Efficient Adjoint Matching for Fine-tuning Diffusion Models

Jeongwoo Shin[*]
Seoul National University
swswss@snu.ac.kr

Dongsoo Shin[*]
Seoul National University
dongsoo@snu.ac.kr

Yuchen Zhu
Georgia Institute of Technology
yzhu738@gatech.edu

Wei Guo
Georgia Institute of Technology
wei.guo@gatech.edu

Yongxin Chen
Georgia Institute of Technology
yongchen@gatech.edu

Joonseok Lee[†]
Seoul National University,
Hanwha Aerospace
joonseok@snu.ac.kr

Jaewoong Choi[†]
Sungkyunkwan University
jaewoongchoi@skku.edu

Jaemoo Choi[†]
Georgia Institute of Technology
jaemoo.choi@gatech.edu

github.com/swswss/EAM

Abstract

Reward fine-tuning has become a common approach for aligning pretrained diffusion and flow models with human preferences in text-to-image generation. Among reward-gradient-based methods, Adjoint Matching (AM) provides a principled formulation by casting reward fine-tuning as a stochastic optimal control (SOC) problem. However, AM inevitably requires a substantial computational cost: it requires (i) stochastic simulation of full generative trajectories under memoryless dynamics, resulting in a large number of function evaluations, and (ii) backward ODE simulation of the adjoint state along each sampled trajectory. In this work, we observe that both bottlenecks are closely tied to the *non-trivial base drift* inherited from the pretrained model. Motivated by this observation, we propose **Efficient Adjoint Matching (EAM)**, which substantially improves training efficiency by reformulating the SOC problem with a *linear base drift* and a correspondingly modified *terminal cost*. This reformulation removes both sources of inefficiency; it enables training-time sampling with a few-step deterministic ODE solver and yields a closed-form adjoint solution that eliminates backward adjoint simulation. On standard text-to-image reward fine-tuning benchmarks, EAM converges up to 4× faster than AM and matches or surpasses it across various metrics including PickScore, ImageReward, HPSv2.1, CLIPScore and Aesthetics.

1 Introduction

Diffusion [18, 35] and flow [23, 26, 1] models have become the standard backbone for large-scale text-to-image (T2I) generation [30, 31, 9]. While their pretraining objective produces photorealistic samples, it is often poorly aligned with the qualities that actually matter in deployment such as aesthetic quality, prompt fidelity, and adherence to human preferences [42, 39]. To close this gap, reward-based fine-tuning has emerged as a standard recipe for aligning text-to-image models, where

[*]Equal contribution.
[†]Corresponding author.

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

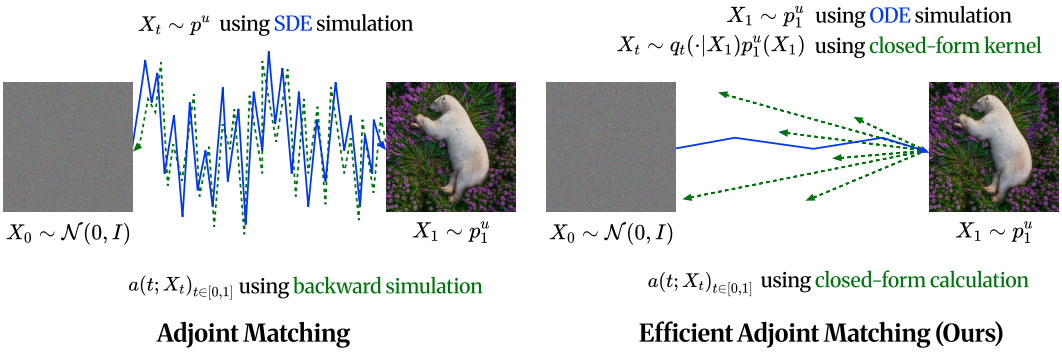


Figure 1: **Comparison of Efficient Adjoint Matching (EAM) with Adjoint Matching (AM).** (*Left*) AM relies on a stochastic SDE solver to construct each training trajectory and a sequential backward simulation to obtain the adjoint state along that trajectory. (*Right*) EAM eliminates both: intermediate states X_t are obtained by first simulating the endpoint X_1 with a few-step ODE and then sampling X_t from the original noising kernel $q_t(\cdot|X_1)$, while the adjoint state is given by a single closed-form evaluation, removing the backward simulation entirely.

pretrained diffusion models are post-trained to maximize an external reward function that reflects these preferences [25, 7, 49, 5, 6, 42, 29].

Among existing reward fine-tuning methods, reward-gradient-based methods provide direct supervision by leveraging the gradient of the reward function with respect to its input [6, 42, 29, 7]. This gradient is readily available when the reward is parameterized by a differentiable network, as is typical for learned preference models such as CLIPScore, HPS, Aesthetic score, ImageReward, and PickScore [17, 39, 32, 42, 22]. While many reward fine-tuning methods use only scalar reward values [25, 49, 5], reward-gradient-based methods additionally use the local gradient direction of the reward model. This provides each generated sample with direct supervision toward higher reward, offering an effective and sample-efficient route when reliable reward gradients are available.

Adjoint Matching (AM) [7] introduces a theoretically principled framework for reward-gradient-based fine-tuning. Specifically, AM casts reward fine-tuning as a stochastic optimal control (SOC) problem and learns an additional control that transforms the pretrained model toward the reward-tilted target distribution $p_1^\star$, defined as

$$p_1^\star(x) \propto e^{\beta r(x)} p_{\text{data}}(x), \tag{1}$$

where $r(x)$ is a given reward function, $\beta > 0$ is the constant that controls the strength of reward guidance, and p_{data} denotes the data distribution. However, AM requires heavy computational cost for optimization (Sec. 3.1). First, AM relies on **forward SDE simulation** of full generative trajectories under memoryless dynamics. This SDE simulation requires a large number of function evaluations (NFEs), substantially increasing sampling time during training. Second, AM requires **backward ODE simulation** for adjoint state along sampled trajectory to propagate reward-gradient signal (Sec. 2.3). Together, these two simulation procedures dominate the training cost.

Our key observation is that these bottlenecks arise from the choice of the drift term of the base dynamics (Sec. 3.2). Building on this observation, we redesign the base dynamics using a *simple linear drift* that satisfies the desired conditions. We accordingly modify the *terminal cost* in the SOC problem so that the optimal terminal distribution matches with $p_1^\star$ in Eq. (1). This reformulation leads to our algorithm, **Efficient Adjoint Matching (EAM)**, which removes both costly simulations in AM in the following way. First, forward trajectory simulation becomes solver-agnostic: the endpoint image X_1 can be generated using any efficient few-step deterministic ODE solver [46, 47], and intermediate states X_t can be sampled from the original noising kernel $q_t(\cdot \mid X_1)$. Second, the backward ODE simulation for computing the adjoint state could be replaced by closed-form solution, eliminating backward ODE simulation entirely. Despite this change of base dynamics, we prove that EAM shares the same optimal path measure as AM, so the efficiency gain comes without altering the optimum. Empirically, EAM reduces the per-iteration training cost by up to 4× compared to AM, while matching or surpassing its performance across various human preference metrics. Our contributions can be summarized as follows:

- We propose **Efficient Adjoint Matching (EAM)**, an efficient reward-gradient-based fine-tuning algorithm derived by redesigning the *base drift* and a *terminal cost* of SOC problem.
- We **characterize the linear base drifts** satisfying our design requirements, enabling efficient ODE-based trajectory construction and a closed-form adjoint state, and prove that the resulting **optimal path measure coincides with that of AM**.
- We show that EAM achieves **comparable or better performance** on standard text-to-image reward fine-tuning benchmarks while converging up to **4× faster** than AM.

2 Preliminaries

2.1 Diffusion and Flow Models

Diffusion [18, 35] and flow matching [23, 1, 26] models share a common formulation through a pretrained velocity field $v^{\mathrm{pt}} : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$, learned by velocity matching along a conditional probability path. Throughout this paper, we focus on the linear path [23, 26]. Equivalently, given an endpoint X_1, intermediate states are sampled from the following noising kernel:

$$q_t(X_t \mid X_1) := \mathcal{N}\big(tX_1, (1-t)^2 I\big), \qquad \textit{i.e.,} \quad X_t = tX_1 + (1-t)\epsilon, \;\; \epsilon \sim \mathcal{N}(0, I), \tag{2}$$

where $X_1 \sim p_{\mathrm{data}}$. We denote the endpoint distributions by $p_0 = \mathcal{N}(0, I)$ and $p_1 = p_{\mathrm{data}}$. The pretrained velocity field v^{pt} induces the following forward generative dynamic:

$$\mathrm{d}X_t = \left(-\frac{1}{t}X_t + 2v^{\mathrm{pt}}(X_t, t)\right)\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \qquad X_0 \sim p_0, \qquad \sigma(t) = \sqrt{\frac{2(1-t)}{t}}. \tag{3}$$

We write p^{pt} for the path measure induced by Eq. (3), and throughout the paper we fix the diffusion coefficient $\sigma(t)$ as in Eq. (3). Since the pretrained model is trained to approximate the data distribution, we use the standard idealization $p_1^{\mathrm{pt}} = p_{\mathrm{data}}$.

2.2 Stochastic Optimal Control for Fine-tuning Diffusion Models

Problem Setting and Notations. We consider the following SOC problem:

$$\min_u \mathbb{E}_{p^u}\left[\int_0^1 \frac{1}{2}\left\|u(X_t,t)\right\|^2 \mathrm{d}t + g(X_1)\right], \tag{4}$$

$$\text{s.t.} \quad \mathrm{d}X_t = \big(b(X_t,t) + \sigma(t)\,u(X_t,t)\big)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \qquad X_0 \sim p_0, \tag{5}$$

where $X_t \in \mathbb{R}^d$ is the state of the controlled dynamic (5), $u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is the control vector field, $b : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is the base drift, and $g : \mathbb{R}^d \to \mathbb{R}$ is the terminal cost. We denote by p^u the path measure induced by the controlled dynamic (5), and by p^{base} the path measure induced by the uncontrolled base dynamic, *i.e.*, Eq. (5) with $u \equiv 0$. With a properly designed terminal cost g, the controlled dynamic (5) under the optimal control $u^\star$ reaches the target distribution $p_1^\star$ at $t = 1$.

Standard setup for SOC-based reward fine-tuning. Suppose the base drift $b(x,t)$ and the terminal cost $g(x)$ are defined as follows:

$$b(x,t) = -\frac{1}{t}x + 2v^{\mathrm{pt}}(x,t), \qquad g(x) = -\beta r(x). \tag{6}$$

Then, the optimal solution $p^{u^\star}$ for the SOC problem yields $p_1^{u^\star}(x) = p_1^\star(x) \propto e^{\beta r(x)} p_{\mathrm{data}}(x)$.

Furthermore, to yield an unbiased estimator of $u^\star$, the base dynamic must be memoryless [7]:

$$p_{0,1}^{\mathrm{base}}(X_0, X_1) = p_0^{\mathrm{base}}(X_0)\, p_1^{\mathrm{base}}(X_1). \tag{7}$$

The standard choice (6) satisfies this condition.

2.3 Adjoint Matching

Adjoint Matching (AM) [7] provides an efficient way to solve the SOC problem (4) under Eq. (6) by learning the control u through regression onto the lean adjoint state $a : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$:

$$\mathcal{L}_{\mathrm{AM}} := \frac{1}{2}\int_0^1 \|u(X_t,t) + \sigma(t)\, a(t;X_t)\|^2 \,\mathrm{d}t, \qquad X_t \sim p^{\bar{u}}, \quad \bar{u} = \mathrm{stopgrad}(u), \tag{8}$$

$$\text{where} \qquad \frac{\mathrm{d}}{\mathrm{d}t} a(t;X_t) = -\nabla_x b(X_t,t)^\top a(t;X_t), \qquad a(1;X_1) = \nabla g(X_1). \tag{9}$$

Each training iteration of AM proceeds in four steps: (i) simulate a trajectory $\{X_t\}_{t\in[0,1]} \sim p^{\bar{u}}$ of the controlled dynamic (5) with a stochastic solver, as required by the memoryless condition (7); (ii) compute the gradient of the terminal cost $\nabla g(X_1) = -\beta \nabla r(X_1)$; (iii) simulate the adjoint ODE backward along the stored trajectory to obtain the adjoint states (9); (iv) and regress the control to the matching target (8).

3 Efficient Adjoint Matching

3.1 Inefficiency of Adjoint Matching

Although Adjoint Matching (AM) [7] achieves strong empirical alignment performance among reward gradient-based fine-tuning methods with distributional guarantees, its per-iteration cost has not been carefully examined. We identify two computational bottlenecks:

- **Forward SDE simulation.** The regression in Eq. (8) requires sampling $X_t \sim p^{\bar{u}}$ under memoryless condition (7). It typically requires substantially more sampling steps than its deterministic counterpart and additionally requires storing the full trajectory in memory for the subsequent adjoint pass.
- **Backward adjoint simulation with Jacobian–Vector Product (JVP).** Computing the lean adjoint state (9) requires a backward simulation along the same trajectory, with each step evaluating the JVP $\nabla_x b(X_t,t)^\top a(t;X_t)$. Since the base drift b contains the pretrained velocity v^{pt}, this JVP must be propagated through the network at every integration step.

Our core observation is that both inefficiencies stem from the **base drift** b in Eq. (6). In Sec. 3.2, we first characterize the properties required for an efficient base dynamic, namely **simulation-free adjoint computation** and **efficient trajectory construction**. Then, in Sec. 3.3, we instantiate these properties by **redesigning the base drift** and **rederiving the terminal cost** g, so that the optimal controlled terminal marginal remains the reward-tilted distribution in Eq. (1).

3.2 Characterizing an Efficient Base Dynamic

We identify two structural conditions for an efficient base dynamic: linearity of the base drift, which enables closed-form adjoint computation, and memorylessness, which enables efficient construction of intermediate states.

Condition 1. The base drift b is **linear** in x.

Condition 2. The base dynamic is **memoryless** (7).

We now show that these two properties jointly eliminate both bottlenecks identified in Sec. 3.1.

Condition 1: Linear drift enables a closed-form adjoint. Suppose the base drift is linear in x, i.e., $b(x,t) = D(t)x$ for some scalar function $D : [0,1] \to \mathbb{R}$. Then the lean adjoint state defined by Eq. (9) admits the closed-form solution

$$a(t;X_t) = \exp\left(\int_t^1 D(\tau)\,\mathrm{d}\tau\right) a(1;X_1). \tag{10}$$

Since $\nabla_x b(x,t) = D(t)I$, the JVP in Eq. (9) reduces to a scalar multiplication and no longer requires differentiating through the pretrained network v^{pt}. Thus, Eq. (10) eliminates the backward adjoint ODE simulation: the adjoint state at any time t is the endpoint gradient $a(1;X_1) = \nabla g(X_1)$ scaled by a coefficient that depends only on t. Accordingly, the AM loss in Eq. (8) reduces to

$$\mathbb{E}_{(X_t,X_1)\sim p^{\bar{u}}}\left[\frac{1}{2}\int_0^1 \left\| u(X_t,t) + \sigma(t)\exp\left(\int_t^1 D(\tau)\,\mathrm{d}\tau\right)\nabla g(X_1)\right\|^2 \mathrm{d}t\right]. \tag{11}$$

Table 1: Comparison between AM and our EAM.

Method	**Adjoint Matching** [7]	**Efficient Adjoint Matching**
Control problem	$\min_u \mathbb{E}_{p^u}\left[\int_0^1 \frac{1}{2}\|u(X_t^u,t)\|^2\,\mathrm{d}t + g(X_1^u)\right]$ s.t. $\mathrm{d}X_t^u = \big(b(X_t^u,t) + \sigma(t)\,u(X_t^u,t)\big)\mathrm{d}t + \sigma(t)\,\mathrm{d}W_t, \quad X_0^u \sim p_0$	
Base drift $b(x,t)$	$-\frac{1}{t}\,x + 2\,v^{\mathrm{pt}}(x,t)$	$\dfrac{2Ct^2-1}{t\,(2Ct^2-2t+1)}\,x$
Grad. of terminal cost $\nabla g(x)$	$-\beta\nabla r(x)$	$-\frac{x}{2C-1} - \nabla \log p_1^{\mathrm{pt}}(x) - \beta\nabla r(x)$
Loss objective	Eq. (8)	Eq. (13)
Adjoint simulation-free	✗	✓
ODE solver for $\{X_t\}_{t\in[0,1]}$	✗	✓
No storage of $\{X_t\}_{t\in[0,1]}$	✗	✓

Condition 2: Memorylessness enables direct sampling of X_t from X_1. Even with the closed-form adjoint in Eq. (11), the loss still requires samples of the joint pair (X_t, X_1). A direct implementation obtains this pair by simulating the full stochastic trajectory. When the base dynamics are memoryless (7), this full rollout can be bypassed [12]: under SOC optimality, the intermediate state can be sampled directly from the endpoint X_1,

$$\int p^{\mathrm{base}}_{t|0,1}(X_t \mid X_0, X_1)\, p^{u^\star}(X_0, X_1)\mathrm{d}X_0 = p^{\mathrm{base}}_{t|1}(X_t \mid X_1)\, p^{u^\star}(X_1). \tag{12}$$

Thus, instead of simulating the full path, we first sample $X_1 \sim p_1^{\bar{u}}$ using an efficient ODE solver, and then sample X_t from the conditional base kernel $p^{\mathrm{base}}_{t|1}(X_t \mid X_1)$. Moreover, since the base drift is linear, this conditional distribution is available in closed form, so each intermediate state can be obtained by adding the appropriate amount of forward noise to the endpoint X_1 in a single step. The control matching loss therefore reduces further to

$$\mathcal{L}_{\mathrm{EAM}} = \mathbb{E}_t \mathbb{E}_{X_1\sim p_1^{\bar{u}},\, X_t \sim p^{\mathrm{base}}_{t|1}(\cdot|X_1)}\left[\frac{1}{2}\left\| u(X_t,t) + \sigma(t) e^{\left(\int_t^1 D(\tau)\,\mathrm{d}\tau\right)} \nabla g(X_1)\right\|^2\right]. \tag{13}$$

Since Eq. (12) holds only at optimality, this sampling is approximate during training, while the regression target remains exact for any control. When $\bar{u} = u^\star$, the sampling becomes exact and $\mathcal{L}_{\mathrm{EAM}}$ reduces to the AM regression (11), whose minimizer is $u^\star$ [7]. Hence $u^\star$ is a fixed point of EAM, although convergence is not guaranteed in general. This type of fixed-point scheme is widely adopted in recent SOC-based methods [15, 24, 14].

Summary. Condition 1 requires the base drift to be linear, which yields a closed-form adjoint and eliminates backward adjoint simulation. **Condition 2** requires the base dynamic to be memoryless, which enables endpoint-conditioned noising in place of full SDE simulation. Together, these conditions remove the two computational bottlenecks identified in Sec. 3.1. We next instantiate them by redesigning the base drift and rederiving the terminal cost to preserve the reward-tilted target (1).

3.3 Redesigning the Base Drift and Terminal Cost for Efficient Adjoint Matching

We now instantiate the conditions identified in Sec. 3.2. Since fine-tuning starts from the pretrained generative dynamic, diffusion coefficient $\sigma(t)$ is fixed as in Eq. (3). Under this restriction, **Conditions 1 & 2** alone do not uniquely determine the base dynamic. We further narrow down the design space by requiring the base noising kernel $p^{\mathrm{base}}_{t|1}(\cdot|X_1)$ to match the original noising kernel q_t (2):

Condition 3. The base noising kernel matches the original noising kernel in Eq. (2):

$$p^{\mathrm{base}}_{t|1}(\cdot \mid X_1) = q_t(\cdot \mid X_1) = \mathcal{N}\big(tX_1, (1-t)^2 I\big). \tag{14}$$

Algorithm 1 Efficient Adjoint Matching

Require: Pretrained velocity model v^{pt}, reward $r(x)$, LoRA parameters θ

Initialize $v^{\text{ft}} \leftarrow \text{LoRA}_\theta(v^{\text{pt}})$

repeat

Sample X_1 via ODE simulation:

$$\mathrm{d}X_t = \bar{v}^{\text{ft}}(X_t, t)\,\mathrm{d}t, \quad X_0 \sim p_0, \quad \bar{v}^{\text{ft}} = \text{stopgrad}(v^{\text{ft}}) \tag{15}$$

Construct intermediate state using $q_t(\cdot \mid X_1)$ (2):

$$X_t = tX_1 + (1-t)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad t \sim \mathcal{U}[0,1] \tag{16}$$

Compute $\nabla g(X_1)$ (18) using Eq. (20)

Minimize the loss objective $\mathcal{L}_{\text{EAM}}$ in Eq. (13)

Update θ by gradient descent on $\nabla_\theta \mathcal{L}_{\text{EAM}}(\theta)$

until convergence

return v^{ft}

The following proposition characterizes the resulting admissible family of base drifts.

Proposition 3.1 (Family of admissible linear base drifts)**.** *Assume that the diffusion coefficient $\sigma(t)$ is fixed as in Eq. (3). Then the base dynamic satisfies **Conditions 1–3** if and only if its drift is*

$$b(x,t) = D(t)x, \qquad D(t) = \frac{2Ct^2 - 1}{t\,(2Ct^2 - 2t + 1)}, \qquad C > \frac{1}{2}. \tag{17}$$

Since the base dynamic is now redesigned, the standard terminal cost (6) no longer yields the desired reward-tilted distribution in Eq. (1). We therefore redesign the terminal cost $g(x)$ to account for our new base dynamic induced by Eq. (17).

Proposition 3.2 (Terminal cost correction)**.** *Let the base dynamic in Eq. (5) use the redesigned drift in Eq. (17), whose terminal marginal is $p_1^{\text{base}} = \mathcal{N}(0, (2C-1)I)$. If*

$$\nabla g(x) = \nabla \log p_1^{\text{base}}(x) - \nabla \log p_1^{\text{pt}}(x) - \beta \nabla r(x), \tag{18}$$

then the optimal controlled terminal distribution satisfies $p_1^{u^\star} = p_1^\star$, with $p_1^\star$ defined in Eq. (1).

Proposition 3.2 guarantees that EAM attains the reward-tilted terminal distribution. We further show that the entire optimal path measure coincides with that of AM.

Theorem 3.3 (Equivalence of optimal path measures)**.** *Let $u^\star_{\text{AM}}$ and $u^\star_{\text{EAM}}$ be the optimal controls of the SOC problem (4)–(5) under the AM setting (6) and under our setting (17)–(18), respectively. Under standard well-posedness assumptions, their induced path measures coincide:*

$$p^{u^\star_{\text{AM}}} = p^{u^\star_{\text{EAM}}}. \tag{19}$$

In particular, both optimal processes share the drift $b^\star(x,t) = \frac{x}{t} + \sigma^2(t)\nabla \log p_t^\star(x)$, where $p_t^\star(x) = \int q_t(x \mid x_1)\, p_1^\star(x_1)\,\mathrm{d}x_1$.

Note that the optimal controls themselves differ ($u^\star_{\text{AM}} \neq u^\star_{\text{EAM}}$), since they are measured relative to different base drifts. Consequently, the control costs and the optimization landscapes differ, which may lead to different training behavior under finite model capacity. Their optimal controlled processes, however, are identical. Hence EAM is not an approximation of AM: it reaches exactly the same optimum through a different reference process. The proof is provided in App. A.5.

Remark 3.4 (Instantiating the EAM objective)**.** *Our EAM can be instantiated by plugging our redesigned base drift (17) and corrected terminal cost (18) into our loss objective $\mathcal{L}_{EAM}$ (13).*

The remaining practical questions are how to approximate $\nabla \log p_1^{\text{pt}}$ in terminal cost (18) and how to parameterize control $u(x,t)$ in our loss objective (13); we address both in the next section.

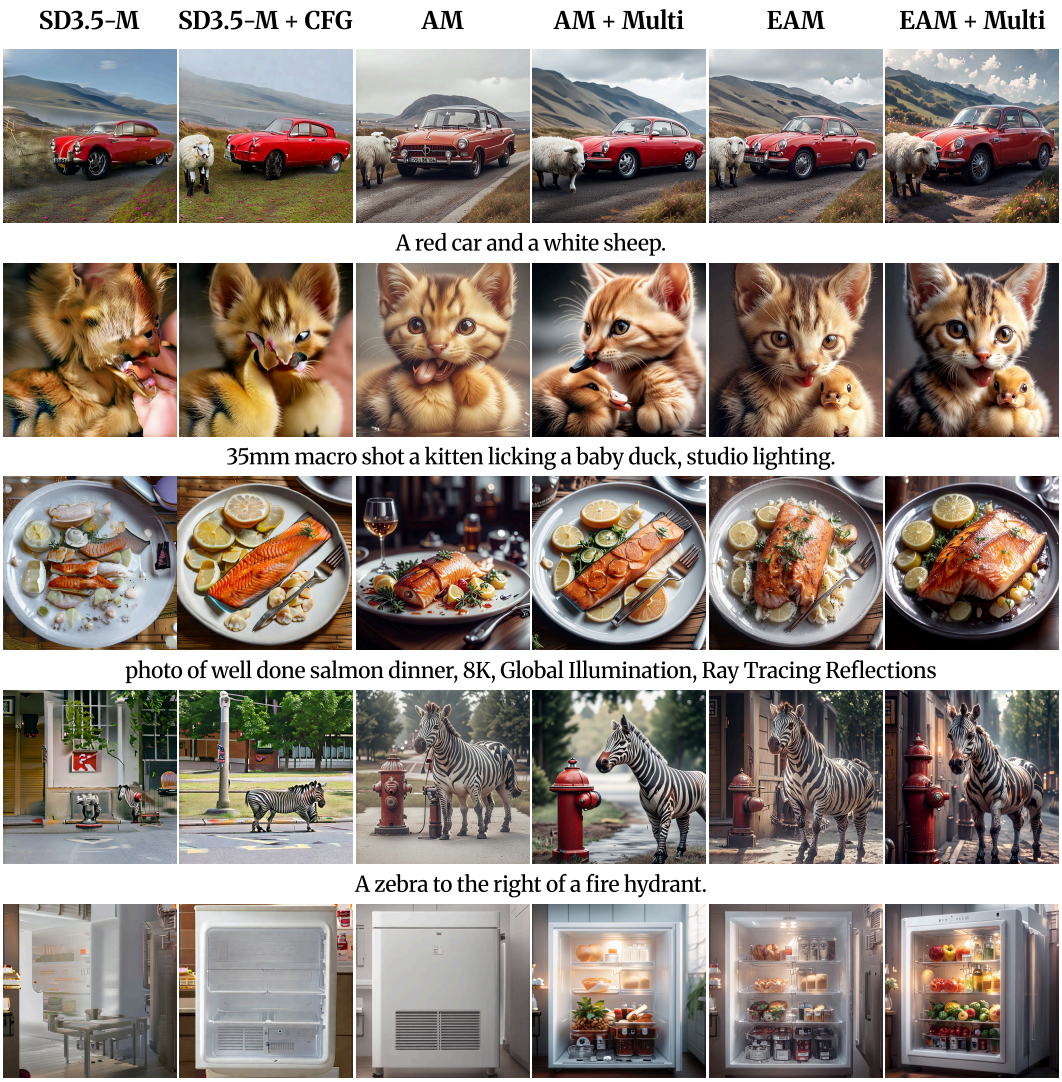


Figure 2: **Qualitative comparison**. See App. D for more examples.

3.4 Training Algorithm

Score estimation. The terminal cost correction in Eq. (18) requires the pretrained terminal score $\nabla \log p_1^{\mathrm{pt}}(x)$. We estimate it using Tweedie's formula [8] along the pretrained linear path (3):

$$\nabla \log p_{\tilde{t}}^{\mathrm{pt}}(x) = \frac{\tilde{t}\, v^{\mathrm{pt}}(x, \tilde{t}) - x}{1 - \tilde{t}}, \qquad \tilde{t} \approx 1. \tag{20}$$

Ideally, this estimate uses a perturbation level $\tilde{t}$ close to 1 to approximate $\nabla \log p_1^{\mathrm{pt}}(x)$. In practice, we reuse the intermediate state (X_t, t) constructed for the matching loss (via noising kernel q_t (2)) and plug it into Eq. (20), avoiding an additional noising step:

$$\nabla g(X_1) \approx -\frac{1}{2C-1} X_1 + \frac{1}{1-t} X_t - \frac{t}{1-t} v^{\mathrm{pt}}(X_t, t) - \beta \nabla r(X_1). \tag{21}$$

While this introduces bias for t far from 1, it avoids the unreliable velocity estimates near $t = 1$ and yields a lower-variance target. Ablation results comparing this estimator (20) with the exact Tweedie estimator and an ELBO-based estimator [5] are provided in App. C.2.

Table 2: **Quantitative evaluation across metrics.** EAM achieves comparable or better alignment quality than AM across most metrics, both when optimizing PickScore alone and when optimizing the combined reward of PickScore, HPSv2.1, and Aesthetics.

Method	PickScore	CLIPScore	ImageReward	HPSv2.1	Aesthetics
SD3.5-M [9]	20.34	0.226	-0.58	0.214	5.29
SD3.5-M + CFG	21.82	**0.275**	0.65	0.265	5.41
PickScore					
AM [7]	23.00	0.259	<u>0.97</u>	0.292	6.25
EAM (ours)	**23.16**	0.252	0.95	0.297	<u>6.27</u>
PickScore + HPSv2.1 + Aesthetics					
AM	<u>23.06</u>	0.253	0.83	<u>0.303</u>	6.21
EAM (ours)	22.93	<u>0.261</u>	**0.98**	**0.306**	**6.56**

Control parameterization. A direct parameterization of u is straightforward but starts fine-tuning from a randomly initialized control. To instead initialize from the pretrained model, we rewrite the pretrained generative dynamic (3) as a controlled dynamic around the redesigned base drift (17):

$$\mathrm{d}X_t = D(t)X_t\mathrm{d}t + \sigma(t)\left(\frac{1}{\sigma(t)}\left(-D(t)X_t - \frac{1}{t}X_t + 2v^{\mathrm{pt}}(X_t,t)\right)\right)\mathrm{d}t + \sigma(t)\mathrm{d}W_t, \quad X_0 \sim \mathcal{N}(0,I). \tag{22}$$

This decomposition suggests parameterizing the control through a trainable velocity model:

$$u(x,t) = \frac{1}{\sigma(t)}\left(-D(t)x - \frac{1}{t}x + 2v^{\mathrm{ft}}(x,t)\right), \tag{23}$$

where v^{ft} is initialized from the pretrained model v^{pt}. When $v^{\mathrm{ft}} = v^{\mathrm{pt}}$, the controlled dynamic (5) exactly recovers the pretrained generative dynamic (3). Thus, this parameterization lets fine-tuning start from the pretrained model while learning the control induced by the redesigned base drift.

Training objective. Our final loss objective is given by Eq. (13), with $p^{\mathrm{base}}_{t|1}$, $D(t)$, $\nabla g(x)$, and $u(x,t)$ specified in Eq. (14), Eq. (17), Eq. (21), and Eq. (23), respectively.

Practical implementation. For numerical stability, we apply timestep-dependent reward scaling, using $w(t) = (1-t)^{0.9}/t^{1.5}$ for the reward term and its inverse for loss weighting.

Summary. Tab. 1 summarizes the difference between AM and our EAM. EAM effectively removes main computational bottlenecks in AM, *i.e.*, adjoint simulation and stochastic trajectory simulation via redesigning the base dynamic. As shown in Algorithm 1, we simulate X_1 with efficient ODE solver (15) and construct the intermediate state X_t by sampling from the original noising kernel $q_t(\cdot|X_1)$ Then, we minimize adjoint matching loss (13), which does not need adjoint ODE simulation.

4 Experiments

4.1 Experimental Settings

Setup. We fine-tune Stable Diffusion 3.5-Medium (SD3.5-M) [9] by training LoRA [20] weights of rank 32 to generate images at 512×512 resolution, using Pick-a-Pic [22] as the training prompt set. We conduct all experiments on 4 NVIDIA A100 GPUs. We consider two reward settings: a single-reward setting using PickScore [22], and a multi-reward setting combining PickScore, HPSv2.1 [39], and Aesthetics [32]. All fine-tuned models are trained for one epoch with an effective batch size of 512 using AdamW [27], with learning rate 1×10^{-4} and momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. AM uses 40 NFEs while EAM uses 10 NFEs for the trajectory simulation. Unless otherwise stated, we set the reward scale to $\beta = 2000$ for both AM and EAM, and use $C = 0.51$ for EAM. We evaluate on DrawBench [31], generating images with 10 NFEs using DPM-Solver++(2M) [28].

Evaluation metrics. We evaluate fine-tuned models using complementary metrics for human preference and prompt alignment. PickScore [22], ImageReward [42], and HPSv2.1 [39] estimate human preference by jointly considering the prompt and generated image. Aesthetics [32] measures visual appeal from image embeddings, while CLIPScore [17] measures image–text compatibility.

Table 3: **Source of EAM's efficiency gain** (SD3.5-M). (a) Matched-NFE comparison. (b) Per-iteration time and peak memory of the trajectory and adjoint phases only; costs shared by both methods (*e.g.*, reward gradient, loss backpropagation) are excluded.

(a) Matched NFE

Method	NFE	GPU hrs ↓	PickScore ↑
AM (SDE)	40	64	23.00
AM (SDE)	10	80	23.03
EAM (ODE)	10	**16**	**23.16**

(b) Per-iteration cost

	Time (s) ↓		Mem. (GB) ↓	
Phase	AM	EAM	AM	EAM
Trajectory	16.7	**4.5**	27.4	**27.2**
Adjoint	34.7	**0.4**	73.6	**1.6**

4.2 Main Results.

Quantitative results. As shown in Tab. 2, EAM consistently matches or outperforms AM across most metrics in both the single-reward and multi-reward settings. Both EAM and AM substantially improve over the pretrained SD3.5-M baseline, while SD3.5-M with Classifier-Free-Guidance (CFG) [19] attains the highest CLIPScore. Applying CFG to AM and EAM further improves most metrics except Aesthetics, as shown in Tab. 5 of App. B. However, it doubles the NFE by requiring both conditional and unconditional velocity evaluations at each sampling step during image generation. Results on additional backbones and resolutions are provided in App. C.1.

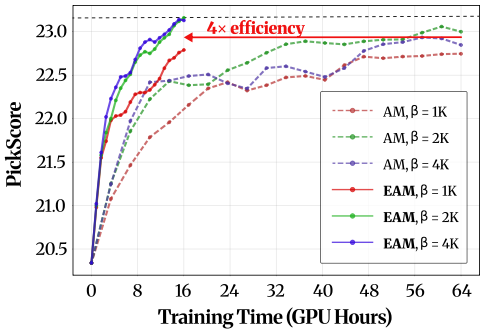


Figure 3: **PickScore on DrawBench by training time (GPU hours).** EAM converges significantly faster than AM (up to 4×).

Training efficiency. Fig. 3 shows the trade-off between training time and performance. EAM converges substantially faster than AM, reducing training time by up to 4× while achieving comparable or better performance. This improvement comes from two simplifications: replacing 40-step SDE trajectory simulation with 10-step ODE endpoint sampling, and computing the adjoint matching target in closed-form without backward simulation or costly JVP evaluations. First, AM cannot use an ODE sampler, as its adjoint must be simulated along a memoryless stochastic trajectory (7), and reducing its SDE steps instead slows convergence by degrading both the samples and the adjoint states simulated along them (Tab. 3a). In contrast, EAM only requires the endpoint X_1, which a few-step ODE solver generates without such degradation, reducing trajectory construction time by 73% (Tab. 3b). Second, the closed-form adjoint removes the backward simulation entirely, reducing the adjoint computation time by 99% and its peak memory from 73.6 GB to 1.6 GB.

Qualitative results. Fig. 2 compares images generated by SD3.5-M, AM, and EAM under the single-reward and multi-reward settings. Fine-tuning with PickScore improves image completeness for both AM and EAM, reducing visibly distorted or broken samples. When HPSv2.1 and Aesthetics are additionally used, both methods produce more polished images with improved prompt fidelity.

Compared with AM, EAM tends to better preserve fine-grained details and complex compositions in both reward settings. We attribute this to two consistency properties of our training procedure. First, EAM constructs each intermediate state X_t by sampling from the original noising kernel (2) conditioned on the generated endpoint X_1. Thus, the training pair (X_1, X_t) preserves the pretrained diffusion coupling between clean samples X_1 and noisy intermediate states X_t, aligning with the core perspective of DiffusionNFT [49]. Second, the endpoint distribution used during training matches the one used at evaluation, since both are obtained via ODE simulation with 10 NFEs. This reduces the mismatch between training and inference trajectories in EAM. As a result, EAM improves reward alignment while better retaining detailed structures, whereas AM often improves overall image quality but can smooth out fine details.

4.3 Ablation Study

Reward scale. As shown in Fig. 3, the reward scale β affects both convergence speed and final performance, and its optimal value may depend on the training setup. In our setting, $\beta = 2000$ gives

the best overall performance for both AM and EAM. Using a larger value, $\beta = 4000$, accelerates early convergence, but its final performance becomes comparable to that of $\beta = 2000$.

Role of the constant C in balancing loss components. To analyze the effect of C during training, we expand our loss objective in Eq. (13) and write the matching target for the trainable velocity v^{ft}, omitting the reward term:

Table 4: PickScore for different C.

C	0.501	0.51	1.0
PickScore	22.53	23.16	21.97

$$\frac{(1-t)(1-2Ct)}{2Ct^2-2t+1}(X_1-\epsilon)+\frac{t(2C-1)}{2Ct^2-2t+1}v^{\text{pt}}(X_t,t)-\frac{(1-t)(2C-1)}{2Ct^2-2t+1}\epsilon. \tag{24}$$

The constant C balances the three non-reward terms: $X_1 - \epsilon$, the pretrained velocity $v^{\text{pt}}(X_t, t)$, and the noise term ϵ. Larger C increases the weight on the pretrained velocity, while smaller C emphasizes $X_1 - \epsilon$. The pretrained velocity keeps v^{ft} close to the pretrained model, whereas $X_1 - \epsilon$ introduces additional exploration at X_t. In our experiments, $C = 0.51$ performs best; larger values, such as $C \geq 1$, make the ϵ term dominant and destabilize optimization. Further ablations on the terminal score estimator, the terminal cost correction, sampling NFE, and LoRA rank are in App. C.2.

5 Related Works

Reward-based fine-tuning methods for diffusion and flow models can be broadly grouped into two streams. *Reward-value-based* methods [2, 11, 10, 48, 25, 49, 5, 4] treat the reward as a black box and rely solely on its scalar values, adopting policy-gradient-style estimators inherited from reinforcement learning for large language models. A prominent line of such methods [25, 44, 16, 37, 36, 45, 43, 5, 49] requires generating 12 to 24 images for each prompt, making the optimization computationally expensive. *Reward-gradient-based* methods [6, 42, 29, 13, 38, 7], in contrast, utilize useful information from the gradient of reward functions. These methods can obtain an update from a single generated sample per prompt, enabling more sample-efficient training. Adjoint Matching (AM) [7] provides a theoretically grounded SOC formulation for targeting the exact reward-tilted distribution, and is the first to identify the necessity of a memoryless noise schedule in the base dynamic. Subsequent works have extended this framework: ASBS [24] generalizes the memoryless condition of the base dynamic, and TR-SOCM [3] enables a more stable optimization framework. While promising, training AM-based methods is compute-intensive due to the stochastic simulation for generating images and backward simulation for adjoint states. Our method simultaneously addresses both issues by adequately selecting the linear base drift, while retaining the theoretical guarantees of AM.

Adjoint Sampling [15] is similar in spirit to our work, but it targets a different task and solves it under a distinct setting and formulation. Its goal is to sample from a Boltzmann distribution starting from a Dirac prior, which is achieved by adopting a zero base drift. In contrast, in the standard fine-tuning setup considered in Eq. (6), the base drift includes the pretrained velocity. Therefore, simplifying the formulation is not immediate and requires satisfying additional structural conditions. Our method addresses this by redesigning the base dynamic so that its drift is linear, the dynamic is memoryless, and its endpoint-conditioned base kernel $p^{\text{base}}_{t|1}$ matches the original noising kernel $q_{t|1}$.

6 Conclusion

We introduce Efficient Adjoint Matching (EAM), an efficient reward fine-tuning method for diffusion models. EAM is based on the observation that the two main bottlenecks of AM, forward trajectory simulation and backward adjoint simulation, stem from the choice of the base drift. By redesigning the base drift to be linear and memoryless, EAM replaces full trajectory simulation with endpoint-conditioned noising and computes the adjoint state in closed-form. With a rederived terminal cost, the redesigned SOC problem still targets the desired reward-tilted distribution. Experiments show that EAM matches or improves AM with up to 4× faster convergence.

Limitations. Exploring practical optimization strategies, such as buffer replay, old-policy trajectory generation, and adaptive hyperparameter schedules, is an important direction for future work.

Acknowledgments

We thank Guan-Horng Liu for his assistance, insightful discussions and comments on the manuscript. This research was supported by Youlchon Foundation, the National Research Foundation of Korea (NRF) grants (RS-2021-NR05515, RS-2023-0022663, RS-2024-00336576, RS-2024-00349646, RS-2025-25402648, RS-2025-25406186, RS-2026-25506577), by the Institute for Information & Communication Technology Planning & Evaluation (IITP) grants (RS-2022-II220264, RS-2024-00353131), by the National IT Industry Promotion Agency (NIPA) grant (RS-2026-25621677), and by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant (RS-2026-25525550) funded by the Korean government. This research was also supported by the High-Performance Research AI Computing Infrastructure Support (RS-2026-25505492) from IITP. W. G. and Y. C. are supported by NSF 2206576 and NSF 2450378.

References

[1] M. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 2025.

[2] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. In *ICLR*, 2024.

[3] D. Blessing, J. Berner, L. Richter, C. Domingo-Enrich, Y. Du, A. Vahdat, and G. Neumann. Trust region constrained measure transport in path space for stochastic optimal control and inference. In *NeurIPS*, 2025.

[4] J. Choi, W. Guo, Y. Zhu, A. Vahdat, M. Tao, J. Berner, and Y. Chen. Scaling reinforcement learning for diffusion models via velocity matching. *arXiv preprint arXiv:2608.23664*, 2026.

[5] J. Choi, Y. Zhu, W. Guo, P. Molodyk, B. Yuan, J. Bai, Y. Xin, M. Tao, and Y. Chen. Rethinking the design space of reinforcement learning for diffusion models: On the importance of likelihood estimation beyond loss design. In *ICML*, 2026.

[6] K. Clark, P. Vicol, K. Swersky, and D. J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *ICLR*, 2024.

[7] C. Domingo-Enrich, M. Drozdzal, B. Karrer, and R. T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *ICLR*, 2025.

[8] B. Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[9] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, and R. Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

[10] Y. Fan and K. Lee. Optimizing DDPM sampling with shortcut fine-tuning. In *ICML*, 2023.

[11] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *NeurIPS*, 2023.

[12] W. Guo, J. Choi, Y. Zhu, M. Tao, and Y. Chen. Proximal diffusion neural sampler. In *ICML*, 2026.

[13] X. Guo, M. Cui, L. Bo, and D. Huang. ShortFT: Diffusion model alignment via shortcut-based fine-tuning. In *ICCV*, 2025.

[14] A. Havens, B. Karrer, and N. Shaul. Flow sampling: Learning to sample from unnormalized densities via denoising conditional processes. In *ICML*, 2026.

[15] A. Havens, B. K. Miller, B. Yan, C. Domingo-Enrich, A. Sriram, B. Wood, D. Levine, B. Hu, B. Amos, B. Karrer, X. Fu, G.-H. Liu, and R. T. Q. Chen. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. In *ICML*, 2025.

[16] X. He, S. Fu, Y. Zhao, W. Li, J. Yang, D. Yin, F. Rao, and B. Zhang. TempFlow-GRPO: When timing matters for grpo in flow models. In *ICLR*, 2026.

[17] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

[18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[19] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv:2207.12598*, 2022.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[21] S. Kim, J. Hahm, J. Shin, and J. Lee. Equivariant latent alignment via flow matching under group symmetries. In *ICML*, 2026.

[22] Y. Kirstain, A. Polyak, U. Singer, S. Matiana, J. Penna, and O. Levy. Pick-a-Pic: An open dataset of user preferences for text-to-image generation. In *NeurIPS*, 2023.

[23] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *ICLR*, 2023.

[24] G.-H. Liu, J. Choi, Y. Chen, B. K. Miller, and R. T. Q. Chen. Adjoint schrödinger bridge sampler. In *NeurIPS*, 2025.

[25] J. Liu, G. Liu, J. Liang, Y. Li, J. Liu, X. Wang, P. Wan, D. Zhang, and W. Ouyang. Flow-GRPO: Training flow matching models via online rl. In *NeurIPS*, 2025.

[26] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[27] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[28] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 2025.

[29] M. Prabhudesai, A. Goyal, D. Pathak, and K. Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv:2310.03739*, 2024.

[30] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[31] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, J. Ho, D. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

[32] C. Schuhmann. LAION-AESTHETICS, Aug. 2022. https://laion.ai/blog/laion-aesthetics/. Accessed: 2026-04-30.

[33] Y. Shi, V. De Bortoli, A. Campbell, and A. Doucet. Diffusion schrödinger bridge matching. In *NeurIPS*, 2023.

[34] J. Shin, J. Sul, J. Lee, J. Choi, and J. Choi. Efficient generative modeling beyond memoryless diffusion via adjoint schrödinger bridge matching. In *ICML*, 2026.

[35] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[36] J. Wang, J. Liang, J. Liu, H. Liu, G. Liu, J. Zheng, W. Pang, A. Ma, Z. Xie, X. Wang, M. Wang, P. Wan, and X. Liang. GRPO-Guard: Mitigating implicit over-optimization in flow matching via regulated clipping. *arXiv:2510.22319*, 2025.

[37] Y. Wang, Z. Li, Y. Zang, Y. Zhou, J. Bu, C. Wang, Q. Lu, C. Jin, and J. Wang. Pref-GRPO: Pairwise preference reward-based grpo for stable text-to-image reinforcement learning. *arXiv:2508.20751*, 2025.

[38] X. Wu, Y. Hao, M. Zhang, K. Sun, Z. Huang, G. Song, Y. Liu, and H. Li. Deep reward supervisions for tuning text-to-image diffusion models. In *ECCV*, 2024.

[39] X. Wu, K. Sun, F. Zhu, R. Zhao, and H. Li. Human preference score: Better aligning text-to-image models with human preference. In *ICCV*, 2023.

[40] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. In *ICLR*, 2025.

[41] E. Xie, J. Chen, Y. Zhao, J. Yu, L. Zhu, C. Wu, Y. Lin, Z. Zhang, M. Li, J. Chen, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. In *ICML*, 2025.

[42] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *NeurIPS*, 2023.

[43] S. Xue, C. Ge, S. Zhang, Y. Li, and Z.-M. Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv:2509.25050*, 2025.

[44] Z. Xue, J. Wu, Y. Gao, F. Kong, L. Zhu, M. Chen, Z. Liu, W. Liu, Q. Guo, W. Huang, and P. Luo. DanceGRPO: Unleashing grpo on visual generation. *arXiv:2505.07818*, 2025.

[45] H. Ye, K. Zheng, J. Xu, P. Li, H. Chen, J. Han, S. Liu, Q. Zhang, H. Mao, Z. Hao, P. Chattopadhyay, D. Yang, L. Feng, M. Liao, J. Bai, M.-Y. Liu, J. Zou, and S. Ermon. Data-regularized reinforcement learning for diffusion models at scale. *arXiv:2512.04332*, 2025.

[46] Q. Zhang and Y. Chen. Fast sampling of diffusion models with exponential integrator. In *ICLR*, 2023.

[47] Q. Zhang, M. Tao, and Y. Chen. gddim: generalized denoising diffusion implicit models. In *ICLR*, 2023.

[48] H. Zhao, H. Chen, J. Zhang, D. D. Yao, and W. Tang. Score as action: Fine-tuning diffusion generative models by continuous-time reinforcement learning. In *ICML*, 2025.

[49] K. Zheng, H. Chen, H. Ye, H. Wang, Q. Zhang, K. Jiang, H. Su, S. Ermon, J. Zhu, and M.-Y. Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *ICLR*, 2026.

Appendix

A Proofs

A.1 Proof of Closed-form Adjoint in Case of Linear Drift in Eq. (10)

By definition, the lean adjoint state $a(t; X_t)$ satisfies the ODE in Eq. (9):

$$\frac{\mathrm{d}}{\mathrm{d}t}\, a(t; X_t) \;=\; -\nabla_x b(X_t, t)^\top\, a(t; X_t), \qquad a(1; X_1) = \nabla g(X_1). \tag{25}$$

When the base drift takes the linear form $b(x, t) = D(t)\, x$, its Jacobian is $\nabla_x b(X_t, t) = D(t)\, I$, so the ODE reduces to a scalar linear ODE in $a(t; X_t)$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\, a(t; X_t) \;=\; -\,D(t)\, a(t; X_t). \tag{26}$$

Integrating backward from $t = 1$ with terminal state $a(1; X_1)$ yields

$$a(t; X_t) \;=\; \exp\!\left(\int_t^1 D(\tau)\,\mathrm{d}\tau\right) a(1; X_1), \tag{27}$$

which is Eq. (10). The prefactor depends only on t, so $a(t; X_t)$ is determined by the endpoint X_1 alone via $\nabla g(X_1)$, with no JVP through the velocity network. □

A.2 Proof of (X_t, X_1) Factorization in Eq. (12)

We show that the joint distribution $p^u(X_t, X_1)$ admits the factorization

$$p^u(X_t, X_1) \;=\; p^{\mathrm{base}}(X_t \mid X_1)\, p_1^u(X_1), \tag{28}$$

under the optimal control $u^\star$.

Step 1: reciprocal projection. Marginalizing the joint (X_t, X_0, X_1) over X_0 gives

$$p^u(X_t, X_1) \;=\; \int p^u(X_t \mid X_0, X_1)\, p^u(X_0, X_1)\,\mathrm{d}X_0. \tag{29}$$

Reciprocal projection [33, 15] replaces the controlled bridge by the base bridge,

$$p^u(X_t \mid X_0, X_1) \;=\; p^{\mathrm{base}}(X_t \mid X_0, X_1), \tag{30}$$

which holds at optimality. Then, the joint distribution becomes

$$p^u(X_t, X_1) \;=\; \int p^{\mathrm{base}}(X_t \mid X_0, X_1)\, p^u(X_0, X_1)\,\mathrm{d}X_0. \tag{31}$$

Step 2: memorylessness at optimality. When the base dynamic is memoryless (7), controlled dynamic with optimal control is also memoryless [34], *i.e.*,

$$p^{u^\star}(X_0, X_1) \overset{\text{memoryless}}{=} p_0^{\mathrm{base}}(X_0)\, p_1^{u^\star}(X_1). \tag{32}$$

As a result, under the SOC optimality, the joint distribution factors as

$$\begin{aligned} p^u(X_t, X_1) \;&=\; \int p^{\mathrm{base}}(X_t \mid X_0, X_1)\, p^u(X_0, X_1)\,\mathrm{d}X_0 && (33)\\ &=\; \int \frac{p^{\mathrm{base}}(X_t, X_0, X_1)}{p_0^{\mathrm{base}}(X_0) p_1^{\mathrm{base}}(X_1)}\, p_0^{\mathrm{base}}(X_0) p_1^u(X_1)\,\mathrm{d}X_0 && (34)\\ &=\; p^{\mathrm{base}}(X_t \mid X_1)\, p^u(X_1), && (35) \end{aligned}$$

which is Eq. (28). □

A.3 Proof of Proposition 3.1

We work with the linear base SDE

$$\mathrm{d}X_t \;=\; D(t)\,X_t\,\mathrm{d}t \;+\; \sigma(t)\,\mathrm{d}W_t, \qquad X_0 \sim \mathcal{N}(0, I), \qquad \sigma(t) = \sqrt{\tfrac{2(1-t)}{t}}. \tag{36}$$

Throughout this proof, we use the abbreviations

$$\Phi_t \;:=\; \exp\!\left(\int_t^1 D(s)\,\mathrm{d}s\right), \qquad I_t \;:=\; \int_0^t \exp\!\left(2\int_r^t D(u)\,\mathrm{d}u\right)\frac{1-r}{r}\,\mathrm{d}r, \tag{37}$$

and adopt the standard convention $J_t := I_1 - I_t\,\Phi_t^2$.

Step 1: bridge of a linear SDE. The solution of Eq. (36) is $X_t = \alpha_t X_0 + \beta_t\,\varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$, where $\alpha_t = \exp(\int_0^t D)$ and $\beta_t^2 = \sigma(t)^2$-driven variance. A direct calculation gives the bridge distribution

$$p(X_t \mid X_0, X_1) \;=\; \mathcal{N}\big(\bar{\alpha}_t X_0 + \bar{\beta}_t X_1,\; \gamma_t^2\, I\big), \tag{38}$$

with bridge coefficients $\bar{\alpha}_t = \Phi_0 J_t/(\Phi_t I_1)$, $\bar{\beta}_t = \Phi_t I_t/I_1$, and $\gamma_t^2 = 2\,I_t J_t/I_1$.

Step 2: reciprocal kernel under Gaussian prior. Since $X_0 \sim \mathcal{N}(0, I)$, marginalizing over X_0 yields

$$X_t \mid X_1 \;\sim\; \mathcal{N}\big(\bar{\beta}_t X_1,\; (\bar{\alpha}_t^2 + \gamma_t^2)\, I\big). \tag{39}$$

Step 3: matching the perturbation kernel. We require Eq. (39) to coincide with the perturbation kernel $\mathcal{N}(tX_1, (1-t)^2 I)$ of Eq. (2), i.e.,

$$\bar{\beta}_t = t, \qquad \bar{\alpha}_t^2 + \gamma_t^2 = (1-t)^2. \tag{40}$$

Step 4: solving the first constraint. Substituting $\bar{\beta}_t = \Phi_t I_t/I_1$, the constraint $\bar{\beta}_t = t$ becomes $\Phi_t I_t = t\,I_1$. Differentiating both sides in t and using Leibniz's rule with $\Phi_t' = -D(t)\,\Phi_t$ and $I_t' = (1-t)/t + 2D(t)\,I_t$, we obtain

$$I_1 \;=\; \Phi_t\left(\frac{1-t}{t} + D(t)\,I_t\right), \tag{41}$$

which, after substituting $I_1 = \Phi_t I_t/t$, gives a closed-form expression for $D(t)$:

$$D(t) \;=\; \frac{I_t - (1-t)}{t\,I_t}. \tag{42}$$

Plugging Eq. (42) back into the ODE for I_t yields

$$\frac{\mathrm{d}I_t}{\mathrm{d}t} - \frac{2}{t}\,I_t \;=\; -\frac{1-t}{t}. \tag{43}$$

Multiplying by the integrating factor t^{-2} gives $\frac{\mathrm{d}}{\mathrm{d}t}(t^{-2}I_t) = -(1-t)/t^3$, and integrating produces

$$I_t \;=\; \tfrac{1}{2} - t + C\,t^2, \qquad C \in \mathbb{R}. \tag{44}$$

The constraint $I_t > 0$ on $(0,1)$ requires $C > \frac{1}{2}$. Substituting this I_t into Eq. (42) yields the announced family

$$D(t) \;=\; \frac{2Ct^2 - 1}{t\,(2Ct^2 - 2t + 1)}, \qquad C > \tfrac{1}{2}, \tag{45}$$

which is Eq. (17).

Step 5: the second constraint is automatic. Using the explicit forms of I_t, $\Phi_t = (2C-1)\,t/(2Ct^2 - 2t + 1)$, and $J_t = I_1 - I_t\,\Phi_t^2$, a direct computation (using $\bar{\alpha}_t = \Phi_0 J_t/(\Phi_t I_1)$ and $\gamma_t^2 = 2I_t J_t/I_1$) verifies that $\bar{\alpha}_t^2 + \gamma_t^2 = (1-t)^2$ holds identically for every $C > \frac{1}{2}$. The second constraint in Eq. (40) therefore imposes no additional restriction on $D(t)$.

Step 6: terminal distribution. At $t = 1$, Φ_0 is the value of Φ at $t = 0$, which evaluates to $\Phi_0 = 0$ (since the numerator $(2C-1)\,t$ vanishes at $t = 0$). Hence $\bar{\alpha}_1 = 0$, and $X_1 = \bar{\beta}_1 X_1 + \gamma_1\,\varepsilon$ marginalizes (using $X_0 \sim \mathcal{N}(0, I)$ and $\bar{\beta}_1 = 1$) to

$$p_1^{\text{base}} \;=\; \mathcal{N}\big(0,\; (2C-1)\,I\big), \tag{46}$$

since $I_1 = C - \frac{1}{2}$ and $2I_1 = 2C - 1$.

Step 7: memorylessness and uniqueness. The joint (X_0, X_1) has $\bar{\alpha}_1 = 0$, so X_1 depends on X_0 only through $\bar{\beta}_1 X_1$ and the independent noise $\gamma_1\varepsilon$; equivalently, $p_{0,1}^{\text{base}} = p_0^{\text{base}}\, p_1^{\text{base}}$, which is the memoryless condition (7).

For the converse, the derivation of Steps 4–5 shows that any linear drift $D(t)$ with the fixed $\sigma(t)$ that satisfies Eq. (40) must take the form of Eq. (17) for some $C > \frac{1}{2}$. The family is therefore unique up to the single scalar C. □

Exact adjoint calculation. Plugging the linear base drift in Eq. (17) into Eq. (10) yields

$$a(t; X_t) = \frac{(2C-1)\,t}{2Ct^2 - 2t + 1}\, a(1; X_1), \qquad a(1; X_1) = \nabla g(X_1). \tag{47}$$

A.4 Proof of Proposition 3.2

We use the standard SOC reduction for reward-tilted sampling [7, 15]: under the SOC problem (4)–(5) with terminal cost g, the optimal control $u^\star$ steers the controlled dynamic so that its terminal distribution is

$$p_1^{u^\star}(x) \;\propto\; p_1^{\text{base}}(x)\, \exp\big(-g(x)\big). \tag{48}$$

Designing g for a target tilt. We want the terminal distribution to be the reward-tilted target $p_1^{u^\star}(x) \propto p_1^{\text{pt}}(x)\exp(\beta r(x))$. Equating with Eq. (48),

$$p_1^{\text{base}}(x)\, \exp\big(-g(x)\big) \;\propto\; p_1^{\text{pt}}(x)\, \exp\big(\beta r(x)\big), \tag{49}$$

which gives, up to an additive constant absorbed in normalization,

$$g(x) \;=\; \log p_1^{\text{base}}(x) \;-\; \log p_1^{\text{pt}}(x) \;-\; \beta\, r(x). \tag{50}$$

Taking the gradient,

$$\nabla g(x) \;=\; \nabla \log p_1^{\text{base}}(x) \;-\; \nabla \log p_1^{\text{pt}}(x) \;-\; \beta\, \nabla r(x). \tag{51}$$

Specialization to AM. Under the standard SOC-based reward fine-tuning setup of Eq. (6), the base dynamic coincides with the pretrained generative dynamic, hence $p_1^{\text{base}} = p_1^{\text{pt}}$. The first two terms in Eq. (51) cancel, and we recover the AM gradient

$$\nabla g(x) \;=\; -\beta\, \nabla r(x). \tag{52}$$

Specialization to our reformulation. By Proposition 3.1, our reformulated base dynamic has terminal distribution $p_1^{\text{base}} = \mathcal{N}\big(0, (2C-1)I\big)$, so

$$\nabla \log p_1^{\text{base}}(x) \;=\; -\frac{x}{2C-1}. \tag{53}$$

Substituting into Eq. (51) yields

$$\nabla g(x) \;=\; -\frac{x}{2C-1} \;-\; \nabla \log p_1^{\text{pt}}(x) \;-\; \beta\, \nabla r(x), \tag{54}$$

which is Eq. (18). □

A.5 Proof of Theorem 3.3

Setup. We use the superscript $m \in \{\mathrm{AM}, \mathrm{EAM}\}$ to distinguish the two SOC problems, and write $p^{\mathrm{ref},m}$ for the corresponding base path measure:

- AM: $p^{\mathrm{ref,AM}} = p^{\mathrm{pt}}$, with drift $b^{\mathrm{AM}}(x,t) = -\frac{x}{t} + 2v^{\mathrm{pt}}(x,t)$ and terminal cost $g^{\mathrm{AM}} = -\beta r$ (6);
- EAM: $p^{\mathrm{ref,EAM}} = p^{\mathrm{base}}$, with drift $b^{\mathrm{EAM}}(x,t) = D(t)x$ (17) and terminal cost g^{EAM} in Eq. (18).

Both share the diffusion coefficient $\sigma(t)$ and the initial distribution $p_0 = \mathcal{N}(0, I)$. As in Sec. 2, we adopt the standard idealization that v^{pt} is exact. Under this idealization, the pretrained dynamic (3) is the time reversal of the Markov noising process with kernel q_t, so that $p^{\mathrm{pt}}_{t|1} = q_t$ and $p^{\mathrm{pt}}_t(x) = \int q_t(x \mid x_1)\, p_{\mathrm{data}}(x_1)\, \mathrm{d}x_1$. For EAM, $p^{\mathrm{base}}_{t|1} = q_t$ holds by **Condition 3** (14). Thus, in both cases,

$$p^{\mathrm{ref},m}_{t|1} = q_t, \qquad p^{\mathrm{ref},m}_{0,1} = p_0\, p^{\mathrm{ref},m}_1, \tag{55}$$

where the second identity is the memoryless condition (7).

Step 1: optimal endpoint coupling. The optimal path measure of the SOC problem (4)–(5) is an endpoint tilt of the base path measure [7]:

$$p^{u^\star_m}(X) \;\propto\; p^{\mathrm{ref},m}(X)\, e^{-g^m(X_1)}. \tag{56}$$

Marginalizing onto (X_0, X_1) and using memorylessness (55),

$$p^{u^\star_m}_{0,1}(x_0, x_1) \;\propto\; p_0(x_0)\, p^{\mathrm{ref},m}_1(x_1)\, e^{-g^m(x_1)} \;\propto\; p_0(x_0)\, p^\star_1(x_1), \tag{57}$$

where the last step uses $p^{\mathrm{pt}}_1 e^{\beta r} \propto p^\star_1$ for AM and Proposition 3.2 for EAM. Hence, in both cases, X_0 and X_1 are independent at optimality with $X_1 \sim p^\star_1$.

Step 2: optimal single-time marginals coincide. By Eq. (56), the optimal path measure shares the base bridge $p^{\mathrm{ref},m}_{t|0,1}$. Combining this with Step 1 and Eq. (55),

$$p^{\star,m}_t(x) = \iint p^{\mathrm{ref},m}_{t|0,1}(x \mid x_0, x_1)\, p_0(x_0)\, p^\star_1(x_1)\, \mathrm{d}x_0\, \mathrm{d}x_1 = \int q_t(x \mid x_1)\, p^\star_1(x_1)\, \mathrm{d}x_1, \tag{58}$$

where the inner integral over x_0 equals $p^{\mathrm{ref},m}_{t|1} = q_t$ because X_0 and X_1 are independent under the base. The right-hand side does not depend on m, so $p^{\star,\mathrm{AM}}_t = p^{\star,\mathrm{EAM}}_t =: p^\star_t$ for all $t \in (0,1)$.

Step 3: optimal drifts coincide. The optimal control is $u^\star_m = \sigma \nabla_x \log h^m$ with $h^m(x,t) = \mathbb{E}_{p^{\mathrm{ref},m}}[e^{-g^m(X_1)} \mid X_t = x]$. By Eq. (56) and Bayes' rule, $p^\star_t \propto h^m_t\, p^{\mathrm{ref},m}_t$. Hence the optimal drift $b^m_\star = b^m + \sigma u^\star_m$ satisfies

$$b^m_\star(x,t) = \Big[b^m(x,t) - \sigma^2(t) \nabla_x \log p^{\mathrm{ref},m}_t(x)\Big] + \sigma^2(t) \nabla_x \log p^\star_t(x). \tag{59}$$

By Step 2, the last term is common to both methods. It remains to compare the bracketed term.

AM. By Tweedie's formula (20), $\sigma^2(t)\nabla_x \log p^{\mathrm{pt}}_t(x) = 2v^{\mathrm{pt}}(x,t) - \frac{2x}{t}$, so

$$b^{\mathrm{AM}}(x,t) - \sigma^2(t)\nabla_x \log p^{\mathrm{pt}}_t(x) = \Big(-\frac{x}{t} + 2v^{\mathrm{pt}}(x,t)\Big) - \Big(2v^{\mathrm{pt}}(x,t) - \frac{2x}{t}\Big) = \frac{x}{t}. \tag{60}$$

EAM. Since $p^{\mathrm{base}}_{t|1} = q_t$ and $p^{\mathrm{base}}_1 = \mathcal{N}(0, (2C-1)I)$, we have $p^{\mathrm{base}}_t = \mathcal{N}(0, N(t)I)$ with $N(t) := t^2(2C-1) + (1-t)^2 = 2Ct^2 - 2t + 1$. Therefore,

$$b^{\mathrm{EAM}}(x,t) - \sigma^2(t)\nabla_x \log p^{\mathrm{base}}_t(x) = \frac{2Ct^2 - 1}{t\, N(t)}\, x + \frac{2(1-t)}{t\, N(t)}\, x = \frac{x}{t}. \tag{61}$$

In both cases the bracketed term equals x/t: the pretrained velocity cancels for AM, and all C-dependence cancels for EAM. Substituting into Eq. (59),

$$b^{\mathrm{AM}}_\star(x,t) = b^{\mathrm{EAM}}_\star(x,t) = \frac{x}{t} + \sigma^2(t)\nabla_x \log p^\star_t(x). \tag{62}$$

Table 5: **Ablation over CFG.** We apply the same guidance scale $\omega = 2$ at inference time and following the formula $(1+\omega)v(x_t \mid c) - \omega\, v(x_t)$ from [19]. Applying CFG improves all the metrics except Aesthetics.

Method	PickScore	CLIPScore	ImageReward	HPSv2.1	Aesthetics
SD3.5-M	20.34	0.226	-0.58	0.214	5.29
SD3.5-M + CFG	21.82	**0.275**	0.65	0.265	5.41
AM	23.00	0.259	0.97	0.292	<u>6.25</u>
AM + CFG	23.05	**0.275**	<u>1.15</u>	<u>0.297</u>	5.90
Ours	<u>23.16</u>	0.252	0.95	<u>0.297</u>	**6.27**
Ours + CFG	**23.25**	<u>0.273</u>	**1.19**	**0.309**	5.99

Step 4: conclusion. The two optimal processes have the same drift, the same diffusion coefficient $\sigma(t)$, and the same initial distribution p_0. Under standard well-posedness assumptions, uniqueness in law for this SDE implies $p^{u^\star_{\mathrm{AM}}} = p^{u^\star_{\mathrm{EAM}}}$, and hence all multi-time joint distributions coincide. □

B Experiment Details

Detailed Setup. We fine-tune Stable Diffusion 3.5-Medium (SD3.5-M) [9] by training LoRA [20] weights of rank 32 to generate images at 512×512 resolution, using Pick-a-Pic [22] as the training prompt set. We conduct all experiments on 4 NVIDIA A100 GPUs. We consider two reward settings: a single-reward setting using PickScore [22], and a multi-reward setting combining PickScore, HPSv2.1 [39], and Aesthetics [32]. For multi-reward setting reported in Tab. 2, we set the reward scaling constant $\beta = 1000$, weighted combination of PickScore$= 1$, HPSv2.1$= 2 \times 26$, Aesthetics$=$ 0.05×26 for AM and $\beta = 2000$, weighted combination of PickScore$= 1$, HPSv2.1$= 0.5 \times 26$, Aesthetics$= 0.02 \times 26$ for EAM. The multi-reward coefficients and reward scaling constants were chosen to balance the gradient norms across rewards and were individually tuned for AM and EAM to achieve the best empirical performance for each method. All fine-tuned models are trained for one epoch with an effective batch size of 512 using AdamW [27], with learning rate 1×10^{-4} and momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. AM uses 40 NFEs for stochastic trajectory simulation, while EAM uses 10 NFEs for the ODE solver that samples X_1. For computing the matching loss in AM, we use four random intermediate states per trajectory, which improves the performance than using a single state. Unless otherwise stated, we set the reward scale to $\beta = 2000$ for both AM and EAM, and use $C = 0.51$ for EAM. We evaluate all models on DrawBench [31], generating images with 10 NFEs using DPM-Solver++(2M) [28].

Classifier-Free Guidance. We demonstrate the effect of applying Classifier-Free Guidance (CFG) [19] in this section. CFG is known to improve alignment between the generated image and the text prompt, but this comes at the cost of doubled NFEs required to compute the unconditional prediction. As shown in Tab. 5, applying CFG improves all the metrics except Aesthetics.

C Additional Experiments

C.1 Additional Backbones and Resolutions

To verify that EAM generalizes beyond SD3.5-M, we further fine-tune SANA 1.0 (1.6B, 512×512) [40] and SANA 1.5 (1.6B, 1024×1024) [41], which adopt a Linear DiT architecture instead of the MMDiT of SD3.5-M. We use PickScore as the reward and keep all hyperparameters identical to the SD3.5-M setting in App. B, including $C = 0.51$, without re-tuning. As shown in Tab. 6, EAM matches or outperforms AM on most metrics for both backbones while requiring $4\times$ fewer GPU hours, consistent with the SD3.5-M results.

C.2 Ablation Studies

Terminal score estimator. On SD3.5-M, we compare our estimator in Eq. (21) with two alternatives for $\nabla \log p_1^{\mathrm{pt}}(X_1)$. The *exact Tweedie* estimator applies Eq. (20) to a separately noised state $\tilde{X}_{\tilde{t}} =$

Table 6: **Results on SANA 1.0 and SANA 1.5.** All setting is identical to the SD3.5-M setting.

Method	GPU hrs ↓	PickScore	CLIPScore	ImageReward	HPSv2.1	Aesthetics
SANA 1.0 (512px)						
SANA 1.0	–	21.33	0.242	0.01	0.235	5.59
SANA 1.0 + CFG	–	22.55	**0.275**	**1.05**	0.296	5.70
AM	36	23.24	0.265	1.03	0.300	6.02
EAM (ours)	**9**	**23.30**	0.266	1.04	**0.302**	**6.15**
SANA 1.5 (1024px)						
SANA 1.5	–	21.49	0.249	0.16	0.244	5.67
SANA 1.5 + CFG	–	22.59	**0.277**	1.00	0.291	5.75
AM	120	23.36	0.273	**1.08**	0.300	6.14
EAM (ours)	**30**	**23.43**	0.267	1.05	**0.305**	**6.25**

$\tilde{t}X_1 + (1-\tilde{t})\epsilon$ with $\tilde{t} \approx 1$. The *ELBO* estimator [5] approximates the log-likelihood by the ELBO and differentiates it with respect to X_1:

$$\nabla \log p_1^{\mathrm{pt}}(X_1) \approx -\nabla_{X_1} \mathbb{E}_{\tilde{t},\epsilon}\left[\left\| v^{\mathrm{pt}}(\tilde{X}_{\tilde{t}}, \tilde{t}) - (X_1 - \epsilon)\right\|^2\right], \qquad \tilde{X}_{\tilde{t}} = \tilde{t}X_1 + (1-\tilde{t})\epsilon. \tag{63}$$

As shown in Tab. 7, both alternatives improve over the pretrained model, but our estimator performs best. Moreover, despite their improved PickScore, both alternatives often produce visibly corrupted images in practice, whereas ours does not. The exact Tweedie estimator relies on the velocity near $\tilde{t} \approx 1$, where the pretrained velocity is less reliable and its error is amplified by $1/(1-\tilde{t})$. The ELBO estimator affords only a few Monte Carlo samples per example, yielding a high-variance regression target that destabilizes training [21], and requires differentiating through v^{pt}. Our estimator avoids both issues by reusing (X_t, t), yielding a lower-variance target at the cost of some bias.

Table 7: **Terminal score estimator** (SD3.5-M).

	SD3.5-M	Exact Tweedie	ELBO	Ours
PickScore	20.34	21.78	21.54	**23.16**

Terminal cost correction. On SANA 1.0, we examine the role and robustness of the correction term in Eq. (18). First, we remove the correction and keep only the reward term, *i.e.*, $\nabla g = -\beta \nabla r$. By Proposition 3.2, the target then collapses to a reward-tilted Gaussian $p_1^{\mathrm{base}} e^{\beta r}$, and performance degrades substantially (Tab. 8a). Second, we perturb the estimated score $\hat{s}$ as $\hat{s} + \alpha\,\mathrm{RMS}(\hat{s})\,\xi$ with $\xi \sim \mathcal{N}(0, I)$, where α is the perturbation magnitude relative to the score. Performance remains stable up to $\alpha = 0.1$ and degrades gradually beyond it (Tab. 8b).

Table 8: **Terminal cost correction** (SANA 1.0). (a) Removing the correction. (b) Perturbing the estimated terminal score with relative magnitude α.

(a) Removal

	Ours	Reward only
PickScore	**23.30**	21.68

(b) Perturbation

α	0	0.05	0.1	0.3	0.5
PickScore	23.30	23.43	23.31	23.05	22.85

Sampling NFE and LoRA rank. On SANA 1.0, we vary the number of ODE steps used to sample X_1 during training and the LoRA rank, with $C = 0.51$ fixed. As shown in Tab. 9, EAM remains stable across all settings, and performance improves with both larger NFE and higher rank.

Table 9: **Sampling NFE and LoRA rank** (SANA 1.0). Default settings are underlined.

(a) Sampling NFE

NFE	5	10	20
PickScore	23.15	23.30	23.41

(b) LoRA rank

Rank	16	32	64
PickScore	23.07	23.30	23.42

D Additional Qualitative Examples

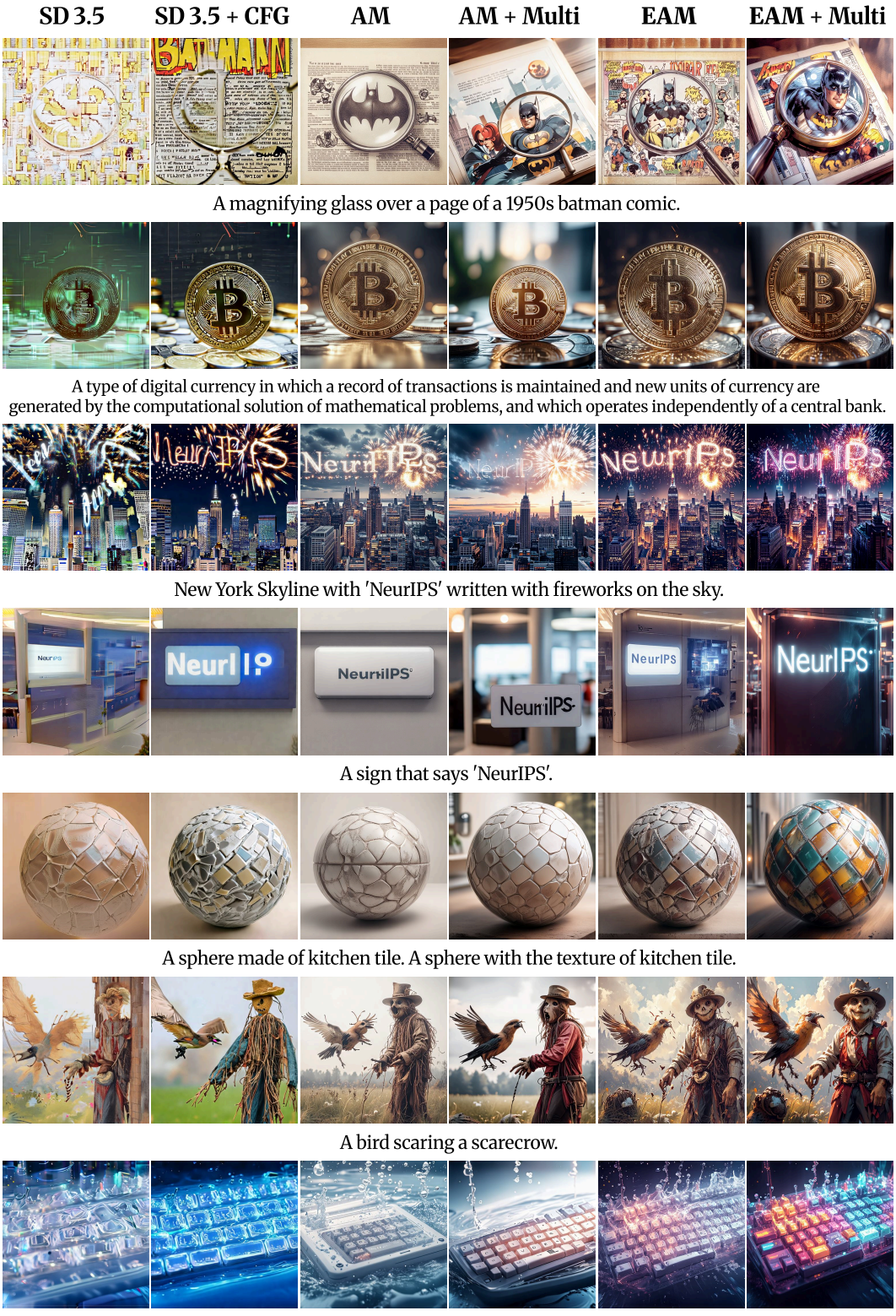


Figure 4: **Additional qualitative results.**

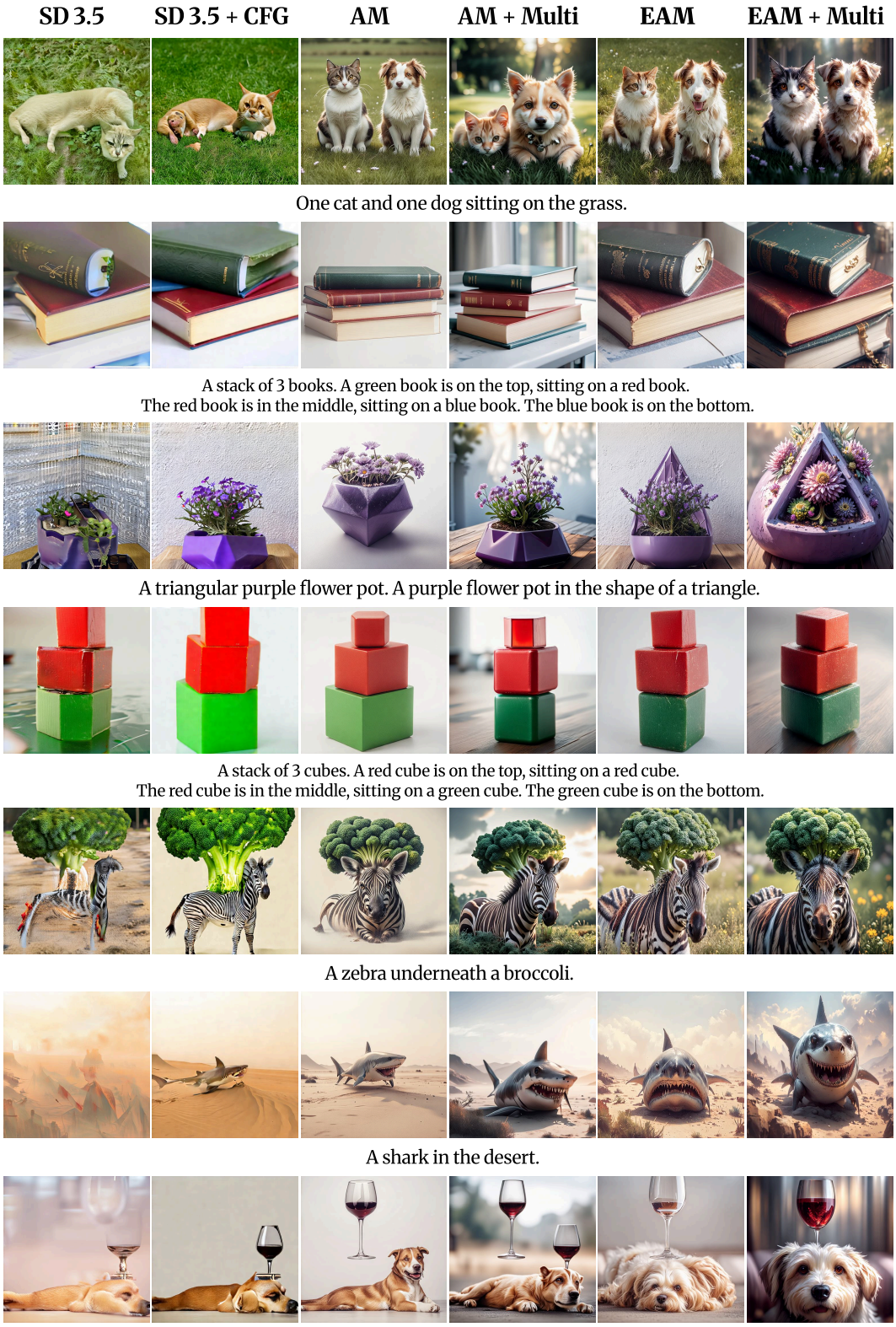


Figure 5: **Additional qualitative results.**

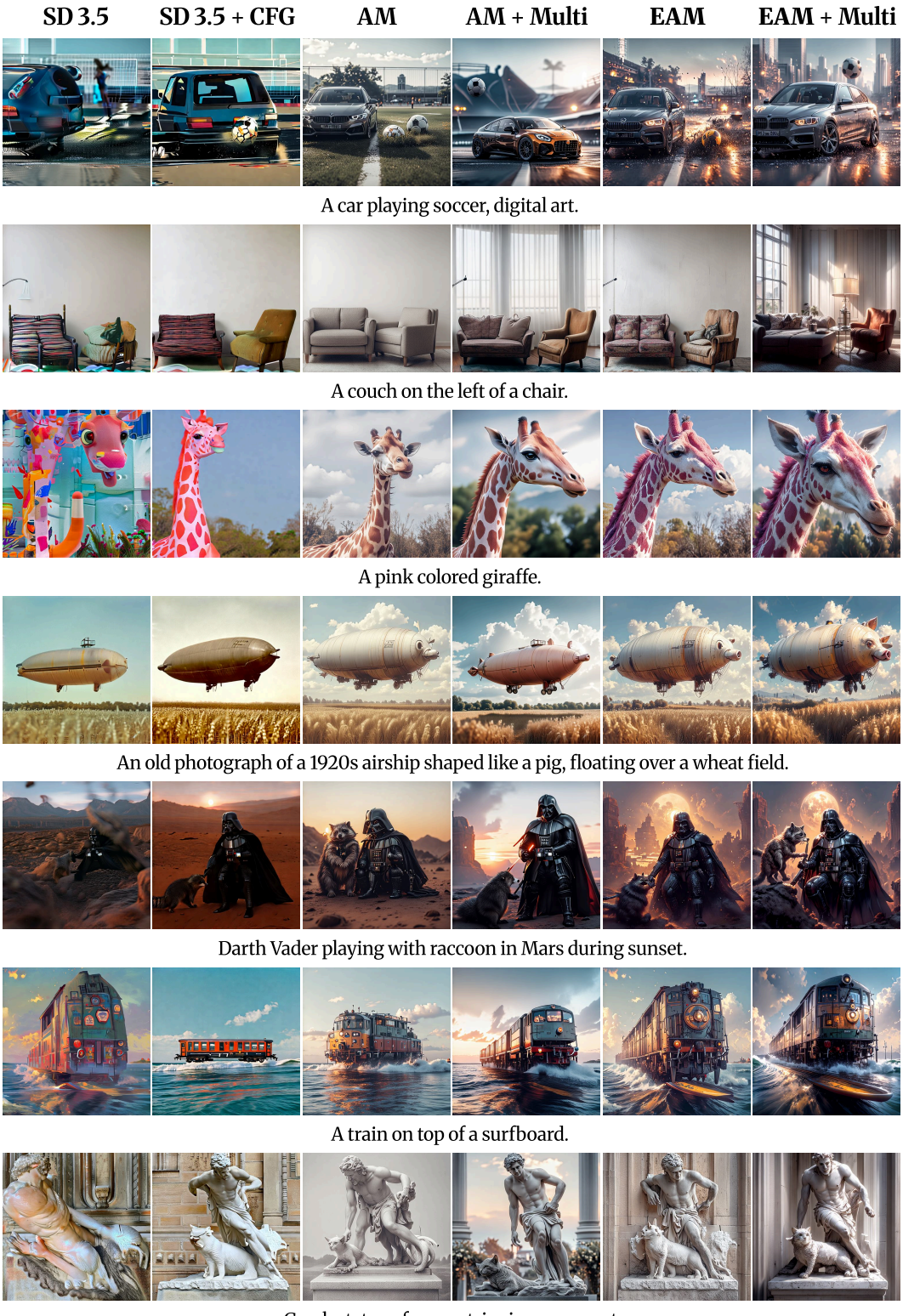


Figure 6: **Additional qualitative results.**