UNDERSTANDING AND MITIGATING STATE EXPOSURE MISMATCH IN FLOW MODEL REWARD ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) offers a flexible way to align pretrained flow models with task-specific objectives. Recent endpoint-based RL methods improve efficiency by reconstructing training states from rewarded endpoints, but these states may differ from policy-induced rollout states. We study this mismatch in state exposure during RL training of flow models. Our empirical analyses show that the discrepancy varies across timesteps, affects reward differently across denoising stages, and benefits from corrections aligned with individual trajectories. Guided by these observations, we propose Drift-Aligned Exposure Correction (DAEC), a simple method for mitigating this mismatch. DAEC perturbs only the trainable model input around analytically reconstructed states, using a readily available bridge residual to identify directions relevant to rollouts and adapting the perturbation magnitude to each timestep and sample, while preserving the original regression target and reference anchoring. DAEC introduces only lightweight tensor operations and requires no additional model evaluations per training update. Evaluations across diverse reward objectives demonstrate that DAEC improves both RL training efficiency and generation quality.

1 INTRODUCTION

Diffusion and flow-matching models have become leading paradigms for high-fidelity visual generation (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023). However, even strong pretrained models such as Stable Diffusion (SD3) (Esser et al., 2024) often fall short of task-specific requirements and human preferences, particularly in compositional generation and visual text rendering (Liu et al., 2025). Reward-based Reinforcement Learning (RL) post-training offers a flexible way to optimize these capabilities directly (Black et al., 2024; Liu et al., 2025). Yet applying RL to flow models remains challenging: iterative generation makes training costly, while reward optimization must improve task performance without driving the policy far from the pretrained reference distribution. Efficient and well-regularized post-training of flow models therefore remains an important problem.

Existing RL methods such as DDPO (Black et al., 2024) and Flow-GRPO (Liu et al., 2025) optimize diffusion and flow models along sampled reverse-process trajectories, achieving strong reward improvements but requiring stochastic rollouts and trajectory-wise training. To reduce these costs, recent endpoint-based RL methods, including Diffusion-NFT (Zheng et al., 2026), AWM (Xue et al., 2026), and RAM (Bergmeister et al., 2026), instead optimize pretraining-like objectives on states analytically reconstructed from generated endpoints. By avoiding trajectory-wise optimization and enabling endpoint reuse, this paradigm has been increasingly adopted for efficient post-training. However, how analytic reconstruction shapes the training-state distribution remains unexplored.

Although these methods use on-policy endpoints, their intermediate training states are analytically reconstructed rather than drawn from the policy's actual inference trajectories. This discrepancy is closely related to exposure bias in autoregressive generation (Ranzato et al., 2016) and diffusion models (Ning et al., 2023), arising from a mismatch between training and inference states. Consequently, reward-weighted supervision on reconstructed states may not transfer reliably to the states encountered during generation, which can lead to error accumulation and undermine reward optimization. This issue is particularly challenging in RL post-training because the policy-induced state distribution evolves throughout optimization, while any mitigation must preserve the balance between reward improvement and proximity to the pretrained reference model.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To investigate exposure bias in RL post-training, we conduct a series of empirical analyses in Sec. 3 and Fig. 1 and make three key observations: **(I)** the trajectory-consistent correction magnitude varies across timesteps and peaks at intermediate denoising timesteps (Sec. 3.2); **(II)** correcting this discrepancy has a timestep-dependent reward effect, with intermediate corrections improving terminal reward while near-clean ones may hurt (Sec. 3.3); and **(III)** the benefit of correction is trajectory-specific and directional, and cannot be explained by generic re-noising or arbitrary same-magnitude perturbations (Sec. 3.4). Together, these observations suggest that directly applying existing exposure-bias designs (Ning et al., 2023; 2024; Ren et al., 2024; Li et al., 2024a) may be insufficient in the RL setting: they do not jointly account for timestep- and trajectory-dependent effects or the reward–reference constraint. Motivated by these findings, we propose **Drift-Aligned Exposure Correction (DAEC)**, which augments state exposure around analytically reconstructed states. Specifically, DAEC uses the existing bridge residual as a rollout-relevant local proxy, modulates the exposure magnitude with a timestep profile and sample-wise gate, and perturbs only the trainable model input while preserving the original regression target and reference anchoring. We evaluate DAEC with diverse reward objectives, including GenEval (Ghosh et al., 2023), PickScore (Kirstain et al., 2023), and OCR-based text-rendering rewards (Liu et al., 2025). The results show consistent gains across the three reward objectives. Our main contributions are summarized as follows:

- Through three observations in RAM, we analyze the state exposure mismatch between analytically reconstructed training states and model-induced rollout states, revealing its timestep-dependent, reward-relevant, and trajectory-specific structure.
- We propose DAEC, a simple plug-in method that adjusts only the input state of the trainable model while preserving the original regression target and reference anchoring.
- Experiments across diverse reward objectives validate DAEC and demonstrate consistent improvements in post-training efficiency and generation quality.

2 Related Work

Reward-Based Post-Training of Flow Models. To align pretrained flow-matching models with human preferences, reward-based post-training has been widely explored. Methods such as DDPO (Black et al., 2024), DPOK (Fan et al., 2023), Flow-GRPO (Liu et al., 2025) and Adjoint Matching (Domingo-Enrich et al., 2025) optimize reward objectives using sampled generation trajectories. More recent endpoint-based methods, including Diffusion-NFT (Zheng et al., 2026), AWM (Xue et al., 2026), and RAM (Bergmeister et al., 2026), instead reuse rewarded endpoints through analytic noising and optimize pretraining-like regression objectives, substantially improving training flexibility and efficiency. Building on this endpoint-based paradigm, we investigate the state exposure induced by analytically reconstructed training states.

Exposure Bias in Flow Models. Flow models are trained on analytically constructed states but generate samples by integrating learned generative dynamics, creating a potential training–sampling discrepancy. Existing methods address this problem either during training through input perturbation, condition corruption, or scheduled sampling (Ning et al., 2023; Li et al., 2025a; Yu et al., 2026; Ren et al., 2024), or during inference by correcting the sampling trajectory (Ning et al., 2024). However, these methods are developed for standard diffusion and flow training and do not account for the additional requirement in RL post-training to improve reward while constraining deviation from a pretrained reference model. Directly transferring such corrections may disturb this balance, particularly because their effects can vary across timesteps and trajectories. We therefore investigate how exposure bias manifests in flow-model RL post-training methods based on analytically reconstructed states and, guided by our empirical findings, develop DAEC to mitigate the resulting state exposure mismatch.

3 Observations

3.1 Preliminaries: Reinforce Adjoint Matching

We follow the formulation of Reinforce Adjoint Matching (RAM) (Bergmeister et al., 2026). Let X_0 denote a clean sample and $\epsilon \sim \mathcal{N}(0, I)$ Gaussian noise. Flow matching constructs an intermediate

state at $t \in [0, 1]$ as

$$X_t = (1 - t)X_0 + t\epsilon, \tag{1}$$

where $t = 0$ corresponds to clean data and $t = 1$ to noise. The velocity field satisfies

$$v_t(x) = \mathbb{E}[\epsilon - X_0 \mid X_t = x], \tag{2}$$

and a model v_t^θ is trained by regressing against the sample-wise target $\epsilon - X_0$.

RAM extends this pretraining-like regression structure to RL post-training. It first generates an on-policy endpoint $X_0 \sim p_0^\theta$, evaluates its reward $r(X_0)$, and then independently reconstructs training states through analytic forward noising,

$$X_t^{\text{ana}} = (1 - t)X_0 + t\epsilon. \tag{3}$$

We follow the RAM implementation with group-relative reward normalization. The implementation maintains an exponential-moving-average (EMA) copy of the trainable policy, which is used in the reward-correction term. Denoting this lagged policy by $v_t^{\bar{\theta}}$, we write the regression target as

$$\widehat{v}_t^{\text{RAM}} = v_t^{\text{ref}}(X_t^{\text{ana}}) + \lambda A(X_0)\left[(\epsilon - X_0) - v_t^{\bar{\theta}}(X_t^{\text{ana}})\right], \tag{4}$$

where $A(X_0)$ denotes the normalized endpoint advantage and λ is the reward scale. Stop-gradient is applied to $\widehat{v}_t^{\text{RAM}}$ during optimization. This reconstruction paradigm avoids training on costly stochastic rollouts (Liu et al., 2025) and allows multiple independent training states to be reused from each rewarded endpoint. Importantly, while these analytic states match the controlled-process joint distribution at initialization and at the optimum, RAM uses them as a plug-in approximation during intermediate optimization. This motivates us to examine what happens between these two regimes. We next analyze the rollout-state discrepancy observed under the post-trained model through three complementary observations. Appendix E provides further details of the observational analysis.

3.2 Observation I: Rollout Deviation Is Structured over Time

Although RAM trains on analytically reconstructed states, generation follows the model-induced probability-flow ODE (Song et al., 2021). Let X_t^{roll} denote a state encountered during generation from the initial noise $\epsilon_0 = X_1$. Given the model prediction $v_t^\theta(X_t^{\text{roll}})$, we can recover its instantaneous estimates of the clean and noise endpoints as

$$\widehat{X}_0^{\,t} = X_t^{\text{roll}} - t\, v_t^\theta(X_t^{\text{roll}}), \qquad \widehat{\epsilon}^{\,t} = X_t^{\text{roll}} + (1 - t)\, v_t^\theta(X_t^{\text{roll}}). \tag{5}$$

By construction,

$$X_t^{\text{roll}} = (1 - t)\widehat{X}_0^{\,t} + t\widehat{\epsilon}^{\,t}. \tag{6}$$

Further derivations and discussions of the instantaneous endpoint estimates are provided in Appendix D. We quantify the rollout deviation at time t as

$$D(t) = \mathbb{E}\left[\frac{\|t\,(\epsilon_0 - \widehat{\epsilon}^{\,t})\|_2}{\|X_t^{\text{roll}}\|_2}\right]. \tag{7}$$

The factor t converts the noise-endpoint discrepancy into the state-space displacement induced by restoring the trajectory's initial noise ϵ_0. Therefore, $D(t)$ measures the relative magnitude of this trajectory-consistent local correction rather than the actual paired rollout gap or its effect on generation quality. Observations II and III examine its reward effect and trajectory specificity. As shown in Fig. 1(a), $D(t)$ varies systematically over the denoising timestep and reaches its maximum in the intermediate regime. While $D(t)$ vanishes at both endpoints by construction, its magnitude in the interior is determined empirically. We observe substantially larger correction displacements at intermediate denoising times, indicating that the rollout deviation is concentrated in this regime.

3.3 Observation II: Rollout Deviation Can Be Harmful

Observation I characterizes the rollout deviation geometrically, but does not establish whether correcting it affects generation quality. We therefore intervene at a rollout state along the trajectory-consistent correction direction and measure the resulting change in terminal reward.

At a selected rollout time t, we view the current state through its instantaneous decomposition $X_t^{\mathrm{roll}} = (1-t)\widehat{X}_0^t + t\widehat{\epsilon}^t$. We then keep the model-implied clean component $\widehat{X}_0^t$ fixed and replace the model-implied noise endpoint $\widehat{\epsilon}^t$ with the actual initial noise seed ϵ_0:

$$X_t^{\mathrm{corr}} = (1-t)\widehat{X}_0^t + t\epsilon_0 = X_t^{\mathrm{roll}} + t\left(\epsilon_0 - \widehat{\epsilon}^t\right). \tag{8}$$

We then resume the same denoising process from X_t^{corr} and measure the terminal reward change relative to the paired normal generation, $\Delta R_{\mathrm{corr}}(t) = R_{\mathrm{corr}}(t) - R_{\mathrm{normal}}$. Importantly, this intervention uses only information available at the current rollout state and does not access the final generated endpoint.

Fig. 1 (b) reveals a clearly stage-dependent effect. Correcting the state during the intermediate denoising regime improves AestheticScore. For example, around $t \approx 0.52$, the intervention improves the AestheticScore by approximately $+0.028$. In contrast, applying the same correction near the clean endpoint becomes harmful, with the reward change turning negative. Additional evaluations using DeQAScore and HPSv2 show similar intermediate-stage gains, as reported in Appendix G.1.

Interestingly, the sign of the intervention effect is not determined by the magnitude of deviation alone. Two timesteps with comparable deviation magnitude can produce opposite reward changes. This indicates that the practical consequence of rollout-state discrepancy is *stage-dependent*: correcting intermediate deviations can improve terminal generation quality, whereas the negative effect near the clean endpoint is consistent with disrupting late-stage refinement.

Thus, Observation II serves as diagnostic evidence that the geometric deviation identified in Observation I has stage-dependent consequences for terminal reward. Its signed reward profile does not directly determine the DAEC perturbation design; Observation III further examines whether the beneficial correction depends on trajectory-specific directions.

3.4 Observation III: The Beneficial Correction Is Trajectory-Specific

The improvement observed in Observation II could arise from a generic re-noising effect rather than from correcting a rollout-specific deviation. To distinguish these explanations, we introduce a *fresh-noise* counterfactual. Instead of restoring the trajectory's original noise ϵ_0, we replace the instantaneous noise estimate $\widehat{\epsilon}^t$ with an independent sample $\epsilon' \sim \mathcal{N}(0, I)$:

$$X_t^{\mathrm{fresh}} = (1-t)\widehat{X}_0^t + t\epsilon'. \tag{9}$$

This intervention has the same form as equation 8, differing only in whether the injected noise is consistent with the current generation trajectory. We further include a norm-matched random-direction control, whose perturbation magnitude matches $\|X_t^{\mathrm{corr}} - X_t^{\mathrm{roll}}\|_2$ while removing the trajectory-specific direction.

For each intervention $a \in \{\mathrm{corr}, \mathrm{fresh}, \mathrm{randdir}\}$, we define the terminal reward change relative to normal generation as $\Delta R_a(t) = R_a(t) - R_{\mathrm{normal}}$. We then measure the trajectory-specific advantage of restoring the original seed over fresh re-noising as $\Delta R_{\mathrm{spec}}(t) = \Delta R_{\mathrm{corr}}(t) - \Delta R_{\mathrm{fresh}}(t) = R_{\mathrm{corr}}(t) - R_{\mathrm{fresh}}(t)$. As shown in Fig. 1 (c), the original-noise correction outperforms fresh re-noising at several intermediate times. Moreover, fresh re-noising alone is typically neutral or detrimental.

These results indicate that the intermediate-stage improvement is not explained by arbitrary perturbation or generic re-noising. The useful correction is strongly tied to the noise seed that generated the current trajectory, and its advantage over both fresh re-noising and norm-matched random perturbations weakens a purely isotropic explanation. Taken together, the observed discrepancy is structured and trajectory-dependent under the post-trained model.

4 Methods

The observations above suggest that the state discrepancy arising from analytically reconstructed training states is neither uniform nor arbitrary. First, rollout deviation exhibits a clear temporal structure. Second, its effect on terminal reward is strongly stage-dependent. Third, the beneficial correction is trajectory-specific, indicating that the relevant perturbation is directional rather than isotropic. These findings motivate us to mitigate this exposure bias by augmenting the analytically

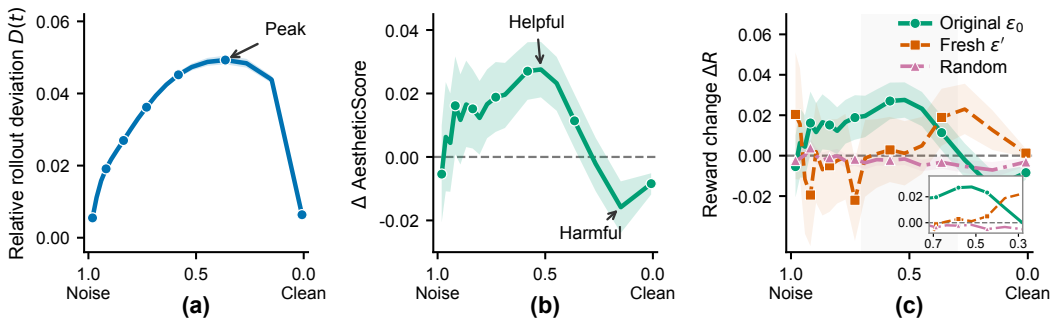

Figure 1: **Rollout-state discrepancy in RAM.** (a) Relative rollout deviation across denoising timesteps. (b) Terminal reward change after trajectory-consistent correction. (c) Comparison with fresh-noise and norm-matched random controls. All panels use 64 prompts and five generation seeds, yielding 320 trajectories per timestep. Lines show means, and shaded regions denote 95% confidence intervals obtained from 20,000 prompt-level cluster bootstrap resamples.

reconstructed training states with nearby, trajectory-relevant perturbations while preserving the underlying regression objective. Specifically, instead of training the current model only at X_t^{ana}, we construct a nearby perturbed state

$$\widetilde{X}_t = X_t^{\mathrm{ana}} + \eta_t \widehat{u}_t, \tag{10}$$

where $\widehat{u}_t$ specifies a residual-defined local exposure direction and η_t controls its time- and sample-dependent magnitude. Unlike generic input perturbation methods that inject random noise to broaden the training distribution (Ning et al., 2023; Chen et al., 2024), our perturbation is explicitly structured by the observations above: its direction is tied to a local proxy for the trajectory-relevant discrepancy. At the same time, its magnitude adapts to the denoising stage and the individual sample. We refer to this strategy as *Drift-Aligned Exposure Correction (DAEC)*. Rather than reconstructing an exact rollout state, DAEC exposes the model to a residual-defined directional neighborhood around each analytic training state.

4.1 Drift-Aligned Exposure Direction

A central question is how to determine the perturbation direction $\widehat{u}_t$. Observation III suggests that the useful correction is not captured by isotropic perturbations, but instead depends on a trajectory-relevant direction. We therefore exploit a quantity that is already available in the RAM update. For an analytically constructed state X_t^{ana}, we define the *bridge residual*

$$u_t = v_t^{\bar{\theta}}(X_t^{\mathrm{ana}}) - (\epsilon - X_0), \tag{11}$$

where $v_t^{\bar{\theta}}$ denotes the lagged policy used to construct the regression target. The term $\epsilon - X_0$ is the sample-wise velocity associated with the analytic forward pair, while $v_t^{\bar{\theta}}(X_t^{\mathrm{ana}})$ is the velocity predicted by the lagged policy at the same state. We use the lagged-policy prediction because it is already evaluated at X_t^{ana} by the underlying RAM update; evaluating the current policy at the same anchor would require an additional model forward once the trainable branch is moved to $\widetilde{X}_t$. Their residual therefore captures a local directional discrepancy between the model prediction and the analytic training pair. This residual also admits a direct interpretation in the same clean/noise decomposition used in Observation I. At the analytic state, define the lagged-policy-implied noise component as

$$\widehat{\epsilon}_{\mathrm{ana}}^{t} = X_t^{\mathrm{ana}} + (1-t)v_t^{\bar{\theta}}(X_t^{\mathrm{ana}}). \tag{12}$$

Using $X_t^{\mathrm{ana}} = (1-t)X_0 + t\epsilon$, we obtain the exact identity

$$\widehat{\epsilon}_{\mathrm{ana}}^{t} - \epsilon = (1-t)\left[v_t^{\bar{\theta}}(X_t^{\mathrm{ana}}) - (\epsilon - X_0)\right] = (1-t)u_t. \tag{13}$$

Hence, u_t points along the discrepancy between the noise component implied by the lagged policy and the Gaussian noise assigned to the analytic pair. Observation II moves a rollout state against its locally inferred noise discrepancy. In contrast, DAEC perturbs an analytic anchor along $+u_t$, expanding training exposure on the outward side of this correction-relevant local axis.

Since we use u_t only to specify the perturbation direction and control its magnitude separately, we normalize the residual as $\widehat{u}_t = \frac{u_t}{\max(\|u_t\|_2, \varepsilon_{\text{num}})}$. ε_{num} is a small numerical constant. The same residual also admits a useful connection to the bridge quantity underlying RAM (Bergmeister et al., 2026). For the linear forward kernel, the Bayes bridge identity gives

$$\nabla_{X_t} \log p_{0|t}(X_0 \mid X_t) = \frac{1-t}{t} \left[v_t(X_t) - (\epsilon - X_0) \right]. \tag{14}$$

Since $(1-t)/t > 0$ for $t \in (0,1)$, the residual inside the brackets determines the bridge-score direction. Hence, when $v_t^{\bar{\theta}}$ approximates the corresponding velocity field, u_t is aligned with this direction. Importantly, u_t is not the exact displacement from X_t^{ana} to an actual rollout state. Rather, it defines a local axis associated with the trajectory-consistent correction geometry. A direct comparison between the bridge residual and the paired rollout gap is presented in Appendix G.2. Moreover, both terms in equation 11 are already computed by the underlying RAM update, so obtaining the perturbation direction $\widehat{u}_t$ requires no additional model evaluation.

4.2 Time- and Sample-Adaptive Exposure Magnitude

Observation I further shows that the magnitude of rollout deviation is strongly time-dependent. We therefore decouple the perturbation direction from its magnitude and adapt the latter at both the timestep and sample levels. For sample i, we parameterize the perturbation magnitude as

$$\eta_i = \eta_0 w(e) g(t_i) \|X_{t_i}^{\text{ana}}\|_2 q_i, \tag{15}$$

where η_0 is a global scale, $w(e)$ is a training warmup, $g(t)$ captures the temporal profile of rollout-state deviation, and q_i adapts the perturbation to the relative residual magnitude of each sample.

Time-dependent profile. Motivated by the structured temporal behavior in Observation I, we estimate a fixed temporal profile $g(t)$ through a one-time offline geometric analysis using only latent state discrepancies. For calibration trajectory n, we define the straight analytic path connecting its initial noise $\epsilon_{0,n}$ and generated endpoint $X_{0,n}^{\text{roll}}$ as $X_{t,n}^{\text{ana}} = (1-t)X_{0,n}^{\text{roll}} + t\epsilon_{0,n}$. At each measured calibration timestep τ_k, we compute

$$g(\tau_k) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \frac{\left\| X_{\tau_k,n}^{\text{roll}} - X_{\tau_k,n}^{\text{ana}} \right\|_2^2}{\left\| X_{\tau_k,n}^{\text{ana}} \right\|_2^2}}. \tag{16}$$

We set $g(0) = g(1) = 0$ and extend the discrete profile to continuous t using linear interpolation. Unlike $D(t)$ in Observation I, which measures the trajectory-consistent correction derived from instantaneous endpoint estimates, $g(t)$ is constructed directly from paired rollout-to-analytic gaps. Both quantities concentrate at intermediate timesteps. The profile is neither learned nor manually fitted. Appendix F provides additional details.

We additionally scale the displacement by the state norm $\|X_t^{\text{ana}}\|_2$, so that $\eta_0 w(e) g(t) q_i$ controls the perturbation magnitude relative to the scale of the current state. To avoid introducing large perturbations at the beginning of post-training, we use a linear warmup $w(e) = \min\left(\frac{e}{E_{\text{warm}}}, 1\right)$, where e denotes the current training epoch and E_{warm} is the number of warmup epochs.

Sample-adaptive residual gating. The residual norm $\rho_i = \|u_{t_i}\|_2$ can itself depend strongly on the timestep. Directly using ρ_i as a global gate would therefore mix sample-specific variation with the temporal trend already captured by $g(t)$. We instead compare each residual against other samples at nearby timesteps. Specifically, we define

$$\kappa_{ij} = \exp\left(-\frac{(t_i - t_j)^2}{2h^2} \right), \qquad \bar{\rho}_i = \frac{\sum_j \kappa_{ij} \rho_j}{\sum_j \kappa_{ij}}, \tag{17}$$

where h controls the temporal neighborhood. The sample-specific gate is then

$$r_i = \text{clip}\left(\frac{\rho_i}{\max(\bar{\rho}_i, \varepsilon_{\text{num}})}, 0, c \right), \qquad q_i = \frac{r_i}{\max\left(\frac{1}{B} \sum_{j=1}^{B} r_j, \varepsilon_{\text{num}} \right)}. \tag{18}$$

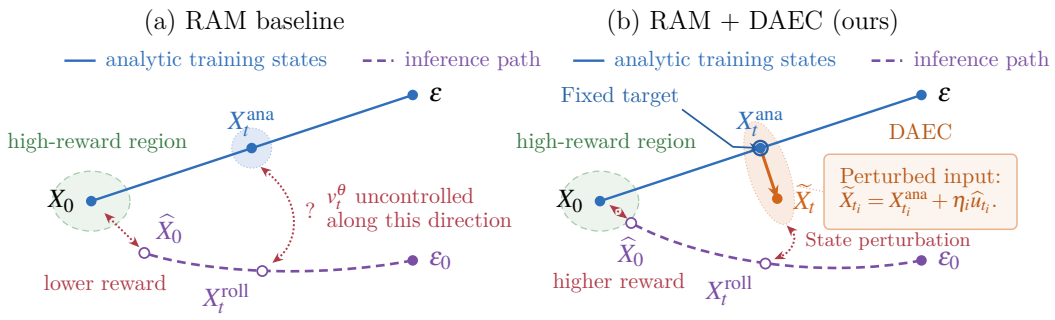

Figure 2: **Drift-Aligned Exposure Correction (DAEC). (a)** RAM supervises the model only at the analytic state X_t^{ana}, leaving it exposed to inference-time drift. **(b)** DAEC perturbs only the model input as $\widetilde{X}_{t_i} = X_{t_i}^{\text{ana}} + \eta_i \widehat{u}_{t_i}$, while keeping the target fixed at $X_{t_i}^{\text{ana}}$.

where B is the mini-batch size and c is a clipping threshold. Thus, q_i measures how large a sample's residual is relative to other samples at comparable denoising times, while its unit-mean normalization redistributes the perturbation magnitude across samples and keeps the average gate value at one. Overall, the final perturbed state is $\widetilde{X}_{t_i} = X_{t_i}^{\text{ana}} + \eta_i \widehat{u}_{t_i}$.

4.3 Anchored Regression Objective

As illustrated in Fig. 2, DAEC modifies only the input to the trainable model. The reference and lagged-policy predictions remain evaluated at the original analytic state X_t^{ana}, and the reward-corrected regression target is constructed exactly at this anchor. Using the same practical RAM target as in equation 4, we form

$$\widehat{v}_t^{\text{RAM}} = v_t^{\text{ref}}(X_t^{\text{ana}}) + \lambda A(X_0)\left[(\epsilon - X_0) - v_t^{\bar{\theta}}(X_t^{\text{ana}})\right]. \tag{19}$$

DAEC then optimizes

$$\mathcal{L}_{\text{DAEC}} = \mathbb{E}\left[\left\| v_t^{\theta}(\widetilde{X}_t) - \text{sg}\big(\widehat{v}_t^{\text{RAM}}\big)\right\|_2^2\right]. \tag{20}$$

When $\eta_t = 0$, equation 20 reduces to the original RAM regression objective. Thus, DAEC differs from RAM only in the input to the trainable model, replacing X_t^{ana} with $\widetilde{X}_t$. This asymmetry is deliberate. We do not assume that $\widetilde{X}_t$ is an exact rollout state or construct a new regression target for it. Instead, the original RAM target remains anchored at X_t^{ana} and supervises the trainable model at the nearby, directionally perturbed state $\widetilde{X}_t$. DAEC can therefore be interpreted as a form of directional vicinal regression around analytic training states. For a small perturbation $\delta_t = \eta_t \widehat{u}_t$,

$$v_t^{\theta}(X_t^{\text{ana}} + \delta_t) \approx v_t^{\theta}(X_t^{\text{ana}}) + J_t^{\theta}(X_t^{\text{ana}})\delta_t, \tag{21}$$

where $J_t^{\theta}(X_t^{\text{ana}})$ denotes the input Jacobian of the trainable velocity field at X_t^{ana}. Near a fitted training point, equation 20 additionally discourages excessive sensitivity of the learned velocity field along the selected drift-aligned direction. Finally, DAEC preserves RAM's computational structure. The reference prediction, lagged-policy prediction, and trainable prediction are the same three model evaluations already present in RAM. DAEC only changes the trainable branch input and adds lightweight tensor operations for equation 11–equation 18. It therefore requires no additional model evaluations per training update.

5 Experiments

5.1 Experimental Setup

We integrate DAEC into the analytic-state regression framework of RAM (Bergmeister et al., 2026) and evaluate it by post-training Stable Diffusion 3.5 Medium (SD3.5M) (Esser et al., 2024) under three reward settings: compositional generation with GenEval (Ghosh et al., 2023), visual text rendering with the OCR reward following Flow-GRPO (Liu et al., 2025), and human-preference alignment with PickScore (Kirstain et al., 2023). More details are provided in Appendix C.

Table 1: GenEval results on Stable Diffusion 3.5 Medium. Values in parentheses provide additional precision for close comparisons, and the better result within each comparison is shown in **bold**.

Model	# Steps	GenEval ↑	Image Quality Metrics			
			Aesthetic ↑	DeQA ↑	ImgRwd ↑	HPSv2 ↑
SD3.5M	0	0.64	5.41	4.08	0.82	0.28
Published Baselines						
Flow-GRPO	>5k	**0.95**	**5.25**	4.01	**1.03**	**0.27**
AWM	300	0.83	5.14	3.75	0.67	0.24
Diffusion-NFT	900	**0.95**	4.98	**4.10**	0.30	0.24
Matched-Budget and Best Results						
RAM	190	0.92 (0.9241)	5.19	3.78	0.98	0.27 (0.2679)
RAM + DAEC	190	**0.94** (0.9376)	**5.33**	**4.03**	**1.01**	**0.27** (0.2734)
RAM + DAEC (best)	230	**0.95** (0.9462)	**5.43**	**3.93**	**1.06**	**0.27** (0.2746)

Table 2: Ablation of the key designs in DAEC on GenEval using Stable Diffusion 3.5 Medium. The best and second-best results are shown in **bold** and underline.

Method	Configuration			Best Results			
	Direction	Time Profile	Adaptive Gating	GenEval ↑	Strict Composition		
					Counting ↑	Position ↑	Color Attr. ↑
RAM	–	–	–	0.92	**0.98**	0.92	0.73
RAM + DAEC (random dir.)	Random	$g(t)$	Time-aware	0.90	0.92	0.89	0.67
RAM + DAEC (uniform profile)	Drift	Uniform	Time-aware	0.94 (0.9354)	0.97 (0.9656)	0.94 (0.9400)	0.76
RAM + DAEC (global gating)	Drift	$g(t)$	Global	0.94 (0.9435)	0.95	0.94 (0.9425)	0.78
RAM + DAEC (full)	**Drift**	$\boldsymbol{g(t)}$	**Time-aware**	**0.95** (0.9462)	0.97 (0.9656)	**0.96**	**0.81**

5.2 Main Results

Multi-Metric Results. As shown in Table 1, at the matched budget of 190 training steps, DAEC improves the GenEval score from 0.9241 to 0.9376 while improving all image quality metrics. At 230 training steps, DAEC further reaches a GenEval score of 0.9462 and retains improvements over RAM across all image quality metrics. For reference, Diffusion-NFT and Flow-GRPO achieve GenEval scores of approximately 0.95 after 900 and more than 5,000 training steps, respectively. The gains across compositional correctness, image quality, and preference metrics indicate that the reward improvement does not come at the expense of the measured generation quality. The qualitative comparisons in Figs. 7 and 8 further illustrate the improvements in semantic fidelity and visual quality. Additional results in Table 3 show that DAEC maintains gains across different inference samplers and configurations. Multi-seed results in Appendix G.5 provide further evidence of the consistency of these improvements.

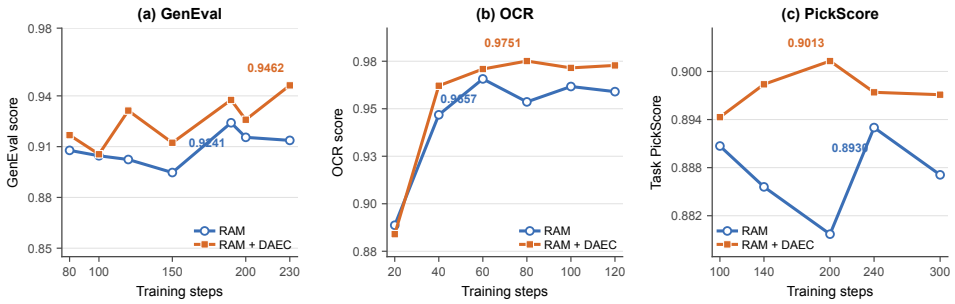

Figure 3: Training dynamics of RAM and RAM + DAEC across GenEval, OCR, and PickScore.

Training Progress. Fig. 3 compares RAM and DAEC at matched training steps across three reward objectives. On GenEval, DAEC achieves higher scores over the compared training interval. On OCR, it exceeds RAM's best evaluated score by step 60. On PickScore, DAEC consistently outperforms RAM at matched checkpoints from step 100 onward. Together, these results demonstrate improved performance at comparable training budgets across the three reward objectives.

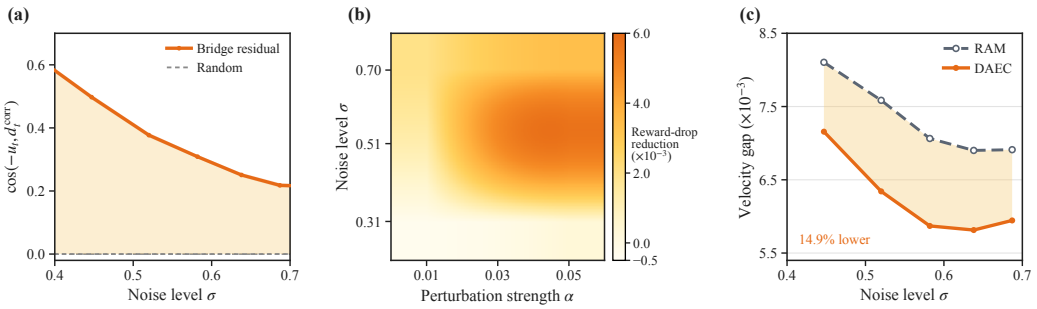

Figure 4: **Directional geometry and robustness to inference exposure bias. (a)** In the targeted regime $\sigma \in [0.4, 0.7]$, the trajectory-consistent pullback remains aligned with $-u_t$ and clearly separated from the random control, identifying the bridge residual as a correction-relevant local axis. **(b)** Compared with RAM, DAEC incurs a smaller reward drop under drift-aligned latent perturbations, with the largest advantage in the intermediate-noise regime and at stronger perturbation strengths. **(c)** DAEC exhibits a lower trajectory-conditioned velocity gap in the same regime.

5.3 Analysis and Discussion

Ablation study. Table 2 evaluates the key design choices of DAEC. Replacing the drift-aligned direction with a random direction degrades performance below RAM, supporting the importance of the selected exposure direction. Using a uniform time profile or replacing time-aware gating with global gating also reduces the GenEval score, supporting the temporal and sample-adaptive magnitude design. Overall, the full method achieves the highest GenEval score and the strongest strict performance on spatial relations and color-attribute binding, while counting remains slightly below RAM.

Robustness to inference exposure bias. Guided by Observations in Sec. 3, we examine whether DAEC improves robustness to rollout-induced deviations in the reward-relevant intermediate denoising regime identified in Fig. 1. We first validate the geometry of the perturbation direction. As shown in Fig. 4(a), the trajectory-consistent pullback remains aligned with $-u_t$ and clearly separated from the random control, supporting the bridge residual as a correction-relevant local axis. We then introduce controlled perturbations along u_t, the direction used for DAEC training exposure. As shown in Fig. 4(b), DAEC incurs a smaller reward drop than RAM, with the largest advantage at intermediate noise levels and stronger perturbations. Finally, we directly measure the velocity gap along each model's inference trajectory. Fig. 4(c) shows that DAEC reduces the trajectory-conditioned velocity gap by 14.9% on average over the intermediate-noise regime $\sigma \in [0.4, 0.7]$. Therefore, these results provide complementary evidence that DAEC mitigates inference exposure bias under both controlled perturbations and actual rollout deviations, demonstrating the effectiveness of its drift-aligned exposure design.

6 Conclusion and Future Work

In this work, we investigate the state exposure mismatch between analytically reconstructed training states and inference-time rollout states in RL post-training of flow models. Through controlled analyses, we show that the discrepancy between analytically reconstructed training states and rollout states varies across timesteps, has stage-dependent effects on reward, and depends on the generation trajectory. Guided by these observations, we propose DAEC, which perturbs the trainable model's input exposure along a drift-aligned direction with timestep- and sample-adaptive magnitude, while keeping the reward-corrected regression target anchored at the original analytic state. DAEC should therefore be viewed as directional exposure augmentation rather than rollout-state reconstruction. Experiments across GenEval, OCR, and PickScore demonstrate consistent improvements in post-training efficiency and generation quality without additional model evaluations. Future work will investigate DAEC on additional model architectures and RL post-training methods that rely on analytically reconstructed training states.

REFERENCES

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling rl post-training of diffusion and flow-matching models, 2026. URL https://arxiv.org/abs/2605.10759.

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

Hao Chen, Yujin Han, Diganta Misra, Xiang Li, Kai Hu, Difan Zou, Masashi Sugiyama, Jindong Wang, and Bhiksha Raj. Slight corruption in pre-training data makes better diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VFpXYBqMSU.

Jaemoo Choi, Wei Guo, Yuchen Zhu, Arash Vahdat, Molei Tao, Julius Berner, and Yongxin Chen. Scaling reinforcement learning for diffusion models via velocity matching. *arXiv preprint arXiv:2608.23664*, 2026a. URL https://arxiv.org/abs/2608.23664.

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*, 2026b. URL https://arxiv.org/abs/2602.04663.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

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

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 *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12606–12633. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/esser24a.html.

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

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a3bf71c7c63f0c3bcb7ff67c67b1e7b1-Abstract-Datasets_and_Benchmarks.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

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

Jisung Hwang and Minhyuk Sung. Gradient preconditioning for efficient and reliable reward-guided generation. *arXiv preprint arXiv:2602.08646*, 2026. URL https://arxiv.org/abs/2602.08646.

Jisung Hwang, Yunhong Min, Jaihoon Kim, I-Chao Shen, and Minhyuk Sung. NoiseTilt: Noise-tilted reverse kernels for diffusion reward alignment. *arXiv preprint arXiv:2606.18066*, 2026. URL https://arxiv.org/abs/2606.18066.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577, 2022. URL https://arxiv.org/abs/2206.00364.

Jaihoon Kim, Taehoon Yoon, Jisung Hwang, and Minhyuk Sung. Inference-time scaling for flow models via stochastic generation and rollover budget forcing. *arXiv preprint arXiv:2503.19385*, 2025. URL https://arxiv.org/abs/2503.19385.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.01569.

Hui Li, Jiayue Lyu, Fu-yun Wang, Kaihui Cheng, Siyu Zhu, and Jingdong Wang. MixFlow training: Alleviating exposure bias with slowed interpolation mixture. *arXiv preprint arXiv:2512.19311*, 2025a. URL https://arxiv.org/abs/2512.19311.

Junzhe Li, Yutao Cui, Tao Huang, Weijie Kong, Yiming Cheng, Chuxuan Zeng, Yinping Ma, Chun Fan, Miles Yang, Zhao Zhong, and Liefeng Bo. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE–SDE. *arXiv preprint arXiv:2507.21802*, 2025b. URL https://arxiv.org/abs/2507.21802.

Mingxiao Li, Tingyu Qu, Ruicong Yao, Wei Sun, and Marie-Francine Moens. Alleviating exposure bias in diffusion models through sampling with shifted time steps. *International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2305.15583.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024b. URL https://arxiv.org/abs/2408.08252.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oCBKGw5HNf.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. URL https://arxiv.org/abs/2211.01095.

Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models. In *International Conference on Machine Learning*, pp. 26245–26265. PMLR, 2023. URL https://arxiv.org/abs/2301.11706.

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xEJMoj1SpX.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.18290.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks, 2016. URL https://arxiv.org/abs/1511.06732.

Zhiyao Ren, Yibing Zhan, Liang Ding, Gaoang Wang, Chaoyue Wang, Zhongyi Fan, and Dacheng Tao. Multi-step denoising scheduled sampling: Towards alleviating exposure bias for diffusion models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 4667–4675. AAAI Press, 2024. URL https://doi.org/10.1609/aaai.v38i5.28267.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

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

Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. *arXiv preprint arXiv:2405.18881*, 2024. URL https://arxiv.org/abs/2405.18881.

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

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

Yixian Xu, Yuanrui Zhang, Shengjie Luo, Liwei Wang, and Di He. Designing reinforcement learning for diffusion models: A unified path-space view. *arXiv preprint arXiv:2608.14430*, 2026. URL https://arxiv.org/abs/2608.14430.

Shuchen Xue, Chongjian GE, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning RL with pretraining in diffusion models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nLY2pOYBrJ.

Meng Yu, Lei Sun, Jianhao Zeng, Xiangxiang Chu, and Kun Zhan. Elucidating the SNR-t bias of diffusion probabilistic models. *arXiv preprint arXiv:2604.16044*, 2026. URL https://arxiv.org/abs/2604.16044.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49842–49869, 2023. URL https://arxiv.org/abs/2302.04867.

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

A Appendix Overview

The appendix is organized as follows. Section B extends the discussion of related work and positions DAEC relative to prior methods. Section C summarizes the experimental settings. Section D derives and interprets the instantaneous endpoint estimates. Section E details the observational analysis. Section F describes the implementation of DAEC. Section G presents additional experimental results. Section H contains the AI use statement, and Section I presents the PyTorch-style pseudocode.

B Additional Related Work and Discussion

Sample weighting and credit assignment. Recent reward-based post-training methods for diffusion and flow models borrow from and extend LLM post-training approaches such as PPO (Schulman et al., 2017), DPO (Rafailov et al., 2023), and GRPO (DeepSeek-AI, 2025; Shao et al., 2024), adapting their policy optimization and relative reward weighting mechanisms to continuous denoising processes. Trajectory-based methods such as Flow-GRPO (Liu et al., 2025) use group-relative rewards to weight clipped importance-ratio objectives across stochastic denoising transitions. GRPO-Guard (Wang et al., 2025) improves this formulation through ratio normalization and timestep-dependent gradient reweighting, preventing particular noise levels or overconfident updates from dominating optimization. Beyond these loss-level modifications, recent analysis separates the design space into policy-gradient objectives, likelihood estimators, and rollout samplers, highlighting the importance of ELBO-based endpoint likelihood estimation (Choi et al., 2026b). By contrast, endpoint-based methods incorporate rewards directly into pretraining-style regression objectives. AWM (Xue et al., 2026) weights flow-matching losses by sample advantages, Diffusion-NFT (Zheng et al., 2026) uses reward-dependent positive and negative velocity targets, and RAM (Bergmeister et al., 2026) applies a reward-scaled correction to a reference-anchored target. RVM (Choi et al., 2026a) provides a unifying interpretation in which signed reward weights act on endpoint-defined velocity losses and an optional anchor controls policy drift. More recently, a complementary path-space analysis derives trajectory- and endpoint-based estimators from a common regularized RL objective and organizes their differences through value estimation, reward weighting, and sampling choices (Xu et al., 2026).

Sampling and inference efficiency. Several methods improve reward optimization by modifying the sampling process during either training or inference. During RL post-training, Flow-GRPO (Liu et al., 2025) converts the deterministic flow ODE into a marginally equivalent SDE to enable stochastic exploration and reduces the number of training denoising steps. MixGRPO (Li et al., 2025b) further restricts SDE sampling and policy updates to a sliding timestep window, while using efficient ODE solvers outside the window. At inference time, DNO (Tang et al., 2024) directly optimizes the sampling noise, whereas Gradient Preconditioning constrains reward gradients to preserve Gaussian noise statistics (Hwang & Sung, 2026). NoiseTilt (Hwang et al., 2026) instead injects whitened reward gradients through the stochastic noise term while keeping the pretrained reverse mean unchanged. Derivative-free approaches such as SVDD (Li et al., 2024b) select high-value intermediate particles; for deterministic flow models, SDE and interpolant conversion enable such particle search, while Rollover Budget Forcing (Kim et al., 2025) adaptively allocates function evaluations across timesteps. These methods improve exploration, timestep allocation, or inference search, whereas DAEC addresses the state discrepancy arising from analytically reconstructed post-training states.

Relation to DAEC. The methods above primarily improve *which samples receive larger updates*, *which timesteps are optimized*, or *how trajectories are explored and selected*. DAEC instead focuses on the state distribution presented to the learner. In endpoint-based post-training, rewarded endpoints are analytically re-noised to construct training states, while inference follows the learned generative dynamics. This can expose the model to a state distribution that differs from the one encountered along its own sampling trajectories. DAEC is designed to mitigate this discrepancy while retaining the RL-specific balance between reward improvement and proximity to the pretrained reference. It is therefore complementary to reward reweighting and sampler optimization, rather than another advantage estimator or inference-time guidance rule.

C Experimental Details

We follow the RAM (Bergmeister et al., 2026) implementation and post-train Stable Diffusion 3.5 Medium (Esser et al., 2024) with LoRA (Hu et al., 2022) (rank 32, $\alpha = 64$), keeping the base model frozen. We train at 512×512 resolution with 20 denoising steps and a guidance scale of 2.0. Following RAM, each epoch contains 48 prompts, 24 on-policy samples per prompt, and eight analytic training targets per sample. We use group-relative advantage centering with epoch-level normalization and maintain the lagged policy using an EMA decay of 0.9.

Evaluation uses 40 denoising steps at 512×512 resolution with CFG guidance. For DAEC, we set the correction scale to 0.4, use a 30-epoch warmup, clip the adaptive gate at 3.0 before normalization, and set the temporal kernel bandwidth to 0.1. The temporal profile and all DAEC-specific parameters are shared across the three reward objectives without task-specific tuning. Within each reward setting, RAM and DAEC use the same training and evaluation configuration and differ only in the state presented to the trainable model. All experiments are conducted in FP16 on 4 NVIDIA H20 GPUs.

D Derivation and Interpretation of the Instantaneous Endpoint Estimates

Here, we provide additional details on the instantaneous clean and noise estimates used in equation 5. For the linear flow-matching path,

$$X_t = (1-t)X_0 + t\epsilon, \tag{22}$$

The corresponding sample-wise velocity is

$$v = \epsilon - X_0. \tag{23}$$

Given a state x and a velocity prediction v, we define $\widehat{X}_0^{\,t}$ and $\widehat{\epsilon}^{\,t}$ as the pair satisfying

$$x = (1-t)\widehat{X}_0^{\,t} + t\widehat{\epsilon}^{\,t}, \qquad v = \widehat{\epsilon}^{\,t} - \widehat{X}_0^{\,t}. \tag{24}$$

Solving this linear system gives

$$\widehat{X}_0^{\,t} = x - tv, \qquad \widehat{\epsilon}^{\,t} = x + (1-t)v. \tag{25}$$

Applying equation 25 to $x = X_t^{\mathrm{roll}}$ and $v = v_t^\theta(X_t^{\mathrm{roll}})$ yields equation 5. Notably, no division by $1-t$ is required because the model is parameterized in velocity form. Under the sample-wise linear decomposition, the corresponding velocity is $v = \epsilon - X_0$; the learned model estimates the conditional velocity field rather than a particular sample-wise target. If an estimate of ϵ were given instead, solving the interpolation equation for X_0 would yield

$$X_0 = \frac{X_t - t\epsilon}{1-t}. \tag{26}$$

We emphasize that $\widehat{X}_0^{\,t}$ and $\widehat{\epsilon}^{\,t}$ are *model-implied instantaneous endpoint estimates*. They are obtained by expressing the current state and velocity under the local linear flow-matching parameterization, and should not be interpreted as the actual endpoints obtained by integrating the probability-flow ODE. In general, the velocity field varies along the rollout trajectory, so the true generated endpoint requires integrating

$$\mathrm{d}X_t = v_t^\theta(X_t)\,\mathrm{d}t. \tag{27}$$

Thus, equation 5 provides a local decomposition of the current rollout state, rather than an assumption that the rollout trajectory itself follows a sample-wise straight line.

The deviation measure in Observation I also has a direct state-space interpretation. Combining equation 5 with the counterfactual correction in equation 8 gives

$$X_t^{\mathrm{corr}} - X_t^{\mathrm{roll}} = t\left(\epsilon_0 - \widehat{\epsilon}^{\,t}\right). \tag{28}$$

Hence, the numerator of $D(t)$ in equation 7 is exactly the norm of the state displacement applied by the trajectory-consistent counterfactual intervention.

E Details of the Observational Analysis

Observation setup. We analyze the RAM model using 64 GenEval prompts and five generation seeds, yielding 320 trajectories per timestep. Images are generated at 512×512 resolution with 20 denoising steps and a guidance scale of 2.0. We evaluate all 19 intermediate timesteps and use AestheticScore for the reward analyses in Observations II and III. The shaded regions show 95% confidence intervals obtained from 20,000 prompt-level cluster bootstrap resamples, with normal and intervened generations paired for Observations II and III.

Observation details. The observations in Secs. 3.2–3.4 are obtained from rollout analyses on a frozen post-trained model. At a selected rollout timestep t, we evaluate the model at the current state X_t^{roll} and compute the instantaneous decomposition $(\widehat{X}_0^t, \widehat{\epsilon}^t)$ using equation 5. For Observation I, we use this decomposition to measure the rollout deviation without modifying the generation trajectory. For Observations II and III, we use it to construct a counterfactual state, replace the current rollout state, and then resume the same denoising procedure. Each intervention is constructed without accessing any future rollout state or the final generated endpoint. We report the resulting terminal reward relative to the paired normal generation.

Besides the original-noise correction X_t^{corr} and the fresh-noise intervention X_t^{fresh} defined in the main text, we include a norm-matched random-direction control. Let

$$d_t^{\mathrm{corr}} = X_t^{\mathrm{corr}} - X_t^{\mathrm{roll}}, \qquad z \sim \mathcal{N}(0, I), \tag{29}$$

and define

$$X_t^{\mathrm{randdir}} = X_t^{\mathrm{roll}} + \|d_t^{\mathrm{corr}}\|_2 \frac{z}{\max(\|z\|_2, \varepsilon_{\mathrm{num}})}. \tag{30}$$

Thus, the random control matches the correction magnitude on a per-sample basis while removing its direction. Together with fresh re-noising, it helps distinguish a trajectory-specific intervention from generic re-noising or arbitrary perturbation effects. For each intervention $a \in \{\mathrm{corr}, \mathrm{fresh}, \mathrm{randdir}\}$, we report the paired terminal reward change $\Delta R_a(t) = R_a(t) - R_{\mathrm{normal}}$.

F Implementation Details of DAEC

This section presents the implementation details of DAEC. We first generate and score groups of on-policy endpoints, and reuse each endpoint to construct multiple analytic training states from independently sampled timestep–noise pairs. The normalized advantage A_i supplies the endpoint-level reward signal in equation 4, while DAEC applies the exposure correction only to the input of the trainable model. Algorithm 1 summarizes the complete batched procedure.

Batched state construction. For each endpoint X_0^i, we sample K independent timestep–noise pairs (t_{ik}, ϵ_{ik}) and construct

$$X_{t_{ik}}^{\mathrm{ana}} = (1 - t_{ik}) X_0^i + t_{ik} \epsilon_{ik}. \tag{31}$$

Both positive- and negative-advantage endpoints are retained. The resulting $G \times K$ analytic states are flattened into a single optimization mini-batch. Since the sample-adaptive gate compares residual magnitudes at nearby timesteps, it is computed after the residuals of the entire mini-batch have been obtained.

Temporal-profile calibration. We calibrate $g(t)$ once using a frozen RAM model post-trained under the GenEval reward setting at training step 100 with 32 GenEval prompts. At each calibration timestep τ_k, we compute the relative RMS rollout-to-analytic gap

$$g_k = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \frac{\left\| X_{\tau_k,n}^{\mathrm{roll}} - X_{\tau_k,n}^{\mathrm{ana}} \right\|_2^2}{\left\| X_{\tau_k,n}^{\mathrm{ana}} \right\|_2^2}}. \tag{32}$$

Here, $X_{t,n}^{\mathrm{ana}} = (1 - t) X_{0,n}^{\mathrm{roll}} + t \epsilon_{0,n}$. For $t \in [\tau_k, \tau_{k+1}]$, we define

$$g(t) = (1 - \alpha) g_k + \alpha g_{k+1}, \qquad \alpha = \frac{t - \tau_k}{\tau_{k+1} - \tau_k}, \tag{33}$$

Algorithm 1 DAEC Training Algorithm

Require: Reference model v^{ref}, lagged policy $v^{\bar{\theta}}$, trainable model v^{θ}, reward r, and targets per endpoint K

while not converged **do**
  Generate groups of on-policy endpoints $\{X_0^i\}_{i=1}^G$ {On-policy endpoints}
  Evaluate rewards and compute normalized advantages $\{A_i\}$ {Endpoint credit assignment}
  for each endpoint X_0^i and $k = 1, \ldots, K$ **in parallel do**
    Sample t_{ik} and $\epsilon_{ik} \sim \mathcal{N}(0, I)$ {Independent time–noise pair}
    $X_{t_{ik}}^{\mathrm{ana}} \leftarrow (1 - t_{ik})X_0^i + t_{ik}\epsilon_{ik}, \quad v_{ik}^{\mathrm{ana}} \leftarrow \epsilon_{ik} - X_0^i$ {Analytic state and path velocity}
  end for
  Flatten all (i, k) pairs into the optimization mini-batch {Batched state construction}
  $v_{ik}^{\mathrm{ref}} \leftarrow v_{t_{ik}}^{\mathrm{ref}}(X_{t_{ik}}^{\mathrm{ana}}), \quad v_{ik}^{\bar{\theta}} \leftarrow v_{t_{ik}}^{\bar{\theta}}(X_{t_{ik}}^{\mathrm{ana}})$ {Frozen anchor predictions}
  $u_{ik} \leftarrow v_{ik}^{\bar{\theta}} - v_{ik}^{\mathrm{ana}}, \quad \rho_{ik} \leftarrow \|u_{ik}\|_2$ {Bridge residual and its magnitude}
  $\widehat{u}_{ik} \leftarrow u_{ik} / \max(\rho_{ik}, \varepsilon_{\mathrm{num}})$ {Normalized drift-aligned direction}
  Compute $\{q_{ik}\}$ over the full batch using equation 17–equation 18 {Sample-adaptive gate}
  $\eta_{ik} \leftarrow \eta_0 w(e) g(t_{ik}) \|X_{t_{ik}}^{\mathrm{ana}}\|_2 q_{ik}$ {Adaptive exposure magnitude}
  $\widetilde{X}_{t_{ik}} \leftarrow X_{t_{ik}}^{\mathrm{ana}} + \eta_{ik}\widehat{u}_{ik}$ {Apply adaptive magnitude along the drift direction}
  $\widehat{v}_{ik} \leftarrow v_{ik}^{\mathrm{ref}} + \lambda A_i \left(v_{ik}^{\mathrm{ana}} - v_{ik}^{\bar{\theta}} \right)$ {Anchored regression target}
  Update θ using $\frac{1}{GK} \sum_{i=1}^{G} \sum_{k=1}^{K} \left\| v_{t_{ik}}^{\theta}(\widetilde{X}_{t_{ik}}) - \mathrm{sg}(\widehat{v}_{ik}) \right\|_2^2$ {Trainable-branch update}
  Update $\bar{\theta}$ from θ using the EMA schedule {Lagged-policy update}
end while

with $g(0) = g(1) = 0$. Calibration uses only latent-state discrepancies and no reward values. **The resulting profile is fixed and reused unchanged across different reward objectives, including GenEval, OCR, and PickScore.**

Adaptive magnitude computation. The local temporal kernel in equation 17 estimates the typical residual magnitude around each sampled timestep. equation 18 then constructs a clipped relative gate and normalizes its mini-batch mean to one. The calibrated profile is evaluated at sampled timesteps using the linear interpolation in equation 33, while the warmup factor $w(e)$ is computed from the current training epoch defined by $w(e) = \min\left(\frac{e}{E_{\mathrm{warm}}}, 1\right)$.

Computational overhead. During training, DAEC retains the same three model evaluations per update as the original pipeline. The reference model and lagged policy are evaluated at X_t^{ana}, while the trainable model is evaluated at $\widetilde{X}_t$. The remaining computations consist of lightweight tensor operations and introduce approximately 2% wall-clock overhead under our training configuration. Excluding the one-time calibration, DAEC adds no model evaluations to the original training update.

G Additional Experimental Results

G.1 Observation II under Additional Reward Models

To examine whether the stage-dependent effect extends to other reward models, we additionally evaluate the paired normal and corrected generations using DeQAScore and HPSv2. As shown in Fig. 5, both DeQAScore and HPSv2 exhibit gains at intermediate timesteps, consistent with the results obtained using AestheticScore. This indicates that the stage-dependent benefit is observed across multiple reward evaluators.

G.2 Bridge Residual and Actual Rollout Gap

We directly examine the relationship between the bridge residual and the actual rollout gap. For each rollout trajectory, we construct a paired analytic state using the same generated endpoint X_0^{roll}

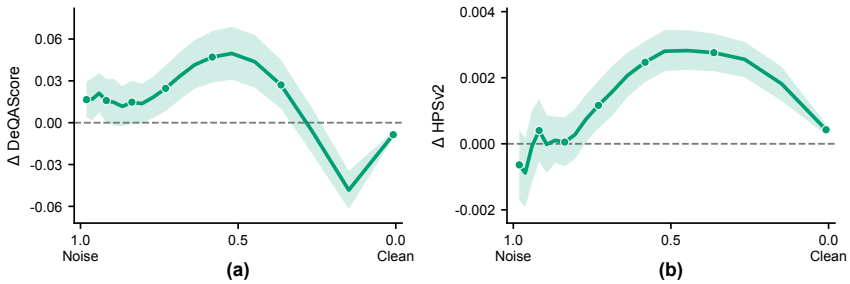

Figure 5: **Observation II under additional reward models.** Paired reward changes following trajectory-consistent correction, evaluated using (a) DeQAScore and (b) HPSv2. Both metrics exhibit gains at intermediate denoising timesteps. Lines show means, and shaded regions denote 95% confidence intervals.

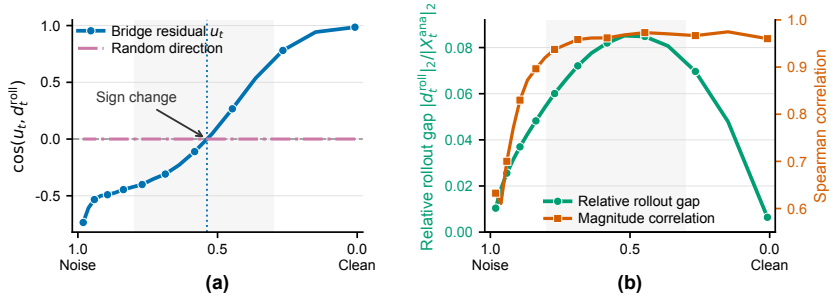

Figure 6: **Bridge residual versus the paired rollout gap.** (a) The signed alignment between u_t and d_t^{roll} varies across timesteps and changes sign near $t \approx 0.54$, while the alignment of an isotropic random direction remains near zero. (b) The relative rollout gap and the sample-wise Spearman correlation between $\|u_t\|_2$ and $\|d_t^{\text{roll}}\|_2$. Gray regions indicate the intermediate regime $t \in [0.3, 0.8]$.

and initial noise ϵ_0:

$$X_t^{\text{ana}} = (1-t)X_0^{\text{roll}} + t\epsilon_0, \qquad d_t^{\text{roll}} = X_t^{\text{roll}} - X_t^{\text{ana}}. \tag{34}$$

We measure their signed directional alignment using $\cos(u_t, d_t^{\text{roll}})$ and their sample-wise magnitude coupling using the Spearman correlation between $\|u_t\|_2$ and $\|d_t^{\text{roll}}\|_2$. The analysis covers 256 prompts with three generation seeds over 19 timesteps.

As shown in Fig. 6(a), the signed alignment between u_t and d_t^{roll} varies across timesteps and changes sign in the intermediate regime. This result complements Fig. 4(a), as the two analyses examine different geometric quantities: d_t^{roll} is an accumulated rollout displacement, whereas d_t^{corr} is a local counterfactual pullback. The alignment between $-u_t$ and d_t^{corr} identifies u_t as a correction-relevant local axis, while its varying alignment with d_t^{roll} shows that it does not directly point from X_t^{ana} toward X_t^{roll}. DAEC therefore uses u_t to define local exposure around the analytic training state, rather than to predict or reconstruct the actual rollout state.

Fig. 6(b) further examines the magnitude component of DAEC. The relative rollout gap is largest at intermediate timesteps, consistent with using $g(t)$ to concentrate exposure augmentation in this regime. Moreover, the strong sample-wise Spearman correlation indicates that $\|u_t\|_2$ is informative about which samples exhibit larger rollout gaps, supporting the residual-magnitude signal underlying the sample-adaptive gate q_i. Importantly, these magnitude relationships do not require u_t to remain directionally aligned with d_t^{roll}.

G.3 Robustness to Inference Configurations

We further evaluate DAEC under different samplers, CFG scales, and inference steps. As shown in Table 3, DAEC consistently outperforms RAM across all evaluated configurations. Notably, DAEC achieves a GenEval score of 0.9367 with only 10 inference steps, exceeding the best RAM result across all evaluated step counts. These results indicate that the gains achieved by DAEC are not tied to a particular inference configuration.

Table 3: GenEval results under different inference configurations. Only one inference parameter is varied in each block, while the others remain fixed at the default setting.

Inference Configuration	RAM	**RAM + DAEC**
Sampler Variation (*CFG* $= 4.5$, *Steps* $= 40$)		
FlowMatch-Euler (Esser et al., 2024)	0.9241	**0.9462**
FlowMatch-Heun (Karras et al., 2022)	0.9227	**0.9408**
DPM-Solver++ (flow) (Lu et al., 2022)	0.9281	**0.9458**
UniPC (flow) (Zhao et al., 2023)	0.9222	**0.9453**
CFG-Scale Variation (*FlowMatch-Euler, Steps* $= 40$)		
CFG $= 2.0$	0.9281	**0.9439**
CFG $= 3.0$	0.9281	**0.9498**
CFG $= 4.5$	0.9241	**0.9462**
CFG $= 6.0$	0.9073	**0.9317**
CFG $= 7.5$	0.9055	**0.9281**
Inference-Step Variation (*FlowMatch-Euler, CFG* $= 4.5$)		
10 steps	0.9164	**0.9367**
20 steps	0.9218	**0.9494**
28 steps	0.9222	**0.9448**
40 steps	0.9241	**0.9462**
50 steps	0.9254	**0.9448**

Table 4: Comprehensive comparison between RAM and RAM + DAEC using their respective best checkpoints. Results are reported as mean and standard deviation over three evaluation seeds. Higher is better for all metrics.

	Task Performance	*Image Quality Metrics*			
Method	GenEval ↑	Aesthetic ↑	DeQA ↑	ImgRwd ↑	HPSv2 ↑
RAM (Best)	0.9227 ± 0.0011	5.1913 ± 0.0034	3.7808 ± 0.0042	0.9816 ± 0.0024	0.2681 ± 0.0005
RAM + DAEC (Best)	$\mathbf{0.9438 \pm 0.0022}$	$\mathbf{5.4315 \pm 0.0050}$	$\mathbf{3.9268 \pm 0.0058}$	$\mathbf{1.0626 \pm 0.0028}$	$\mathbf{0.2746 \pm 0.0001}$
	GenEval Fine-Grained Evaluation				
Method	Single Obj. ↑	Two Obj. ↑	Counting ↑	Colors ↑	Position ↑	Color Attr. ↑
RAM (Best)	0.9812 ± 0.0000	0.9242 ± 0.0041	0.9625 ± 0.0117	0.8741 ± 0.0063	0.9183 ± 0.0031	0.7392 ± 0.0062
RAM + DAEC (Best)	0.9812 ± 0.0044	$\mathbf{0.9512 \pm 0.0137}$	0.9646 ± 0.0015	$\mathbf{0.9043 \pm 0.0095}$	$\mathbf{0.9575 \pm 0.0020}$	$\mathbf{0.7883 \pm 0.0153}$

G.4 Fine-Grained and Multi-Seed Evaluation

To examine the consistency of the results beyond a single generation seed, we evaluate the respective best checkpoints of RAM and DAEC using three independent evaluation seeds. As shown in Table 4, DAEC improves the mean GenEval score from 0.9227 to 0.9438, with small variation across evaluation seeds. It also achieves higher mean scores across all reported image-quality metrics. The fine-grained evaluation shows improvements in two-object generation, color generation, spatial relations, and color-attribute binding under the strict criterion. Single-object performance is nearly identical, while the difference in counting is small relative to its evaluation variability. These results support the consistency of DAEC's gains across evaluation seeds.

Table 5: Multi-seed training robustness of RAM + DAEC on GenEval.

	Task Performance	*Image Quality Metrics*			
Method	GenEval ↑	Aesthetic ↑	DeQA ↑	ImgRwd ↑	HPSv2 ↑
RAM (Best)	0.9241	5.1867	3.7790	0.9803	0.2679
RAM + DAEC (Seed 0)	0.9462	5.4276	3.9342	1.0637	0.2746
RAM + DAEC (Seed 1)	0.9349	5.4430	3.8899	1.0512	0.2704
RAM + DAEC (Seed 2)	0.9317	5.3952	3.8037	1.0155	0.2714
RAM + DAEC (Mean ± Std.)	$\mathbf{0.9376 \pm 0.0062}$	$\mathbf{5.4219 \pm 0.0199}$	$\mathbf{3.8759 \pm 0.0542}$	$\mathbf{1.0435 \pm 0.0204}$	$\mathbf{0.2721 \pm 0.0018}$

G.5 Multi-Seed Training Robustness

We evaluate DAEC using three training seeds. As shown in Table 5, all three runs outperform the RAM baseline on GenEval and image-quality metrics. DAEC achieves a mean GenEval score of 0.9376 ± 0.0062, compared with 0.9241 for RAM. These results support the consistency of DAEC's performance gains across the evaluated training seeds.

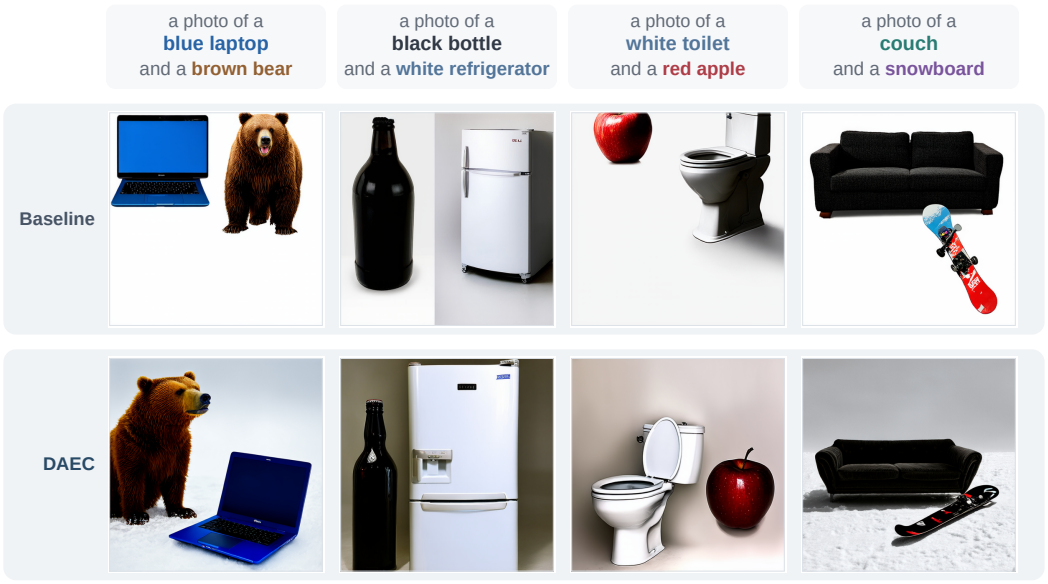

Figure 7: Qualitative comparisons between the RAM baseline and DAEC. DAEC retains the specified objects and color attributes and often yields a more integrated photographic composition.

Table 6: Results on Qwen-Image at 512×512 resolution. Higher is better.

Method	Steps	GenEval ↑
Qwen-Image	0	0.8535
RAM	110	0.9385
RAM + DAEC	80	**0.9498**

G.6 Transferability across Backbones

To examine whether DAEC transfers beyond SD3.5M, we evaluate RAM and RAM + DAEC on Qwen-Image (Wu et al., 2025). All models are evaluated at 512×512 resolution using 50 inference steps and a CFG scale of 4.0. As shown in Table 6, DAEC achieves a GenEval score of 0.9498 after 80 training steps, outperforming RAM's 0.9385 after 110 steps. These results support the transferability of DAEC to an additional backbone.

G.7 Qualitative analysis

Figs. 7 and 8 show samples generated from the same prompts, covering color–object binding, two-object composition, and spatial relationships. As shown in Fig. 7, DAEC preserves the requested objects and attributes while often producing a more integrated scene. For example, it presents the toilet and apple as complete objects and renders the bottle–refrigerator and couch–snowboard pairs in coherent shared scenes. Fig. 8 provides additional qualitative comparisons over a broader range of prompts. Overall, compared with RAM, DAEC generally generates higher-quality images with better semantic fidelity and more coherent visual compositions, further demonstrating its effectiveness in improving generation performance.

H AI use statement

In this work, we used generative AI tools to polish language, correct grammar, provide feedback on experimental design, interpret and present experimental results, and assist with code. The authors reviewed all AI-assisted revisions and suggestions before incorporating them into the manuscript or implementation. We take responsibility for the final content of this work.

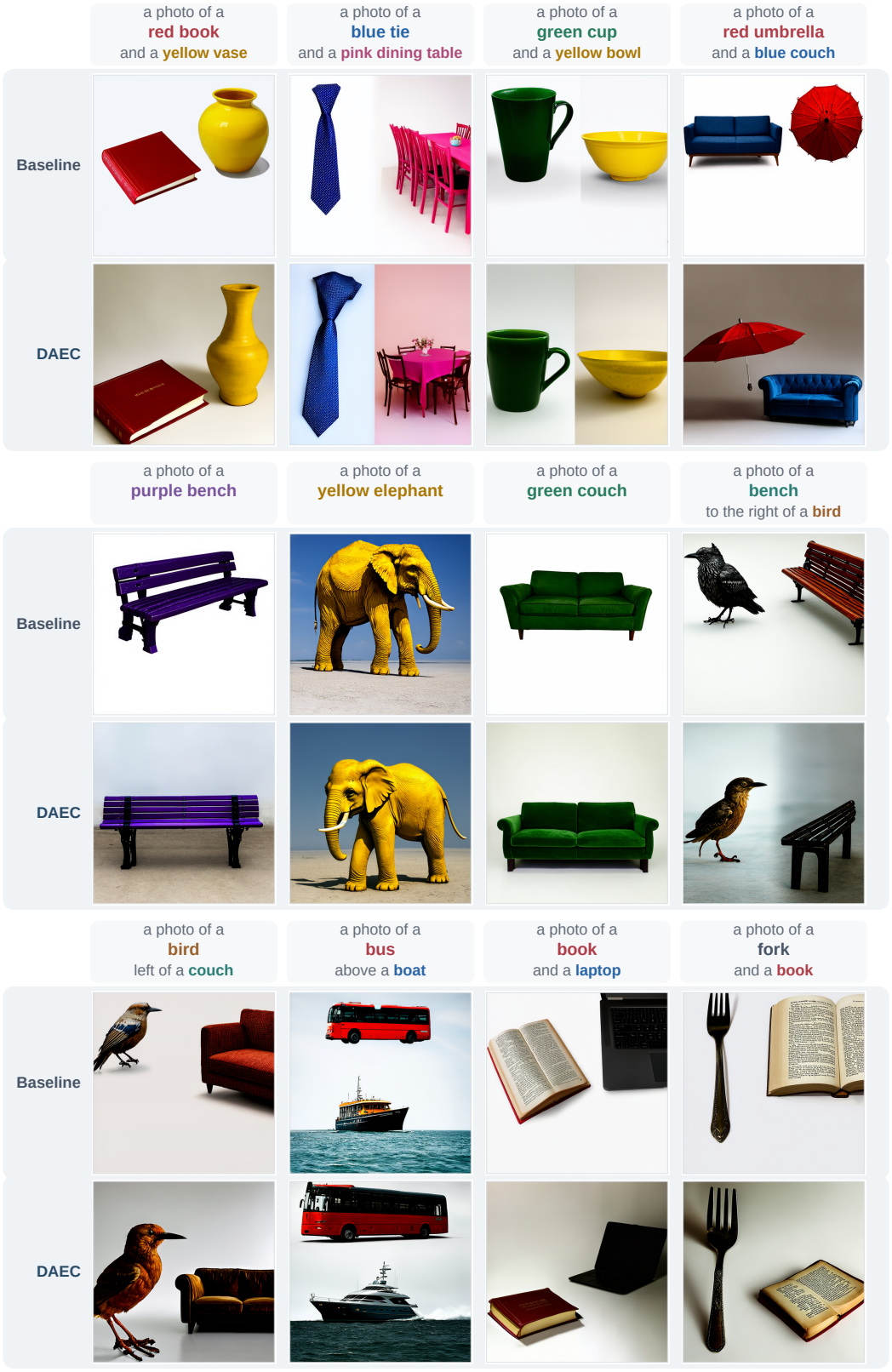

Figure 8: More qualitative comparisons between RAM and DAEC.

I PyTorch-style Pseudocode

The following pseudocode summarizes batched DAEC training. For clarity, we omit distributed execution, low-level adapter switching, gradient accumulation, and logging, while retaining the operations that affect the DAEC update.

```
for epoch in training_epochs:

    # On-policy endpoint sampling and credit assignment
    x0_all, adv_all, cond_all = sample_on_policy_groups(
        lagged_model, prompts
    )

    for x0, adv, cond in endpoint_minibatches(
        x0_all, adv_all, cond_all
    ):

        # Construct K analytic targets for each endpoint
        x0, adv, cond = repeat_targets(x0, adv, cond, K)

        # Analytic states and sample-wise path velocities
        t = sample_timesteps(x0.shape[0])
        eps = torch.randn_like(x0)
        x_ana = (1 - t) * x0 + t * eps
        v_ana = eps - x0

        # Frozen predictions remain anchored at analytic states
        with torch.no_grad():
            v_ref = reference_model(x_ana, t, cond)
            v_old = lagged_model(x_ana, t, cond)

        # Normalized drift-aligned directions
        u = v_old - v_ana
        rho = sample_l2norm(u)
        u_hat = u / rho.clamp_min(1e-8)

        # Time- and sample-adaptive exposure magnitudes
        q = local_residual_gate(rho, t)
        eta = (
            eta_scale
            * warmup(epoch)
            * gap_profile(t)
            * sample_l2norm(x_ana)
            * q
        )

        # Only the trainable branch receives perturbed states
        x_tilde = x_ana + eta * u_hat
        v_pred = trainable_model(x_tilde, t, cond)

        # Regression targets remain anchored at analytic states
        target = v_ref + lambda_ * adv * (v_ana - v_old)

        # Trainable-branch update
        loss = mse(v_pred, target.detach())
        optimizer_step(loss)

    # Update the lagged policy after each training epoch
    update_lagged_policy(lagged_model, trainable_model, epoch)
```