}(0, I), \quad X_t = (1-t)X_0 + t\epsilon. \tag{17}$$

Here $\mathrm{sg}(\cdot)$ is the stop-gradient operator, and we also treat the endpoint sampling as constant, so gradients are taken only with respect to the leading $v_t^\theta(X_t)$ term. A natural alternative is to form the regression loss in control space and convert via (16). This introduces a factor $4/\sigma_t^2$ that downweights the early timesteps that shape the trajectory. Avoiding this scaling was necessary for stable training (Section B.2).

4.1 Relating the RAM fixed point to the KL optimum

The training objective (17) implies the fixed-point equation

$$v_t^\theta(x) - v_t^{\mathrm{ref}}(x) = \mathbb{E}\big[r(X_0)\big((\epsilon - X_0) - v_t^\theta(x)\big) \,\big|\, X_t = x\big], \quad X_0 \sim p_0^\theta, \tag{18}$$

where p_0^θ is the endpoint distribution generated by integrating the ODE under v_t^θ. To relate this self-consistency condition to the KL-regularized optimum, we connect the reference and the optimum through a path of exponentially tilted endpoints.

Lemma 4.2 (Path-integral characterization of the optimum). *For $\lambda \in [0, 1]$, let $p_0^\lambda(x_0) \propto p_0^{\mathrm{ref}}(x_0)\, e^{\lambda r(x_0)}$ interpolate between $p_0^0 = p_0^{\mathrm{ref}}$ and $p_0^1 = p_0^*$, and let v_t^λ denote the velocity field for endpoint distribution p_0^λ. Then*

$$\partial_\lambda v_t^\lambda(x) = \mathbb{E}\big[r(X_0)\big((\epsilon - X_0) - v_t^\lambda(x)\big) \,\big|\, X_t = x\big], \quad X_0 \sim p_0^\lambda, \tag{19}$$

and integrating over $\lambda \in [0, 1]$ yields

$$v_t^*(x) - v_t^{\mathrm{ref}}(x) = \int_0^1 \mathbb{E}\big[r(X_0)\big((\epsilon - X_0) - v_t^\lambda(x)\big) \,\big|\, X_t = x\big]\, \mathrm{d}\lambda, \quad X_0 \sim p_0^\lambda. \tag{20}$$

The proof is given in Section B.3. Equation (18) has the same form as the integrand of (20) evaluated at the right endpoint λ=1, with the on-policy distribution p_0^θ in place of p_0^* and v_t^θ in place of v_t^*. The integrand itself is the conditional reward-velocity covariance $\mathrm{Cov}\big(r(X_0),\, \epsilon - X_0 \;\big|\; X_t = x\big)$ with $X_0 \sim p_0^\lambda$. RAM thus replaces the average of this covariance over the path by a single evaluation at λ=1. The rule is exact when v_t^λ depends linearly on λ along the path, in particular for a Gaussian reference under a linear reward (Section B.4). More generally, RAM and the KL optimum agree to first order in the reward: scaling $r \mapsto \eta r$ and expanding around η=0, both (18) and (20) reduce at leading order to $\eta\,\mathrm{Cov}\big(r(X_0),\, \epsilon - X_0 \;\big|\; X_t = x\big)$ with $X_0 \sim p_0^{\mathrm{ref}}$.

5 Retaining the Path-Cost Correction

RAM drops the path-cost correction in the adjoint decomposition (13). To show that this is the right choice for scaling to image generation, we examine the challenges of retaining it. First, we can preserve RAM's endpoint sampling and analytic noising and estimate the path cost at one random intermediate time. The price is high variance. Alternatively, we can simulate the controlled SDE (8), store the trajectory, and differentiate the running cost pathwise. The length of pathwise differentiation trades variance against compute. Table 1 summarizes the landscape.

Estimator	Samples from	Exactness	Trade-off
RAM (Section 4)	endpoint + analytic noising	biased	drops path cost
Random jump	endpoint + analytic noising	exact at init./optimum	high variance
Full-horizon w/ Bayes	SDE rollout	exact at init./optimum	pathwise VJPs
Full-horizon w/ Malliavin	SDE rollout	exact	pathwise + score VJPs
Local	SDE rollout	exact	high variance

Table 1 *Estimators for the value-function gradient A_t^u.*

Endpoint sampling with a random jump. We insert a uniformly random intermediate time $s \in [0, t)$ into the analytic noising trajectory. Then $\frac{t}{2}\|u_s(X_s)\|^2$ is an unbiased estimate of the integrated path cost. Combined with the Bayes bridge score from intermediate to training state, this yields the jump estimator

$$\widehat{A}_t^{\mathrm{jump}} = \big(r(X_0) - \tfrac{t}{2}\|u_s(X_s)\|^2\big)\nabla_{x_t} \log p_{s|t}(X_s \mid x_t)\big|_{x_t = X_t}, \quad s \sim \mathcal{U}[0, t). \tag{21}$$

The estimator is cheap and exact at initialization and at the optimum (Section C.1). The price is variance: the entire gradient signal flows through that one scalar prefactor.

Pathwise differentiation along an SDE rollout. We simulate the controlled SDE (8) and recover the path-cost gradient by integrating an adjoint ODE backward along the stored trajectory. This is functionally equivalent to differentiating through the SDE solver in autograd, but more memory efficient.

Adjoint integration over the full horizon is expensive and numerically delicate. Dynamic programming lets us replace some of the pathwise integration with a REINFORCE term: at any intermediate time s, the adjoint splits into a REINFORCE term over the prefix $[0, s]$ and a pathwise gradient over the suffix $[s, t]$.

Theorem 5.1 (Generalized adjoint). *For every $0 \le s < t \le 1$ and admissible control u,*

$$\nabla_x V_t^u(x) = \underbrace{\mathbb{E}\Big[V_s^u(X_s^u)\,\nabla_x \log p_{s|t}^u(X_s^u \mid x) \;\Big|\; X_t^u = x\Big]}_{\textit{REINFORCE on prefix}} - \frac{1}{2}\underbrace{\mathbb{E}\Big[\nabla_x \int_s^t \|u_\tau(X_\tau^u)\|^2\,\mathrm{d}\tau \;\Big|\; X_t^u = x\Big]}_{\textit{pathwise suffix}}. \tag{22}$$

The proof differentiates the recursion in x and applies the log-derivative identity to the prefix term (Section C.2).

A single-trajectory estimator of the right-hand side combines three pieces: a running prefix value (the reward minus the integrated path cost over $[0, s]$), a bridge score, and the suffix path-cost gradient. For the bridge score we can use the Bayes estimator (15), which generalizes in closed form to all $s \in [0, t)$ (Section B.1).

Alternatively, the Malliavin score of Pidstrigach et al. (2025) is exact even away from the optimum, at the cost of additional VJPs through the score.

Two values of s are computationally practical. *Full-horizon* ($s = 0$) computes targets for every t in a single backward sweep. *Local* ($s = t - \delta$) avoids the sweep, collapsing the suffix to a single discretization step. Intermediate values would require a separate sweep per target time.

5.1 Estimator variance on a 2D example

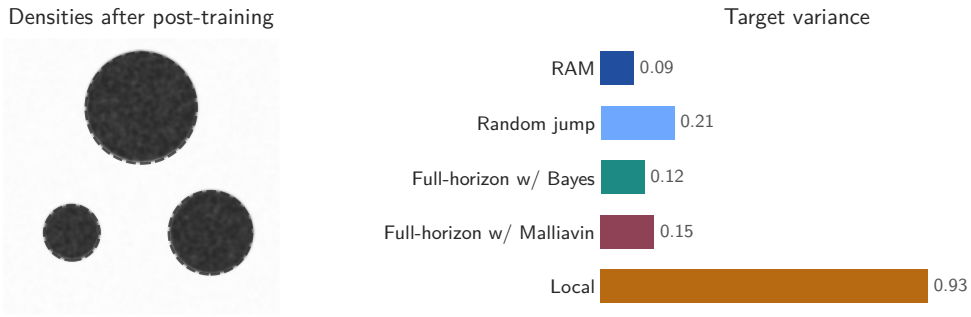

Figure 3 Path-cost-corrected estimators on a 2D toy. *Left: post-trained densities match the tilted target for every estimator. Right: regression loss at convergence, equal to the variance of the control-space target.*

We compare the estimators on a 2D rectified flow with uniform endpoint on $[-1, 1]^2$ under a reward that is positive inside three circles of varying radius. All recover the correct tilted distribution, confirming the shared fixed point. At convergence the regression loss collapses to the variance of the control-space target $\sigma_t(\widehat{A}^u_{s,t} - A^u_t)$, so the right panel reads off estimator variance directly. Among the path-cost-corrected estimators, full-horizon has substantially lower variance than local or random-jump, and the Bayes bridge score beats Malliavin within full-horizon. RAM attains the lowest variance overall: dropping the path-cost gradient eliminates a dominant source of noise. These trade-offs amplify at image scale: variance grows with dimension, and adjoint sweeps over long horizons become prohibitive. RAM is the only estimator that scales to text-to-image post-training.

6 Text-to-Image Experiments

We post-train Stable Diffusion 3.5 Medium (SD3.5M) (Esser et al., 2024), a 2.5B-parameter rectified-flow transformer, on three text-to-image reward objectives, and compare RAM against the strongest recent baselines: Flow-GRPO (Liu et al., 2025a), DiffusionNFT (Zheng et al., 2026), and AWM (Xue et al., 2025). We largely follow the evaluation setup of Liu et al. (2025a): for each reward we train a separate model on its benchmark prompts, report the training reward on held-out prompts from the same benchmark, and check for reward hacking on an independent prompt set. Flow-GRPO numbers come from the original paper, except for HPSv2, which we recompute from the authors' released checkpoints. We retrain DiffusionNFT and AWM. We train each model until the reward plateaus or starts to decrease. Qualitative samples from the pretrained model and from the models post-trained with RAM are shown in Figure 1.

Training rewards. Each benchmark comes with its own training and test prompt sets. During training we sample images for the training prompts and score each prompt-image pair under the reward function associated with that benchmark. GenEval (Ghosh et al., 2023) measures compositional correctness: for prompts such as *"a truck to the left of a refrigerator"*, pretrained vision models verify whether the required objects, attributes, and spatial relations all appear in the generated image. OCR evaluates visual text rendering via an edit-distance reward that checks whether text specified in the prompt appears legibly in the image, following Liu et al. (2025a). PickScore (Kirstain et al., 2023) is a learned human-preference model trained on large-scale pairwise image comparisons.

Image-quality evaluation. Optimizing a single reward can degrade generic image quality, a failure mode known as reward hacking. To detect this, we score each post-trained model on DrawBench (Saharia et al.,

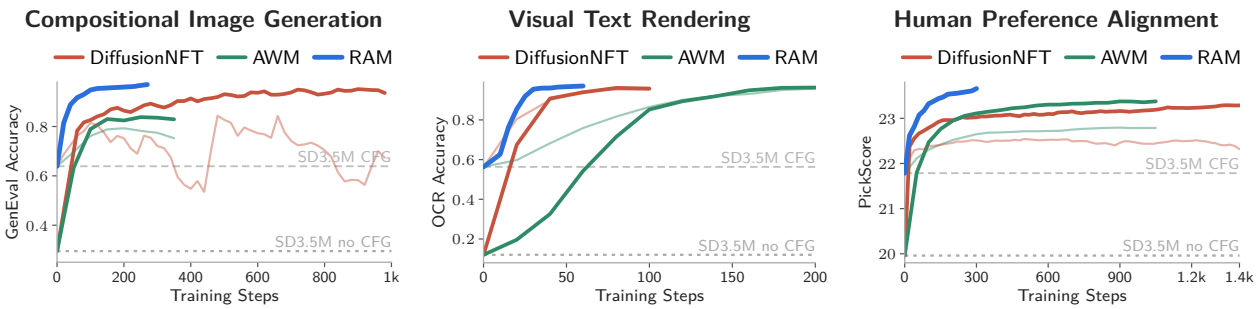

Figure 4 *Training-reward curves comparing RAM with DiffusionNFT and AWM, two methods designed for training efficiency. The lighter companion curves show DiffusionNFT and AWM re-evaluated with CFG, which lowers their task reward. RAM, by contrast, remains compatible with CFG after post-training and is plotted with CFG.*

2022) prompts disjoint from training and test under five off-the-shelf metrics: Aesthetic (Schuhmann and Beaumont, 2022) and DeQA (You et al., 2025) rate perceptual quality; ImageReward (Xu et al., 2023), HPSv2 (Wu et al., 2023), and PickScore (Kirstain et al., 2023) are learned human-preference models.

Reward normalization. Following common practice, we normalize raw rewards using group-relative advantage estimation. For each training prompt we sample a group of $G = 24$ images, evaluate their rewards, and subtract the group mean. We divide by the standard deviation pooled over all samples in the current training step, rather than per group. This stabilizes the scale across training and avoids the instability of per-group normalization when a group happens to draw nearly identical rewards.

Results. RAM achieves the highest training reward on all three tasks (Table 2). For compositional generation and preference alignment, this improvement comes with comparable or better image-quality metrics. For visual text rendering, RAM avoids the severe quality collapse of DiffusionNFT and AWM while matching or exceeding their OCR reward. Flow-GRPO scores higher than RAM on image-quality metrics for this task, but at a substantially lower OCR reward. The OCR reward signal pushes models to sacrifice aesthetic quality for legibility. Stopping RAM earlier or using a smaller reward coefficient yields higher image-quality numbers at a lower OCR reward. Qualitative samples in Figure 1 show that RAM corrects compositional errors (*GenEval*), renders specified text legibly (*OCR*), and produces more preferred images (*PickScore*) while preserving the style and fidelity of the pretrained model. Notably, RAM remains compatible with classifier-free guidance (CFG) after post-training, whereas CFG lowers the task reward of DiffusionNFT and AWM (lighter curves in Figure 4). We therefore evaluate RAM with CFG and the two baselines without.

Training efficiency. Figure 1 (top) plots training reward against training steps for RAM and Flow-GRPO on all three tasks. RAM matches Flow-GRPO's peak reward in roughly 50× fewer steps on GenEval (0.95), 48× fewer on OCR (0.92), and 34× fewer on PickScore (23.31). On GenEval, RAM reaches a reward of 0.90 in 75× fewer steps than Flow-GRPO. Per-step compute cost is comparable between the two (e.g. 0.66 GPU-hours for RAM vs. 0.70 for Flow-GRPO on GenEval), so step ratios translate directly to wall-clock training time (Section D). All methods use the same number of prompts and samples per step, so differences in step efficiency reflect how each sample is used for learning. Flow-GRPO performs one SDE rollout per sample and forms its loss from the strongly correlated noisy states along that trajectory. RAM instead draws a clean endpoint and noises it $K = 8$ times with independent noise vectors. The resulting training states are conditionally independent given the endpoint, providing more gradient signal per step. Figure 4 extends the comparison to DiffusionNFT and AWM, two methods also designed for training efficiency. These baselines amortize endpoints across multiple noise draws as well, so RAM's edge over them comes from its regression target: a closed-form adjoint-matching condition, whereas DiffusionNFT relies on a positive/negative-guidance surrogate and AWM on a flow-matching ELBO. The AWM numbers we report are lower than in the original publication, which uses 72 prompts per training step. We use 48 uniformly across all methods for a fair comparison.

Table 2 *Stable Diffusion 3.5 Medium (SD3.5M) post-training results. We report training rewards on held-out test prompts and image-quality metrics on DrawBench prompts. Higher is better for all metrics. Within each task, the best results are shown in* ***bold*** *and second-best results are underlined. † denotes results obtained by retraining the corresponding baseline.*

Model	# Steps	Training Reward			Image Quality Metrics				
		GenEval	OCR	PickScore	Aesthetic	DeQA	ImgRwd	HPSv2	PickScore
SD3.5M		0.64	0.56	21.79	5.41	4.08	0.82	0.28	22.40
Compositional Image Generation									
Flow-GRPO	> 5k	0.95			5.25	4.01	1.03	0.27	22.37
AWM†	300	0.83			5.14	3.75	0.67	0.24	22.04
DiffusionNFT†	900	0.95			4.98	**4.10**	0.30	0.24	21.59
RAM	270	**0.97**			**5.38**	4.09	**1.19**	**0.29**	**22.52**
Visual Text Rendering									
Flow-GRPO	1.2k		0.92		**5.32**	**4.06**	**0.95**	**0.28**	**22.44**
AWM†	200		**0.97**		5.01	2.83	-0.85	0.18	20.56
DiffusionNFT†	100		0.96		4.87	3.01	-0.97	0.18	20.26
RAM	60		**0.97**		5.23	3.90	0.44	0.26	21.83
Human Preference Alignment									
Flow-GRPO	> 3k			23.31	5.92	**4.22**	1.28	**0.32**	23.53
AWM†	1k			23.39	**6.31**	4.10	1.27	0.31	23.76
DiffusionNFT†	1.4k			23.29	6.16	4.13	1.23	0.31	23.65
RAM	300			**23.67**	6.11	4.17	**1.36**	**0.32**	**23.95**

7 Related Work

Diffusion and flow-matching pretraining. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) and the score-based SDE formulation (Song et al., 2021) established regression against analytically noised data as the standard pretraining recipe. Stochastic interpolants and flow matching unify deterministic flows and stochastic diffusions under the same velocity regression template (Albergo et al., 2025; Lipman et al., 2023), and rectified flow (Liu et al., 2023; Esser et al., 2024) has become the default parametrization for large-scale image and video models. RAM reuses this regression structure, now with a target built from the current model and a scalar reward.

Policy-gradient RL post-training. Trajectory-level policy gradients were first applied to diffusion RL post-training by Black et al. (2024), with KL-regularized variants in DPOK (Fan et al., 2023). Subsequent work scales this view to large text-to-image models and to flow matching, including Flow-GRPO (Liu et al., 2025a) and continuous-time score-as-control formulations (Zhao et al., 2025), with further refinements through dense credit assignment or exploration (Deng et al., 2024; Chae et al., 2025; Yang et al., 2024b; Liu et al., 2025b). Preference-supervised variants replace scalar rewards with pairwise comparisons (Wallace et al., 2024; Yang et al., 2024a; Kim et al., 2024). All of these treat denoising as a generic Markov decision process and learn from stochastic rollouts, with noise schedules tuned empirically or hybridized with ODE solvers (Deng et al., 2026). None exploit the analytic noising structure that makes pretraining cheap. RAM stays in the black-box reward setting, but replaces the policy-gradient rollout with a pretraining-like regression against a closed-form target.

Stochastic optimal control and adjoint methods. KL-regularized reward maximization in continuous time is naturally an entropy-regularized stochastic optimal control problem (Todorov, 2006). ELEGANT (Uehara et al., 2024) develops this perspective for diffusion post-training, with an emphasis on avoiding reward collapse. Adjoint Matching (Domingo-Enrich et al., 2025) introduces the memoryless noise schedule that turns the Bellman fixed point into an on-policy regression of the adjoint. Adjoint Sampling (Havens et al., 2025) extends

this perspective beyond RL post-training to sampling from unnormalized densities. Like RAM, it avoids SDE rollouts during training by sampling an endpoint and noising it analytically. These methods target the KL-regularized objective directly, but their estimators require reward gradients and an adjoint ODE integrated backward along a stored SDE trajectory. RAM sits most directly in the Adjoint Matching line and sharpens its path-space picture: the optimum tilts only the clean-endpoint distribution while preserving the conditional noising law given that endpoint. This justifies endpoint sampling with analytic noising, and yields a closed-form Bayes bridge target in place of a pathwise adjoint sweep. This is what makes RAM substantially more efficient than Adjoint Matching: training avoids both the SDE rollout and the backward adjoint sweep. The closely related Tilt Matching (Potaptchik et al., 2025) arrives at the same equation for how the velocity field changes as the reward tilt increases, and is likewise free of reward gradients and trajectory backpropagation. It reaches the tilted target by annealing through intermediate models. Our adjoint-matching formulation instead targets the fixed point directly.

Pretraining-like regression objectives. A recent line aligns RL post-training with the supervised structure of pretraining. DiffusionNFT (Zheng et al., 2026) defines an improvement direction by contrasting positive and negative endpoint distributions, then distills the resulting guidance into a single model with tunable strength. AWM (Xue et al., 2025) starts from a GRPO objective over clean endpoints, replaces the intractable sequence likelihood by a score/flow-matching ELBO, and adds a velocity-space KL penalty. Choi et al. (2026) further argue empirically that the likelihood estimator matters more than the particular outer loss. RAM instead starts from the KL-regularized control problem, giving the update a direct control interpretation rather than a guidance-distillation or likelihood-surrogate interpretation. The principled derivation also yields a simpler objective in practice.

Differentiable-reward methods. When rewards are differentiable, end-to-end backpropagation through the sampler provides a direct training signal. ImageReward introduces a learned human-preference scorer and the ReFL algorithm (Xu et al., 2023). AlignProp (Prabhudesai et al., 2023), DRaFT (Clark et al., 2024), and DRTune (Wu et al., 2024) explore full- or partial-chain reward backpropagation and the resulting memory/stability trade-offs. These methods rely on reward gradients backpropagated through the sampling chain or a truncated segment of it. RAM instead targets black-box rewards and keeps the KL-regularized control derivation.

Bridge estimators for controlled processes. Recent work estimates bridge scores under general controlled processes using Malliavin calculus or control self-consistency (Pidstrigach et al., 2025; Howard et al., 2025), building on classical Malliavin representations of scores and sensitivities (Lehec, 2013; Baudoin, 2002). The path-cost-corrected estimators we introduce in Section 5 connect to this literature, while the main RAM objective specializes to a cheaper Bayes bridge approximation tailored to large-scale RL post-training.

Flow-map reward alignment. Recently, reward alignment has been applied to flow maps (Boffi et al., 2024), which enable few-step generation. Meta Flow Maps (Potaptchik et al., 2026) and Diamond Maps (Holderrieth et al., 2026) align such models to a reward, but require reward gradients. We believe that extending RAM to flow maps, while keeping its gradient-free regression structure, is a promising direction for future work.

Acknowledgments

We acknowledge support from the Alexander von Humboldt Foundation. We thank the Leibniz Supercomputing Centre (LRZ) for providing computational resources, and Lennart Redl for feedback on the writing.

References

Michael S 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.

Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3): 313–326, 1982.

Fabrice Baudoin. Conditioned stochastic differential equations: theory, examples and application to finance. *Stochastic Processes and their Applications*, 100(1-2):109–145, 2002.

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.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

Daewon Chae, June Suk Choi, Jinkyu Kim, and Kimin Lee. Diffexp: Efficient exploration in reward fine-tuning for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15696–15703, 2025.

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. *arXiv preprint arXiv:2602.04663*, 2026.

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.

Fei Deng, Qifei Wang, Wei Wei, Tingbo Hou, and Matthias Grundmann. PRDP: Proximal reward difference prediction for large-scale reward finetuning of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7423–7433, 2024.

Haoyou Deng, Keyu Yan, Chaojie Mao, Xiang Wang, Yu Liu, Changxin Gao, and Nong Sang. DenseGRPO: From sparse to dense reward for flow matching model alignment. In *The Fourteenth International Conference on Learning Representations*, 2026.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12606–12633. PMLR, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pages 79858–79885, 2023.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Aaron J Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Daniel S. Levine, Brandon M Wood, Bin Hu, Brandon Amos, Brian Karrer, Xiang Fu, Guan-Horng Liu, and Ricky T. Q. Chen. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 22204–22237. PMLR, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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. *arXiv preprint arXiv:2602.05993*, 2026.

Samuel Howard, Nikolas Nüsken, and Jakiw Pidstrigach. Control consistency losses for diffusion bridges. *arXiv preprint arXiv:2512.05070*, 2025.

Minu Kim, Yongsik Lee, Sehyeok Kang, Jihwan Oh, Song Chong, and Se-Young Yun. Preference alignment with flow matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Joseph Lehec. Representation formula for the entropy and functional inequalities. *Annales de l'IHP Probabilités et statistiques*, 49(3):885–899, 2013.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

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 *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

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*, 2025b.

Huyên Pham. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer Science & Business Media, 2009.

Jakiw Pidstrigach, Elizabeth Louise Baker, Carles Domingo-Enrich, George Deligiannidis, and Nikolas Nüsken. Conditioning diffusions using malliavin calculus. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 49292–49315. PMLR, 2025.

Peter Potaptchik, Cheuk-Kit Lee, and Michael S. Albergo. Tilt matching for scalable sampling and fine-tuning. *arXiv preprint arXiv:2512.21829*, 2025.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S. Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment. *arXiv preprint arXiv:2601.14430*, 2026.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494, 2022.

Christoph Schuhmann and Romain Beaumont. Laion-aesthetics. laion.ai, 2022.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *The Ninth International Conference on Learning Representations*, 2021.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in Neural Information Processing Systems*, volume 19, pages 1369–1376, 2006.

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.

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, 2024.

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

Xiaoshi Wu, Yiming Hao, Manyuan Zhang, Keqiang Sun, Zhaoyang Huang, Guanglu Song, Yu Liu, and Hongsheng Li. Deep reward supervisions for tuning text-to-image diffusion models. In *Computer Vision – ECCV 2024*, volume 15141 of *Lecture Notes in Computer Science*, pages 108–124. Springer, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv preprint arXiv:2509.25050*, 2025.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8941–8951, 2024a.

Shentao Yang, Tianqi Chen, and Mingyuan Zhou. A dense reward view on aligning text-to-image diffusion with preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55998–56032. PMLR, 2024b.

Zhiyuan You, Xin Cai, Jinjin Gu, Tianfan Xue, and Chao Dong. Teaching large language models to regress accurate image quality scores using score distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14483–14494, 2025.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 77371–77389. PMLR, 2025.

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.

A Optimal-Control Foundations

A.1 Optimal control as path-space KL regularization

Let $\mathbb{P}$ denote the path law of the reference backward process (5) and $\mathbb{P}^u$ the path law of the controlled process (8). We assume Novikov's condition, a moment bound on u ensuring that the change of measure below is well defined, and $\mathbb{E}_{\mathbb{P}}[\exp(r(X_0))] < \infty$. Girsanov's theorem relates the path laws of two diffusions that share the same noise coefficient but differ in drift; applied here it gives

$$D_{\mathrm{KL}}(\mathbb{P}^u \| \mathbb{P}) = \frac{1}{2}\mathbb{E}_{\mathbb{P}^u}\left[\int_0^1 \|u_t(X_t^u)\|^2 \,\mathrm{d}t\right]. \tag{23}$$

The stochastic optimal control problem (9) is therefore equivalent to the Gibbs variational problem

$$\sup_{\mathbb{Q} \ll \mathbb{P}} \left\{\mathbb{E}_{\mathbb{Q}}[r(X_0)] - D_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{P})\right\}, \tag{24}$$

where $\mathbb{Q} \ll \mathbb{P}$ denotes absolute continuity, so every trajectory possible under $\mathbb{Q}$ must also have positive probability under $\mathbb{P}$. The closed-form solution is the tilted path law with density ratio

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} = \frac{\exp(r(X_0))}{\mathbb{E}_{\mathbb{P}}[\exp(r(X_0))]}. \tag{25}$$

A.2 Path-law factorization

Lemma A.1 (Path-law factorization). *Let $\Pi(\cdot \mid x_0)$ denote the conditional law of the reference backward trajectory $(X_t)_{t\in[0,1]}$ given $X_0 = x_0$. Then*

$$\mathbb{P} = \int \Pi(\cdot \mid x_0) p_0^{\mathrm{ref}}(x_0)\,\mathrm{d}x_0, \tag{26}$$

and $\Pi(\cdot \mid x_0)$ does not depend on the data distribution.

Proof. Let $\Pi^{\mathrm{fwd}}(\cdot \mid x_0)$ be the conditional law of the forward process (2) given $X_0 = x_0$. Because the coefficients of (2) are deterministic, $\Pi^{\mathrm{fwd}}(\cdot \mid x_0)$ does not depend on the data distribution, and the forward path law factors as $\int \Pi^{\mathrm{fwd}}(\cdot \mid x_0) p_0^{\mathrm{ref}}(x_0)\,\mathrm{d}x_0$. The backward process (5) is the exact time reversal of the forward process (Anderson, 1982), so reversing trajectories yields a kernel $\Pi(\cdot \mid x_0)$, obtained from Π^{fwd} by time reversal, for which (26) holds. Time reversal preserves the data-independence. □

This is the path-space factorization underlying what Domingo-Enrich et al. (2025) call the memoryless property: the conditional trajectory law given X_0 carries no information about which data distribution was used. At t=1, the noising kernel (1) reduces to $\mathcal{N}(0, I)$ regardless of X_0, so the source marginal is also data-independent.

A.3 Optimal controlled process

We prove Theorem 3.1.

Proof. We prove the two directions of (10) separately.

Optimality ⇒ form. Let $\mathbb{Q}^*$ denote the optimal path law. By (25), $\mathbb{Q}^*$ tilts $\mathbb{P}$ by $\exp(r(X_0))/Z$ with $Z = \mathbb{E}_{\mathbb{P}}[\exp(r(X_0))]$. Substituting the factorization of Lemma A.1,

$$\mathbb{Q}^* = \int \Pi(\cdot \mid x_0)\frac{\exp(r(x_0))}{Z} p_0^{\mathrm{ref}}(x_0)\,\mathrm{d}x_0 = \int \Pi(\cdot \mid x_0) p_0^*(x_0)\,\mathrm{d}x_0,$$

with $p_0^*(x) \propto p_0^{\mathrm{ref}}(x)\exp(r(x))$. Since the tilt depends only on x_0, it changes only the endpoint marginal to p_0^* while leaving the conditional path law $\Pi(\cdot \mid x_0)$ unchanged, which is (10). Equivalently, the optimal process

is the time reversal of the forward process started from p_0^*, so applying the noising kernel gives the optimal time-t marginal

$$p_t^* = \mathrm{Law}\big((1-t)\,X_0 + t\epsilon\big), \qquad X_0 \sim p_0^*, \quad \epsilon \sim \mathcal{N}(0,\, I).$$

Form $\Rightarrow$ optimality. Conversely, suppose X^u is a controlled process satisfying (10). Integrating the matching conditional law against p_0^* and applying Lemma A.1,

$$\mathbb{P}^u = \int \Pi(\cdot \mid x_0) p_0^*(x_0)\, \mathrm{d}x_0.$$

The reference factorization $\mathbb{P} = \int \Pi(\cdot \mid x_0) p_0^{\mathrm{ref}}(x_0)\, \mathrm{d}x_0$ shares the same conditional kernel, so the Radon–Nikodym derivative depends only on the endpoint,

$$\frac{\mathrm{d}\mathbb{P}^u}{\mathrm{d}\mathbb{P}}(\omega) = \frac{p_0^*(X_0(\omega))}{p_0^{\mathrm{ref}}(X_0(\omega))} = \frac{\exp(r(X_0(\omega)))}{Z},$$

which is exactly the Gibbs-optimal density ratio (25), so $\mathbb{P}^u = \mathbb{Q}^*$. The Gibbs problem (24) has a strictly concave objective and hence a unique maximizer, so u solves (9). □

A.4 Optimal velocity and score identities

We verify the velocity and score identities stated after Theorem 3.1, and derive the control–velocity relation (16).

Proof. By Theorem 3.1, the optimal time-t marginal equals that of the forward process started from p_0^*, so

$$v_t^*(x) = \mathbb{E}[\epsilon - X_0 \mid X_t = x]\,, \qquad X_0 \sim p_0^*,$$

which has the same functional form as (3) with p_0^* in place of p_0. The velocity–score relation (6) carries over with p_t^* in place of p_t by the same derivation.

For the control–velocity relation, compare the drift decompositions of the reference and optimal processes,

$$b_t^{\mathrm{ref}}(x) = v_t^{\mathrm{ref}}(x) - \frac{\sigma_t^2}{2}\nabla_x \log p_t^{\mathrm{ref}}(x),$$
$$b_t^{\mathrm{ref}}(x) + \sigma_t u_t^*(x) = v_t^*(x) - \frac{\sigma_t^2}{2}\nabla_x \log p_t^*(x).$$

Subtracting gives

$$\sigma_t u_t^* = \big(v_t^* - v_t^{\mathrm{ref}}\big) - \frac{\sigma_t^2}{2}\big(\nabla_x \log p_t^* - \nabla_x \log p_t^{\mathrm{ref}}\big).$$

Substituting $\nabla_x \log p_t = 2(\kappa_t x - v_t)/\sigma_t^2$ for both marginals, the $\kappa_t x$ terms cancel and the score difference is $-2(v_t^* - v_t^{\mathrm{ref}})/\sigma_t^2$, so $\sigma_t u_t^* = 2(v_t^* - v_t^{\mathrm{ref}})$. □

A.5 Verification and self-consistency

We prove that the self-consistency condition $u_t = -\sigma_t A_t^u$ (12) is both necessary and sufficient for optimality. One can thereby certify a candidate control by checking a single PDE rather than comparing against all alternatives.

The result is standard HJB / adjoint-matching theory. We include the proof to keep the paper self-contained.

Theorem A.2 (Verification). *Assume the Hamilton–Jacobi–Bellman (HJB) equation associated with (9) admits a unique $C^{1,2}$ solution (once continuously differentiable in t, twice in x) and that the optimal control is a Markov feedback, depending on the current state x only, not on the trajectory history. Let u be an admissible control with finite expected cost whose value function V_t^u is $C^{1,2}$ in (t,x). Then*

$$u \text{ is optimal} \qquad \Longleftrightarrow \qquad u_t(x) = -\sigma_t \nabla_x V_t^u(x) \quad \text{for all } (t,x), \tag{27}$$

and in that case $X_0^u \sim p_0^*$.

Proof. The proof rests on two PDEs the value function can satisfy: a *linear* one that holds for any control, and a *nonlinear* one (the HJB equation) that singles out the optimum. The self-consistency condition is exactly what turns the first into the second.

Value-function PDE. For any admissible control u, the value function V_t^u satisfies the linear PDE

$$0 = -\partial_t V_t^u(x) + \mathcal{B}_t^u V_t^u(x) - \frac{1}{2}\|u_t(x)\|^2, \qquad V_0^u(x) = r(x), \tag{28}$$

where $\mathcal{B}_t^u$ is the backward transition generator of the controlled diffusion (8), encoding how the reverse-time drift and noise act on a test function f,

$$\mathcal{B}_t^u f(x) = -\big(b_t^{\mathrm{ref}}(x) + \sigma_t u_t(x)\big)^\top \nabla_x f(x) + \frac{\sigma_t^2}{2}\Delta_x f(x).$$

The first term captures transport by the drift, the second captures diffusion. Equation (28) is linear in V_t^u because u is fixed; it describes how the expected future value evolves along the controlled process.

Self-consistency ⇒ optimality. Suppose $u_t = -\sigma_t \nabla_x V_t^u$. Substituting this into the generator, the controlled-drift term expands as $\sigma_t^2\|\nabla_x V_t^u\|^2$, while the running cost is $\frac{1}{2}\|u_t\|^2 = \frac{\sigma_t^2}{2}\|\nabla_x V_t^u\|^2$. Substituting both into (28) cancels half of the quadratic terms and leaves

$$0 = -\partial_t V_t^u - (b_t^{\mathrm{ref}})^\top \nabla_x V_t^u + \frac{\sigma_t^2}{2}\Delta_x V_t^u + \frac{\sigma_t^2}{2}\|\nabla_x V_t^u\|^2, \tag{29}$$

which is the HJB equation for (9). Since the HJB equation has a unique solution, V^u must equal the optimal value function, so u is optimal.

Optimality ⇒ self-consistency. Let u^* be an optimal Markov control with value function $V = V^{u^*}$. The HJB equation can be written as

$$0 = -\partial_t V - (b_t^{\mathrm{ref}})^\top \nabla_x V + \frac{\sigma_t^2}{2}\Delta_x V + \sup_{a\in\mathbb{R}^d}\big\{-\sigma_t a^\top \nabla_x V - \tfrac{1}{2}\|a\|^2\big\}, \qquad V_0 = r.$$

The supremum is a concave quadratic with unique maximizer $a^* = -\sigma_t \nabla_x V$. Since u^* attains the supremum, $u_t^*(x) = -\sigma_t \nabla_x V_t^{u^*}(x)$. The endpoint law follows from Theorem 3.1. □

B Derivations for RAM

This appendix collects the derivations behind Section 4.

B.1 Bayes bridge score derivation

We derive (15) in the general form covering any $0 \le s < t \le 1$. The forward SDE (2) is linear, so its transition kernel from s to t is Gaussian:

$$p_{t|s}(x_t \mid x_s) = \mathcal{N}\big(a_{t,s}x_s,\, \beta_{t,s}^2 I\big), \qquad a_{t,s} = \frac{1-t}{1-s}, \qquad \beta_{t,s}^2 = t^2 - \frac{(1-t)^2 s^2}{(1-s)^2}. \tag{30}$$

Bayes' rule gives

$$\nabla_{x_t}\log p_{s|t}(x_s \mid x_t) = \nabla_{x_t}\log p_{t|s}(x_t \mid x_s) - \nabla_{x_t}\log p_t(x_t).$$

The Gaussian term is $\nabla_{x_t}\log p_{t|s}(x_t \mid x_s) = -(x_t - a_{t,s}x_s)/\beta_{t,s}^2$, and the marginal score follows from (6) as $\nabla_x \log p_t(x) = -(x + (1-t)v_t(x))/t$. Combining the two,

$$\nabla_{x_t}\log p_{s|t}(x_s \mid x_t) = -\frac{x_t - a_{t,s}x_s}{\beta_{t,s}^2} + \frac{x_t + (1-t)v_t(x_t)}{t}. \tag{31}$$

This is the Bayes bridge score used by both the jump estimator (21) and the pathwise family of Section 5. Setting $s{=}0$ gives $a_{t,0} = 1-t$ and $\beta_{t,0} = t$. With $\epsilon := (x_t - (1-t)x_0)/t$, equation (31) simplifies to the prefactor form $\frac{1-t}{t}\big(v_t(x_t) - (\epsilon - x_0)\big)$ used in (15).

B.2 Control-space form of RAM and time-dependent weighting

The Bayes bridge score (15) has prefactor $(1-t)/t$, which combines with $\sigma_t^2 = 2t/(1-t)$ via the identity $\sigma_t(1-t)/t = 2/\sigma_t$.

Substituting the reward proxy (14) and the bridge score (15) into the self-consistency condition $u_t = -\sigma_t A_t^u$ gives the control-space target

$$u_t = \mathrm{sg}\left(\frac{2}{\sigma_t}\, r(X_0)\big((\epsilon - X_0) - v_t^\theta(X_t)\big)\right).$$

Regressing $u_t^\theta = (2/\sigma_t)(v_t^\theta - v_t^{\mathrm{ref}})$ against this target, the common $2/\sigma_t$ factor pulls out as $4/\sigma_t^2$ on the velocity-space squared residual:

$$\mathcal{L}_u(\theta) = \mathbb{E}\left[\frac{4}{\sigma_t^2}\left\|v_t^\theta(X_t) - \mathrm{sg}\big(v_t^{\mathrm{ref}}(X_t) + r(X_0)\big((\epsilon - X_0) - v_t^\theta(X_t)\big)\big)\right\|^2\right]. \tag{32}$$

The prefactor $4/\sigma_t^2 = 2(1-t)/t$ downweights timesteps near the source. Dropping it yields the uniformly weighted velocity loss used in the main text.

B.3 Path-integral characterization of the optimum

We prove Lemma 4.2. Throughout, $\epsilon \sim \mathcal{N}(0, I)$ is independent of X_0 and $X_t = (1-t)X_0 + t\epsilon$. The distribution of X_0 is stated alongside each expectation.

The ODE. We show that

$$\partial_\lambda v_t^\lambda(x) = \mathbb{E}\big[r(X_0)\big((\epsilon - X_0) - v_t^\lambda(x)\big) \mid X_t = x\big], \qquad X_0 \sim p_0^\lambda,$$

which is (19). Starting from $v_t^\lambda(x) = \mathbb{E}[\epsilon - X_0 \mid X_t = x]$ with $X_0 \sim p_0^\lambda$, we differentiate in λ. The score in λ of the conditional density is

$$\partial_\lambda \log p_0^\lambda(x_0 \mid X_t = x) = r(x_0) - \mathbb{E}[r(X_0) \mid X_t = x]_{X_0 \sim p_0^\lambda},$$

so for λ-independent f,

$$\partial_\lambda \mathbb{E}[f \mid X_t = x]_{X_0 \sim p_0^\lambda} = \mathbb{E}[r(X_0) f \mid X_t = x]_{X_0 \sim p_0^\lambda} - \mathbb{E}[f \mid X_t = x]_{X_0 \sim p_0^\lambda}\, \mathbb{E}[r(X_0) \mid X_t = x]_{X_0 \sim p_0^\lambda}.$$

Applying with $f = \epsilon - X_0$ and using $\mathbb{E}[\epsilon - X_0 \mid X_t = x] = v_t^\lambda(x)$ under p_0^λ yields the claim.

The integral form. Integrating the ODE from $\lambda = 0$ to $\lambda = 1$ and using $v_t^0 = v_t^{\mathrm{ref}}$, $v_t^1 = v_t^*$ gives

$$v_t^*(x) - v_t^{\mathrm{ref}}(x) = \int_0^1 \mathbb{E}\big[r(X_0)\big((\epsilon - X_0) - v_t^\lambda(x)\big) \mid X_t = x\big]\, \mathrm{d}\lambda, \qquad X_0 \sim p_0^\lambda,$$

which is (20).

B.4 Exact case: Gaussian reference under a linear reward

We show that for p_0^{ref} Gaussian and r linear, the velocity-field family $\{v_t^\lambda\}_{\lambda \in [0,1]}$ from Lemma 4.2 is affine in λ. The integrand of (20) is then constant in λ, the right-endpoint rule is exact, and the RAM target (14) matches the optimal adjoint at every x.

Let $p_0^{\mathrm{ref}} = \mathcal{N}(\mu, \Sigma)$ with positive-definite Σ and $r(x) = b^\top x + c$. Completing the square in the log-density of $p_0^\lambda \propto p_0^{\mathrm{ref}}\, e^{\lambda r}$ gives

$$p_0^\lambda = \mathcal{N}(\mu_\lambda, \Sigma), \qquad \mu_\lambda := \mu + \lambda \Sigma b, \tag{33}$$

a Gaussian whose mean shifts linearly in λ and whose covariance does not depend on λ. Under the analytic noising kernel (1), the joint (X_0, X_t) is Gaussian, and the conditional law of X_0 given $X_t = x$ is

$$X_0 \mid X_t = x \;\sim\; \mathcal{N}\big(m_t^\lambda(x),\, S_t\big), \tag{34}$$

with mean

$$m_t^\lambda(x) = \mu_\lambda + J_t(x - (1-t)\mu_\lambda), \qquad J_t \coloneqq (1-t)\,\Sigma\big((1-t)^2\Sigma + t^2 I\big)^{-1},$$

and covariance $S_t = t^2\big((1-t)^2\Sigma + t^2 I\big)^{-1}\Sigma$ that does not depend on λ. The conditional mean $m_t^\lambda(x)$ is affine in λ.

Using $\epsilon = (X_t - (1-t)X_0)/t$, the velocity field (3) for endpoint p_0^λ evaluates as

$$v_t^\lambda(x) = -\mathbb{E}[X_0 \mid X_t = x] + \frac{1}{t}\big(x - (1-t)\,\mathbb{E}[X_0 \mid X_t = x]\big) = \frac{x - m_t^\lambda(x)}{t},$$

which is affine in $m_t^\lambda(x)$ and therefore affine in λ. Substituting into (19) and applying Lemma 4.2 gives the claim.

A concrete consequence is that $v_t^\lambda(x) - v_t^{\mathrm{ref}}(x) = -\frac{1}{t}\big(I - (1-t)J_t\big)\,\lambda\Sigma b$ is independent of x. Through the control–velocity relation (16), the optimal control u_t^* is therefore independent of position, recovering the path-cost-vanishing instance referenced at (14).

C Derivations for Retaining the Path-Cost Correction

This appendix collects the derivations behind Section 5. We begin with the endpoint-sampling jump estimator, then turn to the pathwise family.

C.1 Random-jump identity and the jump estimator

The jump estimator (21) combines a single-point control-cost evaluation with a Bayes bridge score. Both pieces are justified by the following lemma.

Lemma C.1 (Unbiased single-point cost estimate). *Let $s \sim \mathcal{U}[0,t)$ be independent of the trajectory. Then*

$$\mathbb{E}\left[\frac{t}{2}\|u_s(X_s^u)\|^2 \;\middle|\; X_t^u = x\right] = \frac{1}{2}\mathbb{E}\left[\int_0^t \|u_\tau(X_\tau^u)\|^2\,\mathrm{d}\tau \;\middle|\; X_t^u = x\right], \tag{35}$$

and consequently

$$\mathbb{E}\left[r(X_0^u) - \frac{t}{2}\|u_s(X_s^u)\|^2 \;\middle|\; X_t^u = x\right] = V_t^u(x). \tag{36}$$

Proof. Conditioning on $X_t^u = x$ and applying Fubini's theorem,

$$\mathbb{E}\left[\frac{t}{2}\|u_s(X_s^u)\|^2 \;\middle|\; X_t^u = x\right] = \frac{1}{2}\int_0^t \mathbb{E}\big[\|u_\tau(X_\tau^u)\|^2 \mid X_t^u = x\big]\,\mathrm{d}\tau,$$

which is (35). Subtracting from the reward term in the value function (11) gives (36). □

We now derive the jump estimator. Under Theorem 3.1, at initialization ($v^\theta = v^{\mathrm{ref}}$) and at the optimum, the marginals of the controlled process coincide with those of the forward process from the appropriate endpoint distribution. The Markov property of the noising kernel then guarantees that the triple (X_0, X_s, X_t) obtained by sampling $X_s \sim p_{s|0}(\cdot \mid X_0)$ and then $X_t \sim p_{t|s}(\cdot \mid X_s)$ has the same joint law as (X_0^u, X_s^u, X_t^u) along the controlled process at those two points.

Conditional on (X_0, X_s, X_t), the state X_t enters the joint density only through the bridge $p_{s|t}^u$. Combining this with (36) and the reasoning behind (13), differentiating the value function in x produces

$$A_t^u(x) = \mathbb{E}\left[\Big(r(X_0^u) - \frac{t}{2}\|u_s(X_s^u)\|^2\Big)\nabla_{x_t}\log p_{s|t}^u(X_s^u \mid x_t)\big|_{x_t = x} \;\middle|\; X_t^u = x\right],$$

with the expectation over $s \sim \mathcal{U}[0,t)$ and the trajectory. Replacing the controlled-process sample by the two-step construction above and substituting the Bayes bridge score (31) yields (21).

C.2 Generalized adjoint identity

We prove Theorem 5.1.

Proof. Fix an admissible control u and $0 \le s < t \le 1$; assume the regularity required to interchange differentiation and conditional expectation. The value function satisfies the recursion

$$V_t^u(x) = \mathbb{E}\left[V_s^u(X_s^u) - \frac{1}{2}\int_s^t \|u_\tau(X_\tau^u)\|^2 \,\mathrm{d}\tau \;\middle|\; X_t^u = x \right]. \tag{37}$$

We differentiate both sides in x. For the prefix, the conditional law of X_s^u given $X_t^u = x$ has density $p^u_{s|t}(\cdot \mid x)$, so the score-function identity gives

$$\nabla_x \mathbb{E}[V_s^u(X_s^u) \mid X_t^u = x] = \mathbb{E}\left[V_s^u(X_s^u) \nabla_x \log p^u_{s|t}(X_s^u \mid x) \;\middle|\; X_t^u = x \right].$$

For the suffix, we differentiate pathwise through the rollout. Subtracting gives the right-hand side of (22), equal to $\nabla_x V_t^u(x)$ by the left-hand side. □

From a single SDE rollout we estimate the right-hand side as

$$\widehat{A}^u_{s,t} = \widehat{V}_s\, \widehat{S}_{s,t} - \tfrac{1}{2}\widehat{G}_s,$$

where $\widehat{V}_s = r(X_0^u) - \frac{1}{2}\int_0^s \|u_\tau(X_\tau^u)\|^2 \,\mathrm{d}\tau$ is the prefix value accumulated along the rollout, $\widehat{S}_{s,t}$ estimates the bridge score $\nabla_{x_t} \log p^u_{s|t}(X_s^u \mid x_t)$, and $\widehat{G}_s$ is the suffix path-cost gradient. The next subsections develop estimators for $\widehat{S}_{s,t}$ and the discrete-time machinery that produces $\widehat{G}_s$.

C.3 Malliavin bridge-score estimator

We derive the Malliavin estimator $\widehat{S}_{s,t}$.

Proposition C.2 (Malliavin score estimator). *Let $(X_\tau^u)_{\tau\in[s,t]}$ be a segment of the controlled process (8), and let $J^u_{\tau|s}$ denote the Jacobian flow*

$$\mathrm{d}J^u_{\tau|s} = \nabla_x\big(b_\tau^{\mathrm{ref}} + \sigma_\tau u_\tau\big)(X_\tau^u) J^u_{\tau|s}\,\mathrm{d}\tau, \qquad J^u_{s|s} = I. \tag{38}$$

Define

$$\widehat{S}_{s,t} := \left(\int_s^t \sigma_\tau^2 \,\mathrm{d}\tau\right)^{-1} \int_s^t (J^u_{\tau|s})^\top \sigma_\tau \,\mathrm{d}B_\tau. \tag{39}$$

Then $\mathbb{E}\big[\widehat{S}_{s,t} \;\big|\; X_s^u, X_t^u\big] = \nabla_{x_t} \log p^u_{s|t}(X_s^u \mid x_t)|_{x_t = X_t^u}$.

Proof. We apply Pidstrigach et al. (2025, Proposition 2.4): for a diffusion $\mathrm{d}X_\tau = f_\tau(X_\tau)\,\mathrm{d}\tau + \sigma_\tau\,\mathrm{d}B_\tau$ and any non-decreasing absolutely continuous α on $[s,t]$, the bridge score satisfies

$$\nabla_{x_t} \log p_{s|t}(x_s \mid x_t) = \frac{1}{\alpha_t - \alpha_s}\mathbb{E}\left[\int_s^t \alpha'_\tau \sigma_\tau^{-1} J_{\tau|s}^\top \,\mathrm{d}B_\tau \;\middle|\; X_s = x_s, X_t = x_t\right]. \tag{40}$$

Applied to the controlled process with σ_τ as in (8), the choice $\alpha'_\tau = \sigma_\tau^2$ makes $\alpha_t - \alpha_s = \int_s^t \sigma_\tau^2\,\mathrm{d}\tau$ and gives the estimator (39). □

The weighting $\alpha'_\tau = \sigma_\tau^2$ is convenient because the integrand becomes $(J^u_{\tau|s})^\top \sigma_\tau\,\mathrm{d}B_\tau$; in a discretized rollout this is exactly the realized solver noise.

In practice, rather than accumulating (39) forward in τ, we use the equivalent reverse-mode adjoint SDE indexed by a running time τ and a fixed endpoint t,

$$\mathrm{d}S_\tau^{(t)} = -\nabla_x\big(b_\tau^{\mathrm{ref}} + \sigma_\tau u_\tau\big)(X_\tau^u)^\top S_\tau^{(t)}\,\mathrm{d}\tau - \sigma_\tau\,\mathrm{d}B_\tau, \qquad S_t^{(t)} = 0, \tag{41}$$

with $\widehat{S}_{s,t} = (\int_s^t \sigma_\tau^2 \,\mathrm{d}\tau)^{-1} S_s^{(t)}$. Only vector–Jacobian products through the drift are required, never explicit Jacobians. The same dual-indexed form applies to the suffix running-cost gradient,

$$\mathrm{d}G_\tau^{(t)} = -\nabla_x\big(b_\tau^{\mathrm{ref}} + \sigma_\tau u_\tau\big)(X_\tau^u)^\top G_\tau^{(t)}\,\mathrm{d}\tau - \nabla_x\|u_\tau(X_\tau^u)\|^2\,\mathrm{d}\tau, \qquad G_t^{(t)} = 0, \tag{42}$$

with $\widehat{G}_s \coloneqq G_s^{(t)}$. The full-horizon specialization in Section C.6 produces all endpoint-indexed accumulators $S_0^{(t_i)}, G_0^{(t_i)}$ in a single backward sweep.

C.4 Local one-step Gaussian score

Let $s = t-\delta$ and write $b_t^u \coloneqq b_t^{\mathrm{ref}} + \sigma_t u_t$. A single Euler–Maruyama step of the controlled SDE (8) from t to $t-\delta$ gives

$$X_{t-\delta}^u = X_t^u - b_t^u(X_t^u)\delta + \Delta W_t, \qquad \Delta W_t \sim \mathcal{N}\big(0,\, \sigma_t^2\delta I\big).$$

Conditional on $X_t^u = x$, this defines the Gaussian one-step transition $p_{t-\delta|t}^{u,\delta}(\cdot \mid x) = \mathcal{N}(x - b_t^u(x)\delta,\, \sigma_t^2\delta I)$, with log-density $-\|X_{t-\delta}^u - \mu(x)\|^2/(2\sigma_t^2\delta)$ up to a constant, where $\mu(x) = x - b_t^u(x)\delta$. Differentiating in x,

$$\nabla_x \log p_{t-\delta|t}^{u,\delta}(X_{t-\delta}^u \mid x) = \frac{\big(I - \delta\nabla_x b_t^u(x)\big)^\top \Delta W_t}{\sigma_t^2\delta}, \tag{43}$$

which, evaluated at $x = X_t^u$, is the one-step bridge score for the local estimator. Its difference from the exact continuous-time bridge score is the $O(\delta)$ error already incurred by Euler–Maruyama.

C.5 Cancellation of $\nabla_x u_t$ terms in the local estimator

We show that the $\nabla_x u_t$ contributions from the score and the path-cost gradient cancel to first order in δ, yielding the local target (45). We work at the fixed point $u_t = -\sigma_t \nabla_x V_t^u$.

Let $s = t-\delta$. From (43) with $\Delta W_t = \sigma_t \Delta B_t$,

$$\nabla_{x_t} \log p_{s|t}^{u,\delta}(X_s^u \mid x_t)\big|_{x_t = X_t^u} = \frac{\big(I - \delta\nabla_x b_t^u(X_t^u)\big)^\top \Delta W_t}{\sigma_t^2\delta}. \tag{44}$$

Writing $\nabla_x b_t^u = \nabla_x b_t^{\mathrm{ref}} + \sigma_t \nabla_x u_t$, the part of the REINFORCE term involving $\nabla_x u_t$ is

$$-V_s^u(X_s^u)(\nabla_x u_t(X_t^u))^\top \Delta B_t.$$

The path-cost term on a single step is

$$-\frac{1}{2}\nabla_x \int_s^t \|u_\tau(X_\tau^u)\|^2\,\mathrm{d}\tau = -\delta(\nabla_x u_t(X_t^u))^\top u_t(X_t^u) + O(\delta^2).$$

To compare the two, expand the value function backward from t to s. At the fixed point, Itô's formula combined with $u_t = -\sigma_t \nabla_x V_t^u$ gives

$$V_s^u(X_s^u) = V_t^u(X_t^u) - u_t(X_t^u)^\top \Delta B_t + O(\delta).$$

Substituting this into the $\nabla_x u_t$ contribution of the score and taking conditional expectation given X_t^u, the first expansion term vanishes by martingale property while the second yields $\delta(\nabla_x u_t)^\top u_t$, using $\mathbb{E}[\Delta B_t \Delta B_t^\top \mid X_t^u] = \delta I$. This cancels the path-cost term to first order in δ.

What remains is the reference-drift piece,

$$\widehat{A}_t^{\mathrm{local}} \approx \mathbb{E}\left[\frac{V_{t-\delta}^u(X_{t-\delta}^u)}{\sigma_t^2\delta}\big(I - \delta\nabla_x b_t^{\mathrm{ref}}(X_t^u)\big)^\top \Delta W_t \;\middle|\; X_t^u\right].$$

Replacing the conditional value by the prefix estimate $\widehat{V}_{t-\delta}$ and simplifying yields the continuous-time local target

$$\widehat{A}_t^{\mathrm{local}} \approx \frac{\widehat{V}_{t-\delta}}{\int_{t-\delta}^t \sigma_\tau^2\,\mathrm{d}\tau}\big(I - \delta\nabla_x b_t^{\mathrm{ref}}(X_t^u)\big)^\top \Delta W_t, \qquad \Delta W_t = \sigma_t\sqrt{\delta}\,\xi_t, \quad \xi_t \sim \mathcal{N}\big(0,\, I\big). \tag{45}$$

Since $\nabla_x b_t^{\mathrm{ref}} = 2\nabla_x v_t^{\mathrm{ref}} - \kappa_t I$, the local target requires only a VJP through the frozen reference model.

C.6 Discrete-time full-horizon training procedure

We turn the full-horizon (s=0) instance of the estimator $\widehat{A}^{u}_{s,t}$ into a concrete training procedure. Each iteration rolls out the controlled SDE once and then computes targets for all interior grid points in a single discrete reverse-mode sweep, the discrete analogue of solving the continuous adjoint equations backward from every endpoint t_i to 0. Throughout this subsection, we write $X_i \coloneqq X_{t_i}$ and $b_i^{u^\theta} \coloneqq b_{t_i}^{u^\theta}$ for pointwise values at grid points. Expressions of the form $\nabla_x(f(X_i)^\top v)$ are read with the gradient taken in x and evaluated at $x{=}X_i$; this scalar-backprop form is what the implementation actually uses and never materializes the $d \times d$ Jacobian $\nabla_x f$.

SDE rollout. Fix a grid $0 = t_0 < t_1 < \cdots < t_K < 1$ with $\Delta t_i \coloneqq t_i - t_{i-1}$, and draw independent innovations $\varepsilon_i \sim \mathcal{N}(0,\, I)$ for $i = 1, \ldots, K$. Each innovation is scaled by the exact step standard deviation

$$\sigma_i^2 \coloneqq \int_{t_{i-1}}^{t_i} \sigma_\tau^2 \, \mathrm{d}\tau, \tag{46}$$

which evaluates to $\sigma_i^2 = 2\log\frac{1-t_{i-1}}{1-t_i} - 2\Delta t_i$. The integrated variance (46) is finite for $t_i < 1$ but diverges at the noise endpoint, since $\sigma_t^2 \sim 2/(1-t)$ as $t \uparrow 1$. We therefore take $t_K < 1$ (in practice $t_K = 1 - \delta$ for small δ).

Let m_i^θ denote the deterministic part of the reverse step $t_i \to t_{i-1}$ under the current model, and m_i^{ref} the corresponding reference step map. Starting from $X_K \sim \mathcal{N}(0,\, I)$, we roll out

$$X_{i-1} = m_i^\theta(X_i) + \sigma_i \varepsilon_i, \tag{47}$$

and store $\{X_i\}_{i=0}^K$ together with $\{\varepsilon_i\}_{i=1}^K$. Rather than regressing the continuous control u_i directly, the implementation regresses the exact normalized mean shift of this discrete step, $(m_i^\theta(x) - m_i^{\mathrm{ref}}(x))/\sigma_i$. For a plain Euler step this reduces to $m_i^\theta(x) = x - \Delta t_i b_i^{u^\theta}(x)$. In our experiments we instead use an exact integrator that admits a closed-form m_i^θ.

Adjoint sweep. Let S_i and G_i denote the discrete adjoint accumulators associated with the interval $[0, t_i]$. Although the loop runs in increasing i, it still represents backward propagation through the reverse solver: extending the horizon from t_{i-1} to t_i transports the previously accumulated adjoint through the transpose Jacobian of the step map and adds the new contribution from step i. The recursion is

$$S_i = \nabla_x\big(m_i^\theta(X_i)^\top S_{i-1}\big) + \sigma_i \varepsilon_i, \tag{48}$$

$$G_i = \nabla_x\big(m_i^\theta(X_i)^\top G_{i-1}\big) + \nabla_x \left\| \frac{m_i^\theta(X_i) - m_i^{\mathrm{ref}}(X_i)}{\sigma_i} \right\|^2, \tag{49}$$

for $i = 1, \ldots, K{-}1$, and the bridge-score and full-horizon adjoint targets at grid point t_i are

$$\widehat{S}_i = \Big(\textstyle\sum_{k=1}^i \sigma_k^2\Big)^{-1} S_i, \qquad \widehat{A}_i = r(X_0)\, \widehat{S}_i - \tfrac{1}{2} G_i. \tag{50}$$

Using the Bayes bridge score (31) instead of Malliavin replaces $\widehat{S}_i$ by its plug-in form on (X_0, X_i) and leaves the path-cost sweep for G_i unchanged.

Local variant. The local estimator (45) reuses the same stored rollout with no multi-step sweep. At each interior index i,

$$\widehat{A}_i^{\mathrm{local}} = \frac{\widehat{V}_{i-1}}{\sigma_i} \nabla_x\big(m_i^{\mathrm{ref}}(X_i)^\top \varepsilon_i\big), \qquad \widehat{V}_{i-1} = r(X_0) - \tfrac{1}{2} \sum_{k=1}^{i-1} \left\| \frac{m_k^\theta(X_k) - m_k^{\mathrm{ref}}(X_k)}{\sigma_k} \right\|^2,$$

requiring a single backward pass through the frozen reference step map and no differentiation through the current policy.

Algorithm 2 gives the full-horizon Malliavin procedure. Each iteration accumulates the exact discrete analogue of the velocity-space regression loss over the interior grid points, and the Bayes and local variants substitute their respective targets into the same rollout.

Algorithm 2 Full-horizon Malliavin RAM

Inputs: parameters θ (initialized from v^{ref}), reward r, grid $0{=}t_0 < \cdots < t_K{<}1$

while not converged **do**

▷ **SDE rollout**

  sample $X_K \sim \mathcal{N}(0,\, I)$

  for $i = K, \ldots, 1$ **do**

    sample $\varepsilon_i \sim \mathcal{N}(0,\, I)$

    $X_{i-1} \leftarrow m_i^{\theta}(X_i) + \sigma_i \varepsilon_i$

  end for

  $r_0 \leftarrow r(X_0)$

▷ **Adjoint sweep and loss accumulation**

  $S \leftarrow 0,\ G \leftarrow 0,\ L \leftarrow 0$

  for $i = 1, \ldots, K{-}1$ **do**

    $S \leftarrow \nabla_x\big(m_i^{\theta}(X_i)^{\top} S\big) + \sigma_i \varepsilon_i$

    $G \leftarrow \nabla_x\big(m_i^{\theta}(X_i)^{\top} G\big) + \nabla_x \left\| \frac{m_i^{\theta}(X_i) - m_i^{\mathrm{ref}}(X_i)}{\sigma_i} \right\|^2$

    $\widehat{S} \leftarrow \big(\sum_{k=1}^{i} \sigma_k^2\big)^{-1} S$

    $\widehat{A} \leftarrow r_0\, \widehat{S} - \frac{1}{2} G$

    $L \leftarrow L + \left\| \frac{m_i^{\theta}(X_i) - m_i^{\mathrm{ref}}(X_i)}{\sigma_i} - \mathrm{sg}\big(\sigma_i\, \widehat{A}\big) \right\|^2$

  end for

  update θ with $\nabla_\theta L/(K{-}1)$

end while

D Experimental Details

We adapt the general setup of prior work (Liu et al., 2025a; Zheng et al., 2026; Xue et al., 2025). We post-train Stable Diffusion 3.5 Medium (SD3.5M) (Esser et al., 2024) with LoRA (rank $r = 32$, scaling $\alpha = 64$). We train all models on a cluster of 4 NVIDIA H100 GPUs with 96 GB of memory each.

Training steps. Each training step, we draw 48 prompts, generate 24 samples per prompt, evaluate the reward on every prompt-image pair, construct $K = 8$ noisy training samples per image for which we compute the RAM loss, and perform a single optimizer step. This means that the effective batch size (across all GPUs) is $48 \times 24 \times 8 = 9216$ per parameter update. For a fair comparison, we use 48 prompts per step for all baselines as well (AWM's reference implementation uses 72); otherwise we retain the hyperparameters and implementation of each original work.

Per-step compute cost. Table 3 reports the average per-step compute cost of each method, measured in GPU-hours (wall-clock time multiplied by the four H100 GPUs we train on). Per-step costs are comparable across methods, so RAM's step-count efficiency translates directly into wall-clock training time.

Table 3 *Average per-step training cost, in GPU-hours (training wall-clock time multiplied by four H100 GPUs).*

Method	GenEval	PickScore	OCR
Flow-GRPO	0.701	0.431	0.498
AWM	0.653	0.329	0.370
DiffusionNFT	0.566	0.254	0.304
RAM	0.666	0.399	0.426

Sampling and guidance. We use the default Euler sampler throughout, with 20 steps during training and 40 steps at evaluation. Training uses classifier-free guidance with scale 2.0. At evaluation we use the default SD3.5M scale of 4.5 for GenEval and OCR, and a lower scale of 2.0 for PickScore. The lower scale for PickScore reflects its longer training run, during which guidance gets distilled into the model.

Optimizer. We use AdamW with learning rate 3e−4, weight decay 0.01, and no learning-rate warmup. We lower Adam's second-moment decay β_2 from its default of 0.999 to 0.95. This is a common choice for short training runs, as the smaller β_2 lets the optimizer adapt more quickly to the gradient scale.

Exponential moving average. On-policy samples drawn during training come from an exponential moving average (EMA) of the model parameters with decay 0.9 and warmup rate 0.01. Prior works use task-dependent EMA configurations; we use a single setting that works uniformly well across all three tasks. Final evaluation uses an EMA with decay 0.9 and no warmup.

Reward coefficient. After group-relative normalization, we multiply the rewards by a fixed task-dependent coefficient: 100 for GenEval and OCR, and 1000 for PickScore. Because reward scaling controls the regularization strength in our convention, this coefficient corresponds to $1/$`kl_weight` in methods that instead report an explicit KL-penalty weight. The values were tuned to maximize the training reward while still scoring well on the image-quality metrics.

Timestep sampling. During training we sample the timestep t from the linear density $p(t) = 2t$ on $[0, 1]$, biasing toward values near the noise endpoint t=1 where generation begins. This notably enhances training stability over uniform sampling.