DIFFUSIONART: EFFICIENT, STABLE AND SCALABLE REINFORCEMENT OF DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reward-guided fine-tuning adapts diffusion models to preferences and task-specific objectives beyond pretraining. Yet scaling it remains difficult: reverse-process methods such as Flow-GRPO explore locally and have poor credit assignment, limiting their efficiency, while forward-process methods such as DiffusionNFT can be unstable without carefully tuned off-policy training. We show empirically and analytically that the instability comes from **a hidden forward–reverse consistency force** beyond the ideal NFT-style reward guidance, and that **the restoration gradient is central** to the stabilizing effect of off-policy updates. We introduce **DiffusionART** (Diffusion Reinforcement with Aggregation-First Reward Tilting), a simple yet effective method that systematically cancels the unstable self-consistency force and provides adaptive restoration. It requires minimal hyperparameter tuning, remains effective with fewer samples per prompt, and naturally supports pairwise reward models (**DiffusionART-Pairwise**). Experiments spanning text-to-image, text-to-video, and image-to-video generation show that DiffusionART is highly efficient and stable. For example, on HunyuanVideo-13B text-to-video generation, DiffusionART matches the plateau training rewards of DiffusionNFT and Flow-GRPO-Fast with 11.19× and 11.23× fewer GPU hours, respectively.

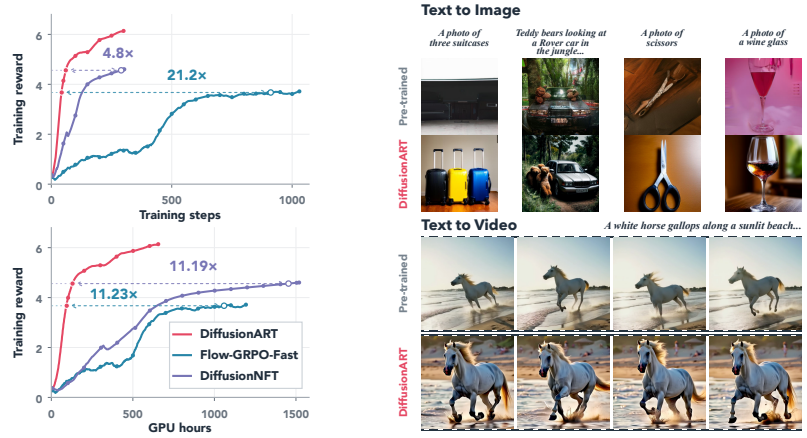

Figure 1: **Efficient learning and improved generations. Left:** T2V reward (VQ:MQ=1:1). Thresholds average each baseline's last 50 updates; ratios compare time to these reward levels. **Right:** Selected same-prompt images and video frames before and after DiffusionART.

1 INTRODUCTION

Visual diffusion and flow-matching models[1] are typically trained to fit the data distribution by regressing noise or velocity targets rather than to optimize downstream criteria such as visual quality,

[1]We use "diffusion" as a general term. Our analysis targets flow matching with linear data–noise interpolation (Lipman et al., 2022), which shares the core principles of diffusion.

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.

motion quality, or human preference, making reward-guided fine-tuning central to recent image and video generation tasks (Xu et al., 2026a; Gao et al., 2025). We focus on online diffusion Reinforcement Learning (RL), which, unlike differentiable reward-model optimization such as ReFL (Xu et al., 2023) or offline preference learning such as Diffusion-DPO (Wallace et al., 2023), evaluates clean samples from the evolving generator and can use black-box rewards. Because these rewards arrive only after a complete output is generated, online diffusion RL faces a credit-assignment problem: costly output-level feedback must be translated into effective updates across many denoising states (Liu et al., 2025a; Zheng et al., 2025).

Within this paradigm, *reverse-process RL* (Black et al., 2023), such as Flow-GRPO and its variants (Liu et al., 2025a; Xue et al., 2025; He et al., 2025; Wang et al., 2026), optimizes sampled denoising transitions via group-based endpoint reward feedback. It trains on states actually visited by the generator but requires ODE-to-SDE conversion for tractable likelihoods, and its policy gradient updates each selected transition along its own injected noise, weighted by a trajectory-level advantage that this transition only partly determines, resulting in poor credit assignment and limiting training efficiency. *Forward-process RL*, such as DiffusionNFT (Zheng et al., 2025), instead applies forward noising to the generated clean samples and regresses velocities with reward-weighted positive and negative objectives. It needs no transition likelihoods and is highly efficient, but it is unstable without two stabilizers: joint positive–negative training and an exponential moving average (EMA) rollout policy whose lag must be tuned for each task (Zheng et al., 2025). Neither family is thus both efficient and stable without per-task tuning. In this paper, we focus on addressing the instability of DiffusionNFT and its reliance on carefully tuned hyperparameters.

We first examine NFT-style training, starting from a puzzle: its positive and negative branches share the same ideal reward-improving direction, so either alone should suffice, yet each alone collapses (Figure 2(a)). We find that NFT's targets, built from forward-noised endpoints, implicitly assume that the model's sampling velocity equals the ideal forward velocity induced by its own endpoints. The two differ in practice, and the gap introduces a hidden *forward–reverse consistency force* that does not improve the reward: the positive branch pulls each state's velocity toward the ideal forward velocity and the negative branch pushes it away. Joint training cancels this force only partially, since reward centering balances rewards across samples rather than at each state. The residual can destabilize training and slow the reward optimization. We further show that EMA stabilizes NFT through the *restoration* force of its lagged reference, not through off-policy rollouts, but sets this force only indirectly and must be tuned per task.

Driven by these insights, we introduce **DiffusionART** (Diffusion Reinforcement with Aggregation-First Reward Tilting). It borrows supervised velocity regression from forward-process training but applies updates to states visited during reverse-process rollouts. It first aggregates each prompt's samples and rewards into a shared reward-tilting direction $G(c)$, and then regresses each visited state's velocity toward its cached prediction plus $G(c)$ (Section 4.1). As this target never involves the forward velocity, the consistency force is removed by construction rather than by cross-sample cancellation, and each state receives an explicit reward-improving direction instead of a scalar advantage passed through likelihood ratios. A second update on each rollout adds a restoration force proportional to each state's drift, replacing EMA with no schedule to tune (Section 4.2). We summarize our main contributions as follows:

Main Contributions

1. Diagnosing the instability of NFT-style training. The NFT objective contains a hidden forward–reverse consistency force that joint training cancels only partially (Section 3.1).

2. Identifying restoration as the key stabilizer. EMA stabilizes NFT through restoration rather than off-policy data, but requires per-task tuning (Section 3.2).

3. Aggregation-first reward tilting with adaptive restoration. DiffusionART removes this force by construction, replaces EMA with drift-adaptive restoration (Section 4).

4. Efficient and stable training across image and video generation. On SD3.5-M (T2I), DiffusionART trains stably without EMA or per-task tuning, with training efficiency comparable to the strongest baseline, DiffusionNFT (EMA); on HunyuanVideo-13B (T2V), it achieves 31.4% and 31.1% higher final VQ and MQ than the best baselines; on Wan2.1-I2V-14B (I2V), it reaches an MQ reward of 6 with 5.6× fewer GPU hours than Flow-GRPO-Fast (Section 5).

2 PRELIMINARIES

Diffusion RL (Xu et al., 2026b) includes reverse-process policy optimization, such as Flow-GRPO (Liu et al., 2025a), and forward-process optimization, such as DiffusionNFT (Zheng et al., 2025; Zhu et al., 2026; Xue et al., 2026). We focus on DiffusionNFT and defer Flow-GRPO to Appendix D.1. At each iteration, for each prompt c, DiffusionNFT samples K clean images or videos x_0 from a frozen rollout policy $p_{\text{old}}(x_0 \mid c)$; we call these generated samples *endpoints*. DiffusionNFT maps each raw reward $R_i := R(x_0^{(i)}, c)$ to a *goodness* weight $r_i := r(x_0^{(i)}, c) = \frac{1}{2}[1+\text{clip}((R_i-\bar{R}_c)/Z_c, -1, 1)] \in [0,1]$, where $\bar{R}_c = K^{-1}\sum_i R_i$ is the prompt's mean reward and $Z_c > 0$ controls the scale; without clipping, $\sum_i(2r_i - 1) = 0$. Weighting p_{old} by r and $1-r$ induces the positive and negative endpoint distributions $p_+(x_0 \mid c) \propto r(x_0, c)p_{\text{old}}(x_0 \mid c)$ and $p_-(x_0 \mid c) \propto [1 - r(x_0, c)]p_{\text{old}}(x_0 \mid c)$. DiffusionNFT is motivated by the result that the ideal velocities induced by forward noising the positive, original, and negative endpoint distributions are collinear. Moving toward the positive field and away from the negative field therefore follows the same ideal improvement direction.

We parameterize generation from $t = 1$ (noise) to $t = 0$ (clean samples), so velocity points along the denoising direction. For a forward state $x_t = \alpha_t x_0 + \sigma_t \epsilon$, let $u_t = -\dot{\alpha}_t x_0 - \dot{\sigma}_t \epsilon$ denote its sample velocity target, where dots denote derivatives with respect to t; standard rectified-flow interpolation (Lipman et al., 2022; Liu et al., 2022) gives $u_t = x_0 - \epsilon$. Let $v_{\text{old}} := v_{\theta_{\text{old}}}(x_t, t, c)$ and $v_\theta(x_t, t, c)$ be the velocities predicted by the rollout policy p_{old} (frozen within an iteration) and the trainable policy, respectively, and let $\mathcal{D}_{\text{old}}$ be the training distribution of (c, x_0, r, t, ϵ) with $x_0 \sim p_{\text{old}}(\cdot \mid c)$ and $\epsilon \sim \mathcal{N}(0, I)$. For $\beta > 0$, DiffusionNFT constructs $v_{\theta,\pm} = v_{\text{old}} \pm \beta(v_\theta - v_{\text{old}})$ and minimizes $\mathcal{L}_{\text{NFT}} = \mathbb{E}_{\mathcal{D}_{\text{old}}}[r\|v_{\theta,+} - u_t\|_2^2 + (1-r)\|v_{\theta,-} - u_t\|_2^2]$. Equivalently, let $a = (2r-1)/\beta$ and define the frozen target $v_{\text{tgt}} = au_t + (1-a)v_{\text{old}}$. Then

$$\mathcal{L}_{\text{NFT}} = \beta^2 \mathbb{E}_{\mathcal{D}_{\text{old}}}\left[\|v_\theta - v_{\text{tgt}}\|_2^2\right] + \text{const}, \tag{1}$$

where const is independent of θ. At the on-policy initialization $v_\theta = v_{\text{old}}$, each sample's output-space descent direction is proportional to $(2r-1)(u_t - v_{\text{old}})$, so high- and low-reward samples drive the same regression residual in opposite directions. After optimizing the rollout batch, DiffusionNFT updates the rollout policy by exponential moving average (EMA): $\theta_{\text{old}} \leftarrow \eta\theta_{\text{old}} + (1-\eta)\theta$.

3 UNDERSTANDING NFT-STYLE DIFFUSION REINFORCEMENT LEARNING

DiffusionNFT is highly efficient, but its stability relies on joint positive–negative training and an off-policy EMA rollout policy (Zheng et al., 2025). We examine why both are needed and where they fall short: joint training only partially cancels a hidden forward–reverse consistency force, while EMA provides restoration whose strength requires task-specific tuning. These limitations motivate the two components of DiffusionART in Section 4.

3.1 JOINT POSITIVE–NEGATIVE TRAINING: PARTIAL CANCELLATION OF A HIDDEN CONSISTENCY FORCE

We first remove EMA to isolate joint training, so $v_\theta = v_{\text{old}}$ at every on-policy update. Consistent with Zheng et al. (2025), either branch alone collapses, whereas joint training learns effectively (Figure 2(a)). This is puzzling: the ideal positive and negative guidance directions are collinear, suggesting that either branch alone should suffice.

To make this precise, for $b \in \{+, \text{old}, -\}$, let P_b denote the endpoint distribution $p_b(x_0 \mid c)$, and let Q_b denote the joint **forward** distribution of $(x_0, \epsilon, x_t, u_t)$ obtained by drawing $x_0 \sim P_b$ and $\epsilon \sim \mathcal{N}(0, I)$ independently and forming x_t and $u_t = x_0 - \epsilon$ by forward noising. With $s_t = (x_t, t, c)$, the *ideal forward velocity* induced by P_b is $v_{\text{fwd}}^b(s_t) := \mathbb{E}_{Q_b}[u_t \mid s_t]$. We write $v_{\text{fwd}}^{\text{old}}$ for $b = \text{old}$, and $v_{\text{rvs}}^{\text{old}} := v_{\text{old}}$ for the field that the reverse sampler actually uses to generate the endpoint distribution.

Lemma 1 (Ideal NFT guidance is conditional reward covariance)**.** *At a fixed state s_t, let Q_{old} denote Q_b with $b = \text{old}$, and let $q := \mathbb{E}_{Q_{\text{old}}}[r(x_0, c) \mid s_t]$ and $g := \text{Cov}_{Q_{\text{old}}}[r(x_0, c), u_t \mid s_t]$. If the tilted distributions and moments exist and $0 < q < 1$, then*

$$q\big(v_{\text{fwd}}^+ - v_{\text{fwd}}^{\text{old}}\big) = (1-q)\big(v_{\text{fwd}}^{\text{old}} - v_{\text{fwd}}^-\big) = g, \tag{2}$$

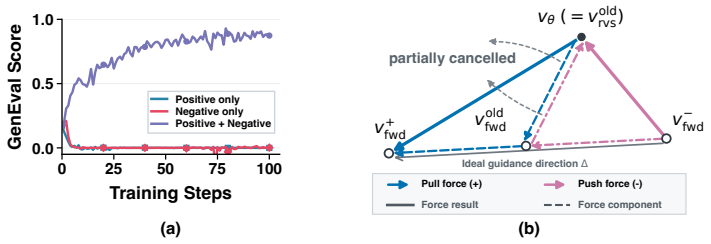

Figure 2: (a) Ablation results for positive and negative objectives. (b) The positive and negative objectives partially cancel the consistency force.

where all velocities are evaluated at s_t. Hence the positive and negative branches point along the same reward-tilt direction g.

Appendix B.1 provides the proof. The ideal guidance of the two branches thus differs only in step size: under *forward–reverse compatibility*, i.e., if the reverse sampler satisfied $v^{\text{old}}_{\text{rvs}} = v^{\text{old}}_{\text{fwd}}$, either branch alone would suffice. In practice, however, the reverse sampling field generally differs from the ideal forward velocity induced by its own endpoints. We denote this forward–reverse gap by $e := v^{\text{old}}_{\text{fwd}}(s_t) - v^{\text{old}}_{\text{rvs}}(s_t)$. Proposition 1 shows how it enters the NFT update.

Proposition 1 (Practical forward-process RL contains a forward–reverse consistency force)**.** *Let $\mathcal{L}_{\text{NFT}}(s_t)$ denote Equation 1 conditioned on s_t, q and g follows the definition of Lemma 1. At the pure on-policy update point $v_\theta = v^{\text{old}}_{\text{rvs}}$ and for $\beta > 0$, its normalized output-space force is*

$$\mathcal{F}_{\text{NFT}}(s_t) := -\frac{1}{2\beta}\,\nabla_{v_\theta(s_t)}\mathcal{L}_{\text{NFT}}(s_t)\big|_{v_\theta = v^{\text{old}}_{\text{rvs}}} = 2g + (2q-1)e. \tag{3}$$

Why both branches are needed. Proposition 1 decomposes the NFT force into reward-improving guidance and a self-consistency force weighted by the statewise reward imbalance $2q - 1$. The individual branch forces are $\mathcal{F}_+ = g + qe$ and $\mathcal{F}_- = g - (1-q)e$, which sum to $\mathcal{F}_{\text{NFT}}$ (proof in Appendix B.2). Each branch alone thus carries an uncancelled consistency force: the positive branch pulls the velocity toward $v^{\text{old}}_{\text{fwd}}$ with weight q, whereas the negative branch pushes it away with weight $1 - q$. Joint training therefore does more than double the guidance: the branches reinforce each other along g but oppose each other along e, leaving $(2q-1)e$. Without clipping, prompt-level reward centering makes q average to $1/2$ across states, allowing partial cancellation after gradient aggregation (Figure 2(b)).

Why joint training is not enough. For nonzero consistency gaps, cancellation is exact only if $q = 1/2$ at every state, which prompt-level normalization does not guarantee: $\sum_i (2r_i - 1) \approx 0$ neither balances each state nor implies $\sum_i (2r_i - 1)e_i \approx 0$, where e_i denotes the one-sample Monte Carlo estimate of e. States with $q > 1/2$ are systematically pulled toward $v^{\text{old}}_{\text{fwd}}$, while those with $q < 1/2$ are pushed away. At $t = 1$, the state contains no endpoint information, so without clipping $q = 1/2$ and the residual vanishes. As noise decreases, states reveal more endpoint quality, making q more extreme and increasing the weight $|2q - 1|$ on the gap. The residual dilutes reward-improving guidance in the aggregate gradient and requires an additional stabilizer (Section 3.2).

Insight 1: Joint training cancels the consistency force only partially

In NFT-style forward-process training, either branch alone carries an uncancelled forward–reverse consistency force that can destabilize training; joint training only partially cancels it across samples, not at each state. A better design should remove this force statewise rather than rely on cross-sample cancellation (Section 4.1).

3.2 Off-Policy EMA: Stability through Restoration, at the Cost of Task-Specific Tuning

Even with joint training, pure on-policy NFT collapses after a sharp rise in its pre-clipping parameter-gradient norm (Figure 3(a)). EMA may stabilize training through a slower-moving rollout distribution or through the lagged reference v_{old} retained in the loss. We show that the latter, namely restoration toward this reference, is the key stabilizing mechanism. Let $d = v_\theta - v_{\text{old}}$ denote the trainable field's drift from the reference and $h = u_t - v_{\text{old}}$ the sample training residual.

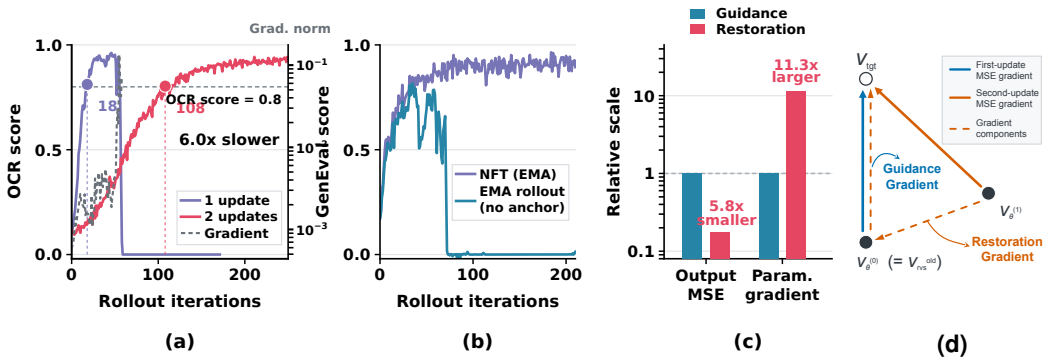

Figure 3: (a) One update per rollout collapses after a spike in its pre-clipping gradient norm (dashed, right axis); two MSE updates remain stable but require $6\times$ more rollouts to reach an OCR score of 0.80. (b) EMA-policy rollouts without the old-policy anchor still collapse. (c) Averaged over rollouts 0–17, restoration MSE is $5.8\times$ smaller than guidance MSE, yet its parameter-gradient norm is $11.3\times$ larger. (d) The second MSE update decomposes into guidance and restoration.

Proposition 2 (EMA brings an explicit restoring term to DiffusionNFT)**.**

$$\mathcal{L}_{\mathrm{NFT}} = \mathbb{E}_{\mathcal{D}_{\mathrm{old}}}\left[-2\beta(2r-1)\langle d, h\rangle + \beta^2\|d\|_2^2\right] + \mathrm{const}, \tag{4}$$

where const is independent of the trainable parameters.

Why EMA stabilizes. Proposition 2 separates the NFT loss into a linear term carrying both reward guidance and the consistency force, and a quadratic term restoring v_θ toward v_{old} (proof in Appendix B.2). At $d = 0$, the restoration gradient vanishes. Pure on-policy NFT therefore receives no restoration, leaving the residual consistency force unopposed. EMA maintains a nonzero drift from the lagged reference, allowing restoration to counteract this residual.

We test these explanations with two interventions. First, retaining EMA-policy rollouts but removing the old-policy anchor from the gradient still causes collapse (Figure 3(b)): off-policy data alone does not stabilize NFT. Second, without EMA, we apply two minibatch MSE updates to each frozen rollout batch using the same frozen target v_{tgt}. The first update starts at $v_\theta^{(0)} = v_{\mathrm{old}}$, where restoration is zero. After the model moves to $v_\theta^{(1)}$, the second update combines the original guidance with restoration toward $v_\theta^{(0)}$ (Figure 3(d)). This two-update run remains stable without EMA (Figure 3(a)), supporting restoration as the key stabilizing mechanism.

Why EMA is not enough. EMA controls restoration indirectly through the reference policy's lag: too little restoration risks collapse, while too much slows reward improvement. Its schedule therefore requires task-specific tuning. DiffusionNFT's results illustrate this sensitivity (Zheng et al., 2025): fully on-policy updates collapse, nearly offline updates converge impractically slowly, and the default schedule $\eta = \min(0.001i, 0.5)$, where i is the iteration, is unstable on OCR, which requires a maximum retention of 0.999. Such tuning limits scalability.

Two NFT updates per rollout provide restoration without EMA (Figure 3(d)), but require $6\times$ more rollouts to reach an OCR score of 0.80 (Figure 3(a)). Averaged over rollouts 0–17, restoration MSE is $5.8\times$ smaller than guidance MSE, yet its aggregated parameter gradient is $11.3\times$ larger (Figure 3(c)). This suggests that heterogeneous per-sample guidance largely cancels across states, while restoration survives aggregation and dominates the update, slowing reward optimization. We leave a detailed analysis to future work.

Insight 2: EMA stabilizes NFT, but at the cost of tuning or efficiency

Restoration stabilizes NFT-style training, but EMA requires task-specific tuning, while two updates per rollout avoid EMA tuning at the cost of slower reward improvement. A better design should adapt restoration to each state's drift without task-specific tuning or overwhelming reward guidance (Section 4.2).

4 DIFFUSIONART: AGGREGATION-FIRST REWARD TILTING WITH ADAPTIVE RESTORATION

DiffusionART addresses the two requirements in Section 3. It aggregates each prompt's endpoints and rewards into a shared reward-tilting direction for reverse-rollout states, removing the consistency force statewise without relying on cross-sample cancellation (Section 4.1). A second MSE update per rollout batch adapts restoration to each state's drift without task-specific tuning or overwhelming reward guidance (Section 4.2). Section 4.3 extends the method to pairwise reward models; Algorithm 1 summarizes one training iteration.

4.1 AGGREGATION-FIRST GUIDANCE

The consistency force enters NFT because its regression target is built from the forward velocity $v_{\text{fwd}}^{\text{old}}$, which differs from the reverse field by e (Proposition 1). We instead aggregate each prompt's endpoints and rewards alone into guidance for reverse-rollout states. At each iteration, we exponentially tilt the current rollout distribution, rather than a fixed pretrained distribution (Potaptchik et al., 2025; Zhu et al., 2026): $p_\tau(x_0 \mid c) \propto p_{\text{old}}(x_0 \mid c) \exp\big(\tau R(x_0, c)\big)$. Here $\tau \geq 0$ controls the preference for higher-reward endpoints, and $\tau = 0$ recovers p_{old}. The first-order change in the endpoint mean is a covariance:

Lemma 2 (Reward tilting moves the endpoint mean along the reward–endpoint covariance)**.** *If the relevant moments exist and differentiation can be exchanged with expectation near $\tau = 0$, then* $\frac{\partial}{\partial \tau} \mathbb{E}_{p_\tau}[x_0 \mid c]\big|_{\tau=0} = \text{Cov}_{p_{\text{old}}}(R, x_0 \mid c)$.

Lemma 2, proved in Appendix B.3, shows that up-weighting higher-reward endpoints shifts their mean along $\text{Cov}_{p_{\text{old}}}(R, x_0 \mid c)$. Under rectified flow, $u_t = x_0 - \epsilon$ with ϵ independent of x_0, so $\text{Cov}_{Q_{\text{old}}}(R, u_t \mid c) = \text{Cov}_{p_{\text{old}}}(R, x_0 \mid c)$ at every noise level. This direction is the prompt-level counterpart of ideal NFT guidance g (Lemma 1): without clipping, r is affine in R, and at $t = 1$, when the forward state reveals no endpoint information, g equals this covariance up to a positive factor. We estimate the covariance from each prompt's K rollout endpoints and rewards, then normalize it to obtain $G(c)$:

$$\boxed{\overline{\text{Cov}}(R, x_0 \mid c) = \frac{1}{K} \sum_{i=1}^{K} (R_i - \bar{R}_c)(x_0^{(i)} - \bar{x}_{0,c}), \qquad G(c) = \frac{\overline{\text{Cov}}}{\text{RMS}(\overline{\text{Cov}})}}. \tag{5}$$

Here $\bar{R}_c$ and $\bar{x}_{0,c}$ are the group means. RMS normalization fixes the guidance magnitude across prompts and reward models, removing differences in reward scale and spread without a guidance-strength coefficient. We apply $G(c)$ to every rollout state s_t of prompt c and freeze the target $v_{\text{tgt}}^{\text{ART}}(s_t) = v_\theta^{(0)}(s_t) + G(c)$, where $v_\theta^{(0)} = v_{\text{old}}$ is the prediction cached during rollout. At $v_\theta = v_\theta^{(0)}$, regression produces an output-space force proportional to $G(c)$ at every state (Equation 6). Unlike the NFT force in Equation 3, this force contains no consistency term, satisfying Insight 1. Figure 4 summarizes the construction.

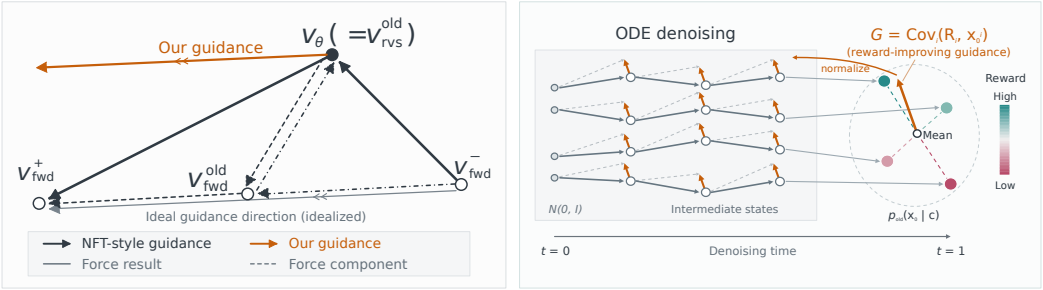

Figure 4: **Aggregation-first guidance. Left:** NFT's positive and negative branches introduce opposing consistency forces that need not cancel (Section 3.1). DiffusionART avoids these forces by applying $G(c)$ directly at the reverse velocity. **Right:** We aggregate each prompt's endpoints and rewards into normalized guidance $G(c)$ (Equation 5) and apply it to every state along its rollouts.

4.2 Two-Update Restoration: Drift-Adaptive Stability without EMA Tuning

Instead of tuning an EMA restoration schedule (Section 3.2), we split each rollout batch into two disjoint minibatches and apply one MSE update to each. Let $v_\theta^{(0)}$ and $v_\theta^{(1)}$ denote the velocity fields before and after the first update. Both updates use the frozen target $v_{\text{tgt}}^{\text{ART}} = v_\theta^{(0)} + G(c)$. For each state s_t, the loss is $\mathcal{L}_\theta(s_t) = w(s_t)\,\text{MSE}(v_\theta(s_t), v_{\text{tgt}}^{\text{ART}}(s_t))$, where $\text{MSE}(a,b) = D^{-1}\|a-b\|_2^2$, D is the number of output coordinates, and w combines timestep weighting and masking. Writing the first update's displacement as $d^{(1)} = v_\theta^{(1)} - v_\theta^{(0)}$, the loss and its output gradient before each update are:

$$\begin{aligned} \mathcal{L}^{(0)} &= w\,\text{MSE}(v_\theta^{(0)}, v_{\text{tgt}}^{\text{ART}}) = \frac{w}{D}\|G(c)\|_2^2 & \mathcal{L}^{(1)} &= w\,\text{MSE}(v_\theta^{(1)}, v_{\text{tgt}}^{\text{ART}}) = \frac{w}{D}\|G(c) - d^{(1)}\|_2^2 \\ -\nabla_{v_\theta}\mathcal{L}_\theta\big|_{v_\theta = v_\theta^{(0)}} &= \frac{2w}{D}\,G(c) & -\nabla_{v_\theta}\mathcal{L}_\theta\big|_{v_\theta = v_\theta^{(1)}} &= \frac{2w}{D}\big(G(c) - d^{(1)}\big). \end{aligned} \tag{6}$$

The second update combines the original guidance $G(c)$ with restoration $-d^{(1)}$ toward $v_\theta^{(0)}$. Restoration grows with each state's drift, counteracting large displacements while leaving guidance dominant when drift is small, without additional tuning. In NFT, cross-state cancellation weakens guidance, allowing restoration to dominate and slow training (Section 3.2). DiffusionART instead shares $G(c)$ across each prompt's states, reducing this cancellation and improving stability without sacrificing efficiency (Insight 2; Figure 6(c)).

Implementation details. The weight $w(s_t)$ combines noise-level weighting t with a one-sided mask adapted from Flow-DPPO (Ping et al., 2026). The second update masks a state only when its drift exceeds a threshold and its MSE descent direction would increase that drift (Appendix D.3).

Algorithm 1 DiffusionART: one training iteration

1: Sample with an ODE solver using $v_\theta^{(0)} = v_\theta$. **Rollout Stage**
   For each prompt c, collect K endpoints with rewards R, states s_t, and predictions $v_\theta^{(0)}(s_t)$.
2: Compute $G(c)$ by Equation 5 for each prompt c.
3: Freeze $v_{\text{tgt}}^{\text{ART}}(s_t) = v_\theta^{(0)}(s_t) + G(c)$ for each state s_t of prompt c.

4: Split the samples into two equal-sized disjoint minibatches B_1, B_2. **Training Stage**
5: For each $B \in \{B_1, B_2\}$, recompute $w(s_t)$ and update θ with one step on
$\mathcal{L}(B;\theta) = \frac{1}{|B|}\sum_{s_t \in B} w(s_t)\,\text{MSE}(v_\theta(s_t), v_{\text{tgt}}^{\text{ART}}(s_t))$.

4.3 Extension to Pairwise Reward Models

DiffusionART-Pairwise supports pairwise preference feedback (Kirstain et al., 2023). For each prompt, we divide K endpoints into $K/2$ disjoint pairs and assign winner/loser labels $(1, 0)$, or $(1/2, 1/2)$ for ties. These labels replace scalar rewards in the otherwise unchanged covariance and training pipeline. Appendix D.4 provides the covariance identity and pseudocode.

5 Experiments

In our experiments, we study the efficiency, stability, and scalability of DiffusionART. Our experiments progress from text-to-image (T2I) generation to larger models and higher-dimensional text-to-video (T2V) and image-to-video (I2V) generation.

5.1 Text-to-Image Generation

Setup. We compare DiffusionART with Flow-GRPO, Flow-GRPO-Fast (CPS) (Liu et al., 2025a; Wang & Yu, 2025), DanceGRPO (Xue et al., 2025), Flow-DPPO (Ping et al., 2026), and DiffusionNFT (Zheng et al., 2025) with and without EMA, denoted DiffusionNFT (EMA) and DiffusionNFT (on-policy), respectively. For all methods, we train LoRA adapters (Hu et al., 2021) with a rank of $r = 32$ and a scaling factor of $\alpha = 64$ at 512×512 resolution, without CFG. Our method and all baselines use a learning rate of 3×10^{-4}, 10-step rollouts, without reference-policy KL penalty.

Single-reward and multi-reward optimization. We evaluate SD3.5-M (Stability AI, 2024) on GenEval (Ghosh et al., 2023), OCR, and PickScore (Kirstain et al., 2023). Figure 5(a–c) shows that DiffusionART avoids the severe collapse of DiffusionNFT (on-policy) and reaches Flow-GRPO's step-500 reward with $10.0\times$, $5.2\times$, and $9.6\times$ fewer training steps on GenEval, OCR, and PickScore, respectively. DiffusionART achieves training efficiency comparable to DiffusionNFT (EMA) without the per-task EMA tuning that DiffusionNFT needs for stability. We also jointly optimize GenEval, PickScore, CLIPScore (Hessel et al., 2021), and HPSv2.1 (Wu et al., 2023) on SD3.5-M. Figure 5(d) reports their summed reward; Appendix A.2 gives the component-wise curves.

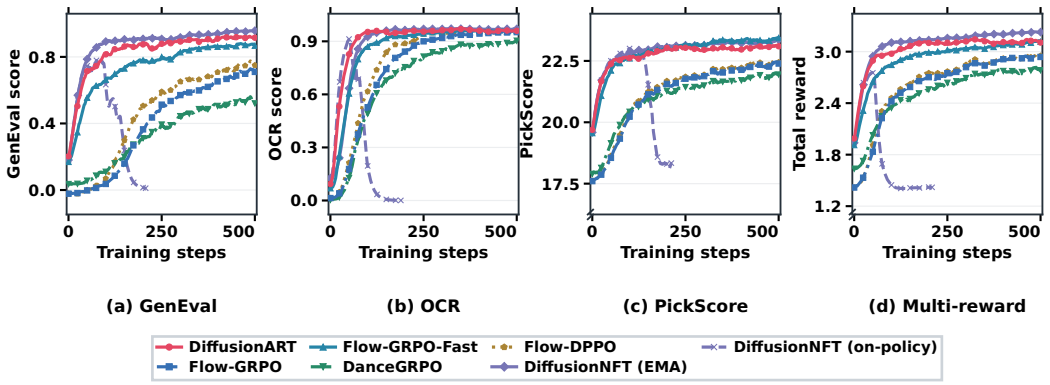

Figure 5: Single- and multi-reward training on SD3.5-M over 500 steps.

Scaling to larger models. We extend the same recipe to FLUX.1-dev (12B) (Black Forest Labs, 2024) on GenEval, using LoRA with $r = 32$ and $\alpha = 64$, and the model's guidance scale of 3.5. We use six-step rollouts. Figure 6(a) compares it with Flow-GRPO-Fast and Flow-DPPO. At step 500, DiffusionART achieves a GenEval evaluation score of **0.97**, versus **0.84** and **0.92**, respectively.

Learning from pairwise feedback. Pairwise comparisons can be easier to obtain than precise reward values for visual outputs. Treating the scalar reward as ground truth, we simulate a pairwise reward model by comparing scalar rewards within each pair: training uses only the $(0, 0.5, 1)$ preferences, while curves and evaluations report the original reward. With pairwise feedback, DiffusionART retains much of its scalar-feedback efficiency and stability, reaching a final GenEval score of **0.91**, whereas Flow-GRPO-Fast degrades significantly to **0.85** (Figure 6(b), where "ART" and "Fast" denote DiffusionART and Flow-GRPO-Fast).

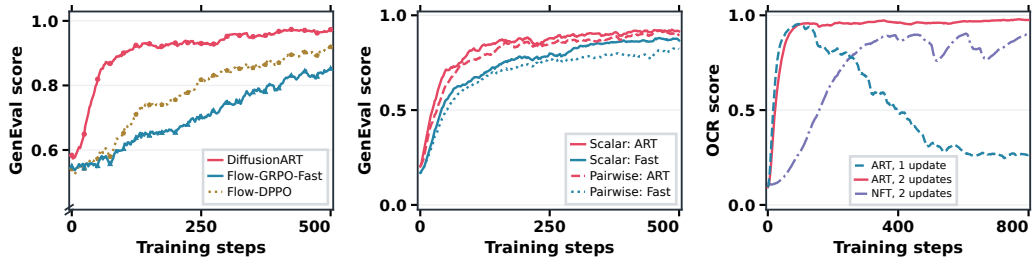

Figure 6: Scaling, feedback, and guidance ablations. (a) FLUX.1-dev on GenEval. (b) Scalar versus pairwise feedback on GenEval. (c) Guidance and restoration on OCR.

Guidance and restoration. With ART guidance, one MSE update learns quickly but then deteriorates, whereas a second update provides restoration and maintains high reward (Figure 6(c)). Two updates with NFT guidance learn more slowly and undergo larger transient drops before recovering. This guidance ablation uses the respective ART and NFT training recipes, including their normalization, masking, and data-reuse settings. Appendix B.3 discusses the single-update limitation.

5.2 Text-to-Video Generation

Setup. We fine-tune all parameters of the distilled **HunyuanVideo-13B** (Kong et al., 2024) with a learning rate of 10^{-5} and no KL regularization. Each rollout uses 32 prompts and 8 videos per prompt.

We generate 53-frame videos at 480×480 and 8 FPS with 16 sampling steps. VideoReward (Liu et al., 2025b), scoring videos at 4 FPS, provides both the training rewards and the evaluation metrics. Since the reward model is imperfect, DiffusionART and DiffusionNFT use TA/VQ/MQ weights of $0.7/0.2/0.1$ tuned to avoid reward hacking, whereas DanceGRPO and Flow-GRPO-Fast use $0/0.5/0.5$ for better visual evaluation. Appendix A.5 reports matched reward-weight comparisons.

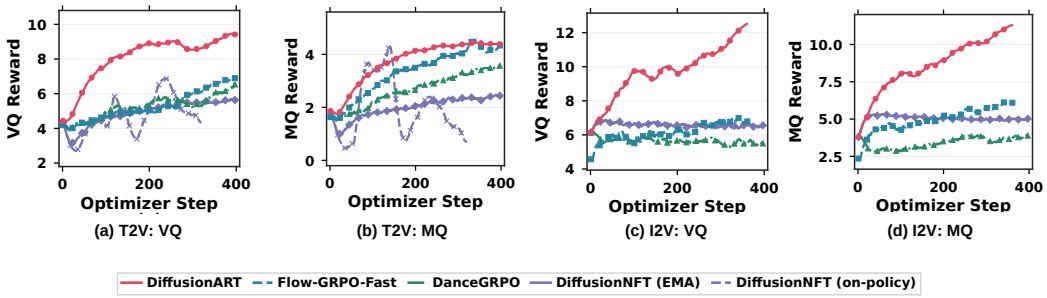

Figure 7: VQ and MQ optimization on T2V (a,b) and I2V (c,d).

Performance. Figure 7(a,b) shows a clear efficiency advantage: DiffusionART learns faster than all baselines, most markedly on VQ, and remains stable. It reaches a VQ reward of 6 with 4.9× fewer GPU hours than Flow-GRPO-Fast, while DiffusionNFT (EMA) never sustains this level. This advantage is notable because 70% of DiffusionART's training reward weight is assigned to TA, whereas the two GRPO baselines devote their full reward weight to VQ and MQ. In the final-step evaluation (Table 3(a) in Appendix A.3), DiffusionART achieves the highest VQ of 8.970 and MQ of 4.224, which are 31.4% and 31.1% higher than the best baselines, DiffusionNFT (EMA) with a VQ of 6.827 and Flow-GRPO-Fast with an MQ of 3.222.

Ablation on group size. Under a fixed optimizer-step budget, $K = 2$ achieves a final VQ of 12.235, no lower than 12.092 for $K = 8$. Since $K = 2$ uses a quarter of the rollout videos and training samples, it reaches a training VQ of 12.5 in 151.1 GPU hours, 2.8× fewer than the 422.3 needed by $K = 8$. One pair per prompt thus suffices for stable and compute-efficient pairwise optimization in this setting.

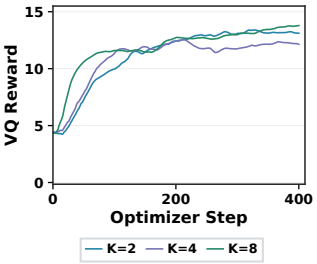

Figure 8: **Ablation on group size.** VQ versus optimizer steps for $K = 2, 4, 8$.

5.3 Image-to-Video Generation

Setup. We fine-tune all parameters of the distilled **Wan2.1-I2V-14B** (Wan Team, 2025) (LightX2V (LightX2V, 2025)). Each rollout uses 32 prompts with FLUX.1-dev reference images and 8 videos per prompt, and we generate 53-frame videos at 400×640 and 8 FPS with four sampling steps. All methods use 1:1 VQ/MQ weights; all other settings follow Section 5.2.

Performance. Figure 7(c,d) shows a clear efficiency advantage: DiffusionART keeps improving VQ and MQ, whereas the baselines improve slowly or plateau. It reaches an MQ reward of 6 with **5.6× fewer GPU hours** than Flow-GRPO-Fast. DiffusionNFT (EMA) plateaus early and stays there, which we attribute to the residual consistency force diluting the reward-improving guidance after sample-level aggregation. Table 2 in Appendix A.4 reports the final-step evaluation.

6 Conclusion

We studied why NFT-style diffusion post-training can be unstable and found that its objective can introduce an implicit forward–reverse self-consistency force that is not reliably canceled across states, while stronger restoration can improve stability at the cost of slower reward optimization. DiffusionART presents a simple but effective way to systematically cancel the unstable force by aggregating endpoint reward guidance before assigning it to reverse-rollout states, and then using a second update to provide drift-adaptive restoration for stabilization. Experiments on image and video generation show efficient, stable reward improvement, including with pairwise feedback and small rollout groups.

AI USE STATEMENT

We used generative AI tools ChatGPT, to assist with paper writing. The authors are responsible for verifying AI-assisted material and for the final scientific claims, results, and manuscript.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Black Forest Labs. FLUX.1 [dev]. https://huggingface.co/black-forest-labs/FLUX.1-dev, 2024.

Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, et al. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. *arXiv preprint arXiv:2310.11513*, 2023.

Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. TempFlow-GRPO: When timing matters for GRPO in flow models. *arXiv preprint arXiv:2508.04324*, 2025.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, et al. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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. *arXiv preprint arXiv:2305.01569*, 2023.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, et al. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

LightX2V. Wan2.1-I2V-14B-720P-StepDistill-CfgDistill-Lightx2v. https://huggingface.co/lightx2v/Wan2.1-I2V-14B-720P-StepDistill-CfgDistill-Lightx2v, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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. *arXiv preprint arXiv:2505.05470*, 2025a.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, et al. Improving video generation with human feedback. *arXiv preprint arXiv:2501.13918*, 2025b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

Bowen Ping, Xiangxin Zhou, Penghui Qi, Minnan Luo, Liefeng Bo, and Tianyu Pang. Flow-DPPO: Divergence proximal policy optimization for flow matching models. *arXiv preprint arXiv:2606.11025*, 2026.

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

Stability AI. Stable diffusion 3.5 medium. https://huggingface.co/stabilityai/stable-diffusion-3.5-medium, 2024.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

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

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5988–5998, 2026.

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

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*, 2023.

Yixian Xu, Kaiyuan Gao, Yuxiang Chen, Yilei Chen, Zecheng Tang, et al. Qwen-Image-2.0-RL technical report. *arXiv preprint arXiv:2606.27608*, 2026a.

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*, 2026b.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models, 2026. URL https://arxiv.org/abs/2509.25050.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, et al. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025.

Yuanzhi Zhu, Xi Wang, Stephane Lathuiliere, and Vicky Kalogeiton. Diffusion reinforcement learning via centered reward distillation. *arXiv preprint arXiv:2603.14128*, 2026.

A Additional Experimental Results

A.1 Scaling to FLUX

DiffusionART has the highest score at each of the three evaluation points in Table 1, with the largest gap at step 500; by step 1,000, all methods are much closer.

Table 1: **FLUX.1-dev GenEval evaluation.**

Method	Step 500	Step 800	Step 1,000
Flow-GRPO-Fast	0.8374	0.9100	0.9353
Flow-DPPO	0.9183	0.9189	0.9474
DiffusionART	**0.9684**	**0.9682**	**0.9516**

Table 2: **I2V evaluation.** VQ:MQ = 1:1.

Method	VQ ↑	MQ ↑
Flow-GRPO-Fast	5.771	4.710
DanceGRPO	5.123	3.266
DiffusionNFT (EMA)	5.516	4.161
DiffusionART	**11.196**	**10.476**

A.2 Multi-Reward Optimization for T2I

We jointly optimize GenEval, PickScore, CLIPScore, and HPSv2.1 on a single SD3.5-M policy for 500 training steps. Figure 9 reports each reward separately, comparing how the methods balance compositional accuracy, image–text alignment, and human preference under a shared multi-reward objective.

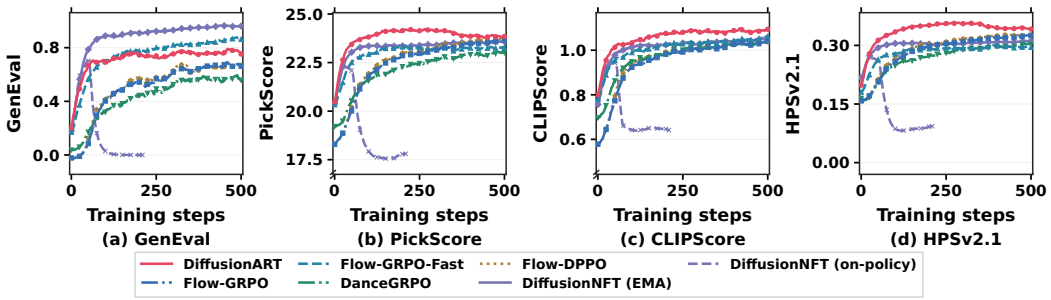

Figure 9: Training reward curves for four jointly optimized objectives on SD3.5-M over 500 steps.

A.3 Text-to-Video Evaluation

Table 3 reports the final-step T2V evaluation for the overall comparison and the pairwise-feedback experiments in Section 5.2.

Table 3: **Final-step T2V evaluation.** The VQ-only comparisons in (b) are not directly comparable to the multi-reward results in (a); concentrating the objective on VQ yields higher VQ scores.

(a) Overall performance

Method	VQ ↑	MQ ↑
Pretrained	5.195	1.943
Flow-GRPO-Fast	6.279	3.222
DanceGRPO	6.173	3.121
DiffusionNFT (EMA)	6.827	2.683
DiffusionNFT (on-policy)	5.543	1.839
DiffusionART	**8.970**	**4.224**

(b) Pairwise feedback

Method	GPU Hours ↓	VQ ↑
Scalar, $K = 8$	891.1	**13.637**
Pairwise, $K = 2$	**260.8**	12.235
Pairwise, $K = 4$	465.6	10.883
Pairwise, $K = 8$	883.3	12.092

A.4 Image-to-Video Evaluation

Table 2 compares the methods under shared VQ:MQ = 1:1 reward weights.

A.5 Matched Reward-Weight Controls for T2V

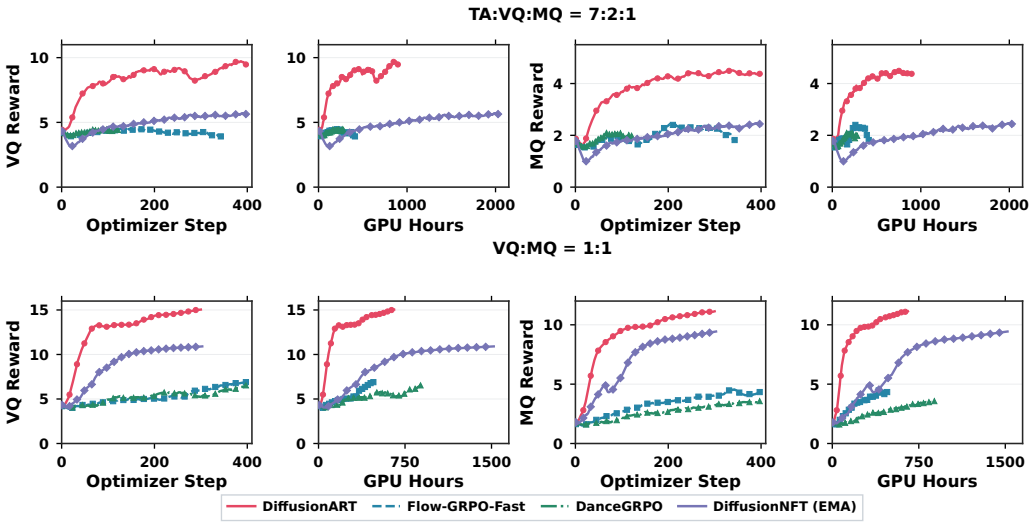

Figure 10: **Matched reward-weight controls for T2V.** Rows fix TA:VQ:MQ = 7:2:1 (top) or VQ:MQ = 1:1 (bottom). The left and right pairs show VQ and MQ against optimizer steps and GPU hours.

Figure 10 separates optimization efficiency from the reward-weight choices in the main comparison. With TA:VQ:MQ = 7:2:1 for all algorithms, DiffusionART reaches training VQ of 6 and MQ of 3 after 29 and 46 updates, respectively; none of the matched baselines reaches these thresholds within its recorded trajectory. DiffusionART maintains its gains through 400 updates. With VQ:MQ = 1:1, DiffusionART reaches the same VQ of 6 in **2.5× fewer optimizer steps than DiffusionNFT (EMA)** (20 versus 50) and **15.0× fewer than Flow-GRPO-Fast** (20 versus 299). The corresponding GPU-hour speedups are **5.7× and 8.2×**, respectively: DiffusionART uses 44.2 GPU hours, compared with 250.2 for NFT and 364.5 for Fast. DanceGRPO reaches this level after 349 updates and 782.1 GPU hours.

The advantage also holds for motion quality. To reach MQ of 3, DiffusionART needs 19 updates and 42.9 GPU hours, compared with 33 updates and 165.6 GPU hours for NFT, and 118 updates and 150.1 GPU hours for Flow-GRPO-Fast. This gives **1.7×/6.2× step speedups** and **3.9×/3.5× GPU-hour speedups** over NFT/Fast. Thus, the efficiency gain becomes larger when the reward weights are held fixed and fair.

Optimizing VQ and MQ more aggressively can reduce text alignment due to the flaw of reward model, and we observe that this happens for each baseline. We therefore choose the main-section reward weights by inspecting generated videos, rather than maximizing VQ and MQ alone. Assigning 70% of DiffusionART's reward weight to TA makes its main-section VQ/MQ efficiency comparison more conservative than the matched 1:1 comparison.

Direct optimization with a differentiable reward. We originally follows the undifferentiable reward model setting (i.e. we can only access the scaler reward value, not including the gradient of reward model). However, methods like ReFL (Xu et al., 2023) utilize the gradient of a VLM reward model to help training. We also compare DiffusionART with ReFL under matched VQ:MQ = 1:1 weights (Figure 11). ReFL uses 32 prompts and one video per prompt ($K = 1$), with gradients through the frozen reward model and VAE to the final denoising step; earlier steps are detached. It is trained for 400 updates at a learning rate of 10^{-5}, without KL regularization or EMA.

DiffusionART improves both rewards with fewer optimizer updates, while ReFL uses fewer samples per update ($K = 1$ versus $K = 8$ of DiffusionART) but requires reward-model backpropagation; DiffusionART uses reward values alone.

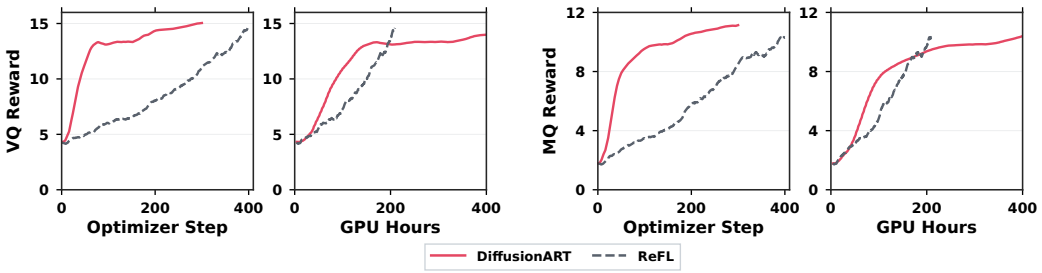

Figure 11: **DiffusionART versus ReFL with matched VQ:MQ** = 1:1 **weights.** The left pair shows VQ and the right pair MQ, each versus optimizer steps and GPU hours. GPU-hours panels show the first 400 hours.

B Proofs

B.1 Conditional Covariance Identity for Ideal NFT

Lemma 1 (restated). At a fixed supported state s_t, let Q_{old} denote Q_b with $b = \text{old}$, i.e., the joint law obtained by forward noising the rollout endpoints $x_0 \sim p_{\text{old}}(\cdot \mid c)$, and let $q := \mathbb{E}_{Q_{\text{old}}}[r(x_0, c) \mid s_t]$ and $g := \text{Cov}_{Q_{\text{old}}}[r(x_0, c), u_t \mid s_t]$. If the tilted distributions and moments exist and $0 < q < 1$, then

$$q\big(v^{+}_{\text{fwd}} - v^{\text{old}}_{\text{fwd}}\big) = (1-q)\big(v^{\text{old}}_{\text{fwd}} - v^{-}_{\text{fwd}}\big) = g,$$

where all fields are evaluated at s_t. Hence the positive and negative branches point along the same reward-tilt direction g.

Proof. Let $Z_+(c) = \mathbb{E}_{P_{\text{old}}}[r \mid c] > 0$ and $Z_-(c) = \mathbb{E}_{P_{\text{old}}}[1 - r \mid c] > 0$ normalize P_+ and P_-. Since $P_\pm$ reweight only x_0, while ϵ is independent of x_0 and (x_t, u_t) are functions of (x_0, ϵ), the joint laws of $(x_0, \epsilon, x_t, u_t)$ satisfy

$$Q_+(\cdot \mid t, c) = \frac{r(x_0, c)}{Z_+(c)} Q_{\text{old}}(\cdot \mid t, c), \tag{7}$$

$$Q_-(\cdot \mid t, c) = \frac{1 - r(x_0, c)}{Z_-(c)} Q_{\text{old}}(\cdot \mid t, c). \tag{8}$$

Marginalizing onto x_t and using $q = \mathbb{E}_{Q_{\text{old}}}[r(x_0, c) \mid s_t]$ gives

$$Q_+(x_t \mid t, c) = \frac{q}{Z_+(c)} Q_{\text{old}}(x_t \mid t, c), \tag{9}$$

$$Q_-(x_t \mid t, c) = \frac{1 - q}{Z_-(c)} Q_{\text{old}}(x_t \mid t, c). \tag{10}$$

Bayes' rule therefore yields

$$Q_+(\cdot \mid s_t) = \frac{r(x_0, c)}{q} Q_{\text{old}}(\cdot \mid s_t), \tag{11}$$

$$Q_-(\cdot \mid s_t) = \frac{1 - r(x_0, c)}{1 - q} Q_{\text{old}}(\cdot \mid s_t). \tag{12}$$

Taking conditional expectations of u_t gives

$$q v^{+}_{\text{fwd}}(s_t) = \mathbb{E}_{Q_{\text{old}}}[r u_t \mid s_t], \tag{13}$$

$$(1-q) v^{-}_{\text{fwd}}(s_t) = \mathbb{E}_{Q_{\text{old}}}[(1-r) u_t \mid s_t]. \tag{14}$$

Since $v^{\text{old}}_{\text{fwd}}(s_t) = \mathbb{E}_{Q_{\text{old}}}[u_t \mid s_t]$,

$$q\big(v^{+}_{\text{fwd}}(s_t) - v^{\text{old}}_{\text{fwd}}(s_t)\big) = \mathbb{E}_{Q_{\text{old}}}[r u_t \mid s_t] - q\mathbb{E}_{Q_{\text{old}}}[u_t \mid s_t] = g, \tag{15}$$

$$(1-q)\big(v^{\text{old}}_{\text{fwd}}(s_t) - v^{-}_{\text{fwd}}(s_t)\big) = (1-q)\mathbb{E}_{Q_{\text{old}}}[u_t \mid s_t] - \mathbb{E}_{Q_{\text{old}}}[(1-r) u_t \mid s_t] = g. \tag{16}$$

This proves both the collinearity and the conditional covariance identity.

B.2 NFT Force and Restoration Decompositions

B.2.1 Reward Guidance and Self-Consistency Force

Proposition 1 (restated). Let $\mathcal{L}_{\mathrm{NFT}}(s_t)$ denote Equation 1 conditioned on s_t, q and g follow Lemma 1, and $e = v^{\mathrm{old}}_{\mathrm{fwd}} - v^{\mathrm{old}}_{\mathrm{rvs}}$. At the pure on-policy update point $v_\theta = v^{\mathrm{old}}_{\mathrm{rvs}}$ and for $\beta > 0$, its normalized output-space force is

$$\mathcal{F}_{\mathrm{NFT}}(s_t) := -\frac{1}{2\beta}\,\nabla_{v_\theta(s_t)}\mathcal{L}_{\mathrm{NFT}}(s_t)\big|_{v_\theta = v^{\mathrm{old}}_{\mathrm{rvs}}} = 2g + (2q-1)e.$$

Proof. Fix s_t, and write $d = v_\theta(s_t) - v^{\mathrm{old}}_{\mathrm{rvs}}(s_t)$ and $h = u_t - v^{\mathrm{old}}_{\mathrm{rvs}}(s_t)$. The conditional objective is

$$\mathcal{L}_{\mathrm{NFT}}(s_t) = \mathbb{E}_{Q_{\mathrm{old}}}\big[r\|\beta d - h\|_2^2 + (1-r)\|-\beta d - h\|_2^2 \mid s_t\big] \tag{17}$$

$$= \beta^2\|d\|_2^2 - 2\beta\,\langle d, \mathbb{E}_{Q_{\mathrm{old}}}[(2r-1)h \mid s_t]\rangle + \mathbb{E}_{Q_{\mathrm{old}}}[\|h\|_2^2 \mid s_t]. \tag{18}$$

At the pure on-policy update point, $d = 0$. Therefore,

$$\mathcal{F}_{\mathrm{NFT}}(s_t) = -\frac{1}{2\beta}\,\nabla_d\mathcal{L}_{\mathrm{NFT}}(s_t)|_{d=0} \tag{19}$$

$$= \mathbb{E}_{Q_{\mathrm{old}}}[(2r-1)h \mid s_t]. \tag{20}$$

Because $\mathbb{E}_{Q_{\mathrm{old}}}[h \mid s_t] = v^{\mathrm{old}}_{\mathrm{fwd}} - v^{\mathrm{old}}_{\mathrm{rvs}} = e$ and $\mathrm{Cov}_{Q_{\mathrm{old}}}(r, h \mid s_t) = \mathrm{Cov}_{Q_{\mathrm{old}}}(r, u_t \mid s_t) = g$,

$$\mathbb{E}_{Q_{\mathrm{old}}}[(2r-1)h \mid s_t] = 2\,\mathrm{Cov}_{Q_{\mathrm{old}}}(r, h \mid s_t) + (2\mathbb{E}_{Q_{\mathrm{old}}}[r \mid s_t] - 1)\mathbb{E}_{Q_{\mathrm{old}}}[h \mid s_t] \tag{21}$$

$$= 2g + (2q-1)e, \tag{22}$$

which proves Equation 3.

B.2.2 Guidance and Quadratic Restoration

Proposition 2 (restated). Let $d = v_\theta - v_{\mathrm{old}}$ and $h = u_t - v_{\mathrm{old}}$.

$$\mathcal{L}_{\mathrm{NFT}} = \mathbb{E}_{\mathcal{D}_{\mathrm{old}}}\big[-2\beta(2r-1)\langle d, h\rangle + \beta^2\|d\|_2^2\big] + \mathrm{const},$$

where const is independent of the trainable parameters.

Proof. Using the same d and h, the positive and negative residuals are $\beta d - h$ and $-\beta d - h$, respectively. Expanding their weighted squared norms gives

$$r\|\beta d - h\|_2^2 + (1-r)\|-\beta d - h\|_2^2 = \beta^2\|d\|_2^2 - 2\beta(2r-1)\langle d, h\rangle + \|h\|_2^2. \tag{23}$$

Taking $\mathbb{E}_{\mathcal{D}_{\mathrm{old}}}$ and absorbing the final term into the constant proves Equation 4.

B.3 Covariance as the First-Order Reward Tilt

Lemma 2 (restated). If the relevant moments exist and differentiation can be exchanged with expectation near $\tau = 0$, then $\frac{\partial}{\partial\tau}\mathbb{E}_{p_\tau}[x_0 \mid c]\big|_{\tau=0} = \mathrm{Cov}_{p_{\mathrm{old}}}(R, x_0 \mid c)$.

Proof. Fix c and suppress it below. Write $M(\tau) = \mathbb{E}_{p_{\mathrm{old}}}[x_0 e^{\tau R}]$ and $Z(\tau) = \mathbb{E}_{p_{\mathrm{old}}}[e^{\tau R}]$. The normalized tilted density is $p_\tau(x_0) = p_{\mathrm{old}}(x_0)e^{\tau R(x_0)}/Z(\tau)$, so its endpoint mean is $M(\tau)/Z(\tau)$. Assume differentiation under these expectations is valid; a sufficient condition is $\mathbb{E}[(1+\|x_0\|)(1+|R|)e^{a|R|}] < \infty$ for some $a > 0$. Since $Z(0) = 1$, the quotient rule gives

$$\frac{\partial}{\partial\tau}\frac{M(\tau)}{Z(\tau)}\bigg|_0 = \mathbb{E}[x_0 R] - \mathbb{E}[x_0]\mathbb{E}[R] = \mathrm{Cov}(R, x_0). \tag{24}$$

From a distribution identity to practical guidance. Lemma 2 describes the first-order change of an endpoint mean. Likewise, the prompt-level covariance does not equal the state-conditional posterior covariance in Lemma 1 (i.e., it is biased). This is a trade-off for systematically omitting the self-consistency force while maintaining the simplicity and scalability of the algorithm. Moreover, the adaptive restoration mechanism can cancel the negative effects of bias. This could explain the result in Figure 6(c) that a single MSE update results in score degradation in late training, while two MSE updates avoid this drop and enable long-term stability of DiffusionART. We reuse the K endpoints already sampled per prompt to obtain a shared, scalable approximation rather than sample additional endpoints conditioned on each training state.

C VISUALIZATION

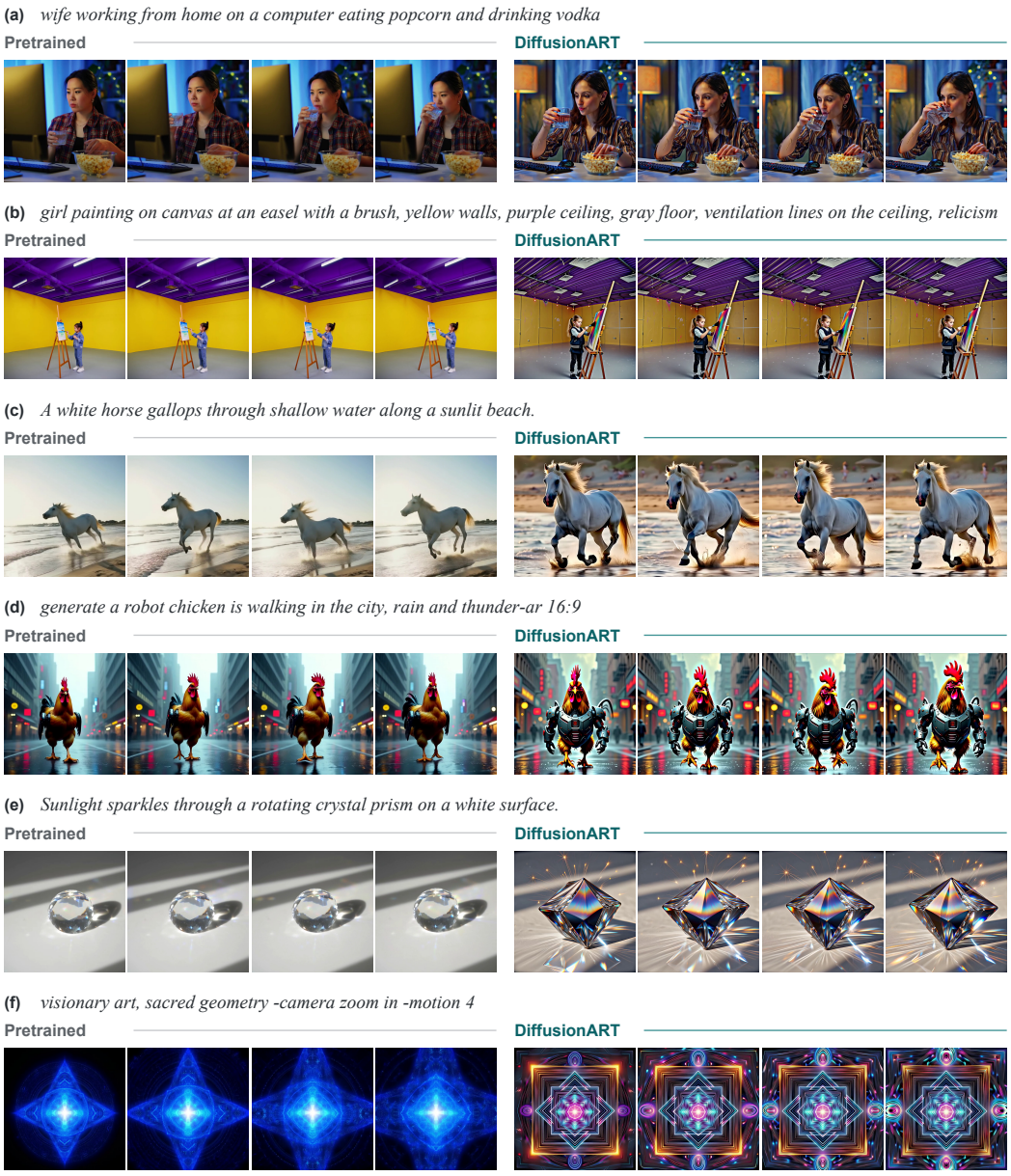

Figure 12: **Text-to-video generation before and after DiffusionART.** Each pair uses the same prompt, seed, and sampling settings. Four chronologically ordered frames are shown at identical frame positions within each pair: pretrained outputs on the left and DiffusionART outputs on the right.

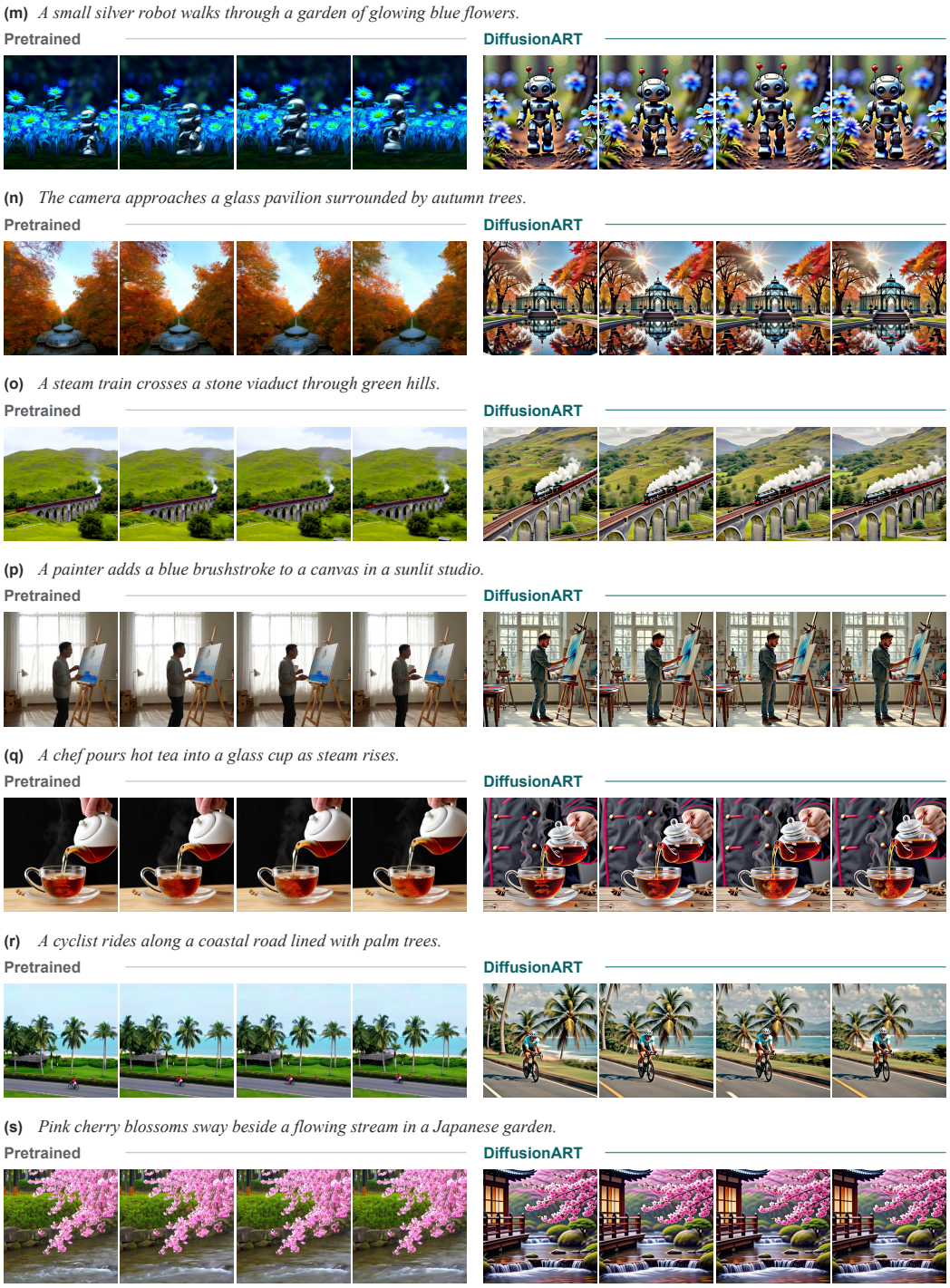

Figure 13: **Text-to-video comparisons across scenes and activities.** Each row pairs pretrained and DiffusionART outputs at the same four frame positions.

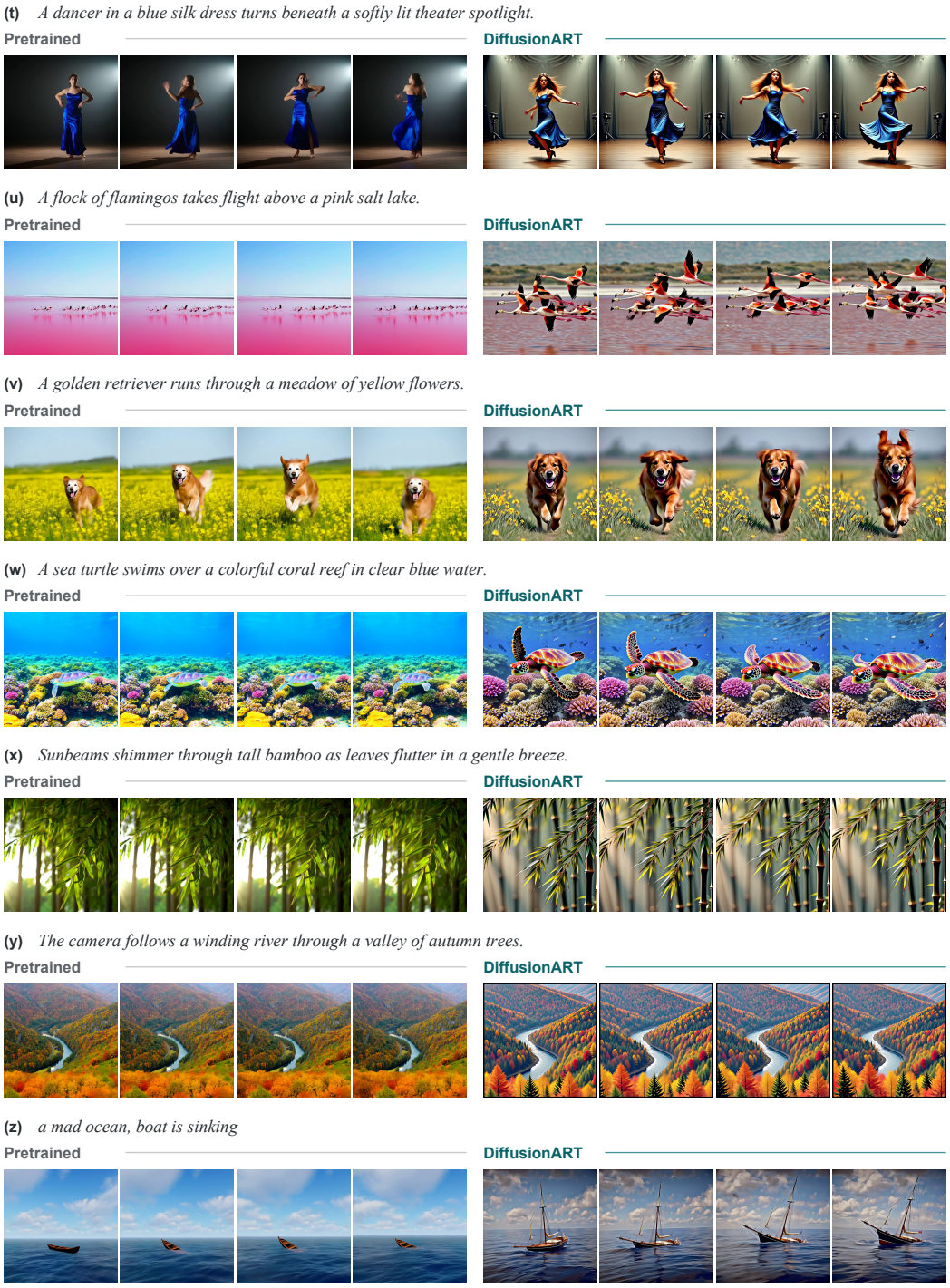

Figure 14: **Text-to-video comparisons of motion and natural scenes.** Prompt, seed, and sampling settings are matched within each pair.

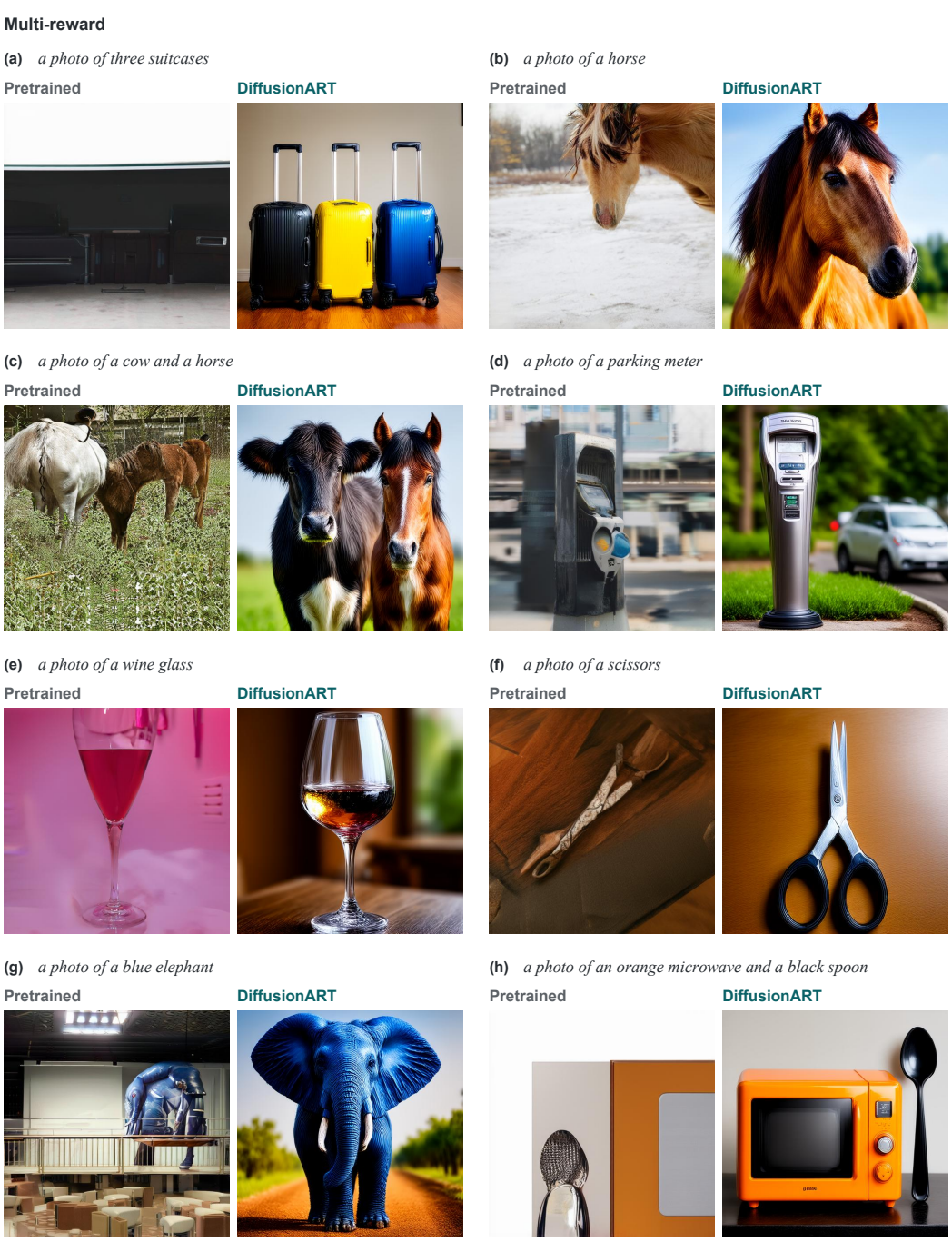

Figure 15: **Text-to-image generation with multi-reward optimization.** Selected SD3.5-M outputs before and after DiffusionART, jointly optimized with GenEval, CLIPScore, PickScore, and HPSv2. Each pair shows the same prompt, with pretrained outputs on the left and DiffusionART outputs on the right.

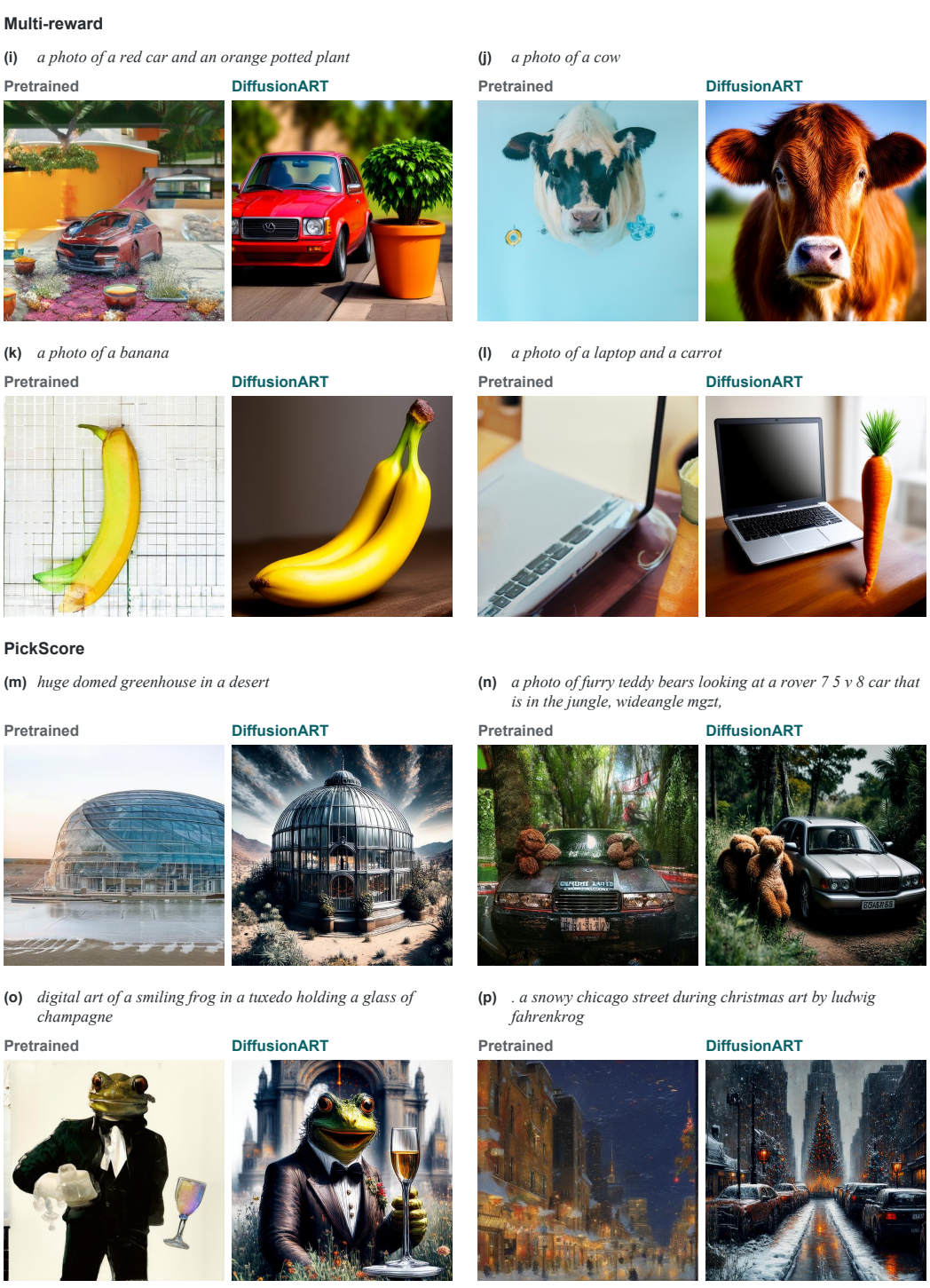

Figure 16: **Additional text-to-image comparisons.** The top two rows use multi-reward optimization; the bottom two use PickScore. Each pair shows pretrained and DiffusionART outputs under the same prompt.

D CLARIFICATION OF METHODS AND EXPERIMENTS

D.1 REVERSE-PROCESS POLICY OPTIMIZATION

FlowGRPO treats denoising as a finite-horizon policy. For each prompt, it samples K stochastic reverse trajectories, group-normalizes their terminal rewards, and applies clipped PPO to every transition (Liu et al., 2025a). This directly supervises sampled states but requires tractable transition densities and stored trajectories.

For reverse trajectory $x_{T:0}^{(i)} \sim P_{\text{old}}^{\text{rev}}(\cdot \mid c)$, FlowGRPO assigns the group-normalized terminal advantage A_i to every transition and maximizes

$$\begin{aligned}\mathcal{J}_{\text{FG}} = \mathbb{E}_{i,k}\Big[&\min\big(\rho_{i,k}A_i,\ \text{clip}(\rho_{i,k}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})A_i\big) \\ &- \lambda_{\text{KL}} D_{\text{KL}}\big(p_\theta(\cdot \mid x_k^{(i)}, c) \,\|\, p_{\text{ref}}(\cdot \mid x_k^{(i)}, c)\big)\Big].\end{aligned} \tag{25}$$

where $\rho_{i,k} = p_\theta(x_{k-1}^{(i)} \mid x_k^{(i)}, c)/p_{\text{old}}(x_{k-1}^{(i)} \mid x_k^{(i)}, c)$ and $\mathbb{E}_{i,k}$ also averages over prompts and old-policy trajectories. Here $\epsilon_{\text{clip}} > 0$ is the PPO clip radius, $\lambda_{\text{KL}} \geq 0$ weights the penalty, and p_{ref} is a fixed reference transition. This requires the stochastic reverse transition used for rollout to expose a tractable density.

D.2 MATCHED TWO-UPDATE LINEAR CONTROL

The two-linear-update control replays the same rollout batch twice and optimizes only the detached linear term

$$\mathcal{L}_{\text{linear}} = -2\beta\mathbb{E}_{\mathcal{D}_{\text{old}}}\left[(2r-1)\langle v_\theta - v_{\text{old}}, u_t - v_{\text{old}}\rangle\right], \tag{26}$$

where r, u_t, and v_{old} are detached. It therefore matches the data reuse and guidance of two-update MSE training while removing the quadratic restoration term $\beta^2\|v_\theta - v_{\text{old}}\|_2^2$. This three-arm diagnostic uses an NFT-compatible AdvBridge decomposition to isolate restoration; it is a mechanism control rather than an exact Mirror-NFT reproduction.

D.3 TIMESTEP WEIGHTING AND CORRECTIVE MASKING

Following the corrective masking principle of Flow-DPPO (Ping et al., 2026), we suppress updates that further increase an already excessive output displacement, while retaining corrective updates. Let $d(s_t) = v_\theta(s_t) - v_\theta^{(0)}(s_t)$ denote the displacement from the old policy after first update, and let $m(s_t)$ measure its magnitude as defined below. For states with nonzero guidance, we use

$$w(s_t) = t\left[1 - \mathbf{1}\{m(s_t) \geq \delta\}\,\mathbf{1}\{\langle d(s_t), g(s_t) - d(s_t)\rangle > 0\}\right]. \tag{27}$$

The factor t emphasizes high-noise states. Since $g - d$ is the MSE descent direction in output space, the inner-product test identifies directions that increase the displacement.

We measure displacement using the mean and variance of a reference Flow-SDE transition. This reference defines only the masking metric; rollouts remain deterministic ODE trajectories. With reverse Euler interval $\Delta t < 0$, reference noise coefficient $\nu = 0.8$, and threshold $\delta = 10^{-5}$, the score is

$$m(s_t) = \frac{\text{RMS}(d(s_t))^2}{2\nu^2}\begin{cases}(\Delta t)^2, & t = 1, \\ (1+\nu^2/2)^2\dfrac{1-t}{t}(-\Delta t), & 0 < t < 1.\end{cases} \tag{28}$$

We recompute w at each update and treat it as constant during backpropagation. Masked states contribute zero to the original batch mean.

D.4 PAIRWISE FEEDBACK: IDENTITY AND IMPLEMENTATION

Pairwise covariance identity. For K endpoints forming $M = K/2$ disjoint pairs, with pairwise labels $y_i \in \{0, \frac{1}{2}, 1\}$ replacing R_i in Equation 5, use population reduction $1/K$ globally and $1/2$ within each pair. The global covariance equals the average of the M pair covariances: $\overline{\text{Cov}} =$

$(2K)^{-1}\sum_{\substack{j=1\\ j \text{ non-tied}}}^{M}(x_{0,j}^{\text{win}} - x_{0,j}^{\text{lose}})$. Equivalently, the coefficient is $1/(4M)$. Tied pairs contribute zero.

Proof. Every pair has label mean $1/2$, including ties, so the global label mean is also $1/2$. Thus centering endpoints changes no cross term:

$$\overline{\text{Cov}} = \frac{1}{K}\sum_{i=1}^{K}(y_i - \tfrac{1}{2})(x_0^{(i)} - \bar{x}_{0,c}) = \frac{1}{K}\sum_{i=1}^{K}(y_i - \tfrac{1}{2})x_0^{(i)} \tag{29}$$

$$= \frac{1}{2K}\sum_{\substack{j=1\\ j \text{ non-tied}}}^{M}\left(x_{0,j}^{\text{win}} - x_{0,j}^{\text{lose}}\right). \tag{30}$$

A non-tied pair's covariance, computed with denominator two, is one quarter of its winner-minus-loser difference. Averaging those pair covariances gives the same expression; tied pairs contribute zero. The pairwise labels enter the same covariance operator as scalar rewards, establishing the stated specialization. The coefficient $\frac{1}{2K}$ doesn't matter since the covariance will be normalized to form the guidance.

Implementation.

Algorithm 2 DiffusionART-Pairwise: one training iteration

1: Sample with an ODE solver using $v_\theta^{(0)} = v_\theta$. **Rollout Stage**
   For each prompt c, collect an even number K of endpoints $x_0^{(i)}$, states s_t, and predictions $v_\theta^{(0)}(s_t)$.
2: Form $M = K/2$ disjoint pairs per prompt; query the pairwise reward model.
3: Assign pairwise rewards $y_i = 1$ to winners, $y_i = 0$ to losers, and $y_i = \frac{1}{2}$ for ties.
4: Compute covariance guidance by Equation 5, replacing scalar rewards R_i with pairwise rewards y_i.
5: Freeze $v_{\text{tgt}}(s_t) = v_\theta^{(0)}(s_t) + g(s_t)$ for each state.

6: Split the samples into two equal-sized disjoint minibatches B_1, B_2. **Training Stage**
7: For each $B \in \{B_1, B_2\}$, recompute $w(s_t)$ and update θ with one step on
   $\mathcal{L}(B;\theta) = \frac{1}{|B|}\sum_{s_t \in B} w(s_t)\,\text{MSE}(v_\theta(s_t), v_{\text{tgt}}(s_t))$.

Additional Experiments. We also compare prompt-global guidance, shared across all trajectories of a prompt, with pair-local guidance, applied to the two trajectories within each pair. Both use 24 trajectories per prompt, forming 12 disjoint pairs. At step 400, prompt-global guidance reaches a held-out GenEval score of **0.91**, versus **0.78** for pair-local guidance. Figure 17 shows their training curves. Pooling endpoints may reduce estimation variance and yield more reliable guidance.

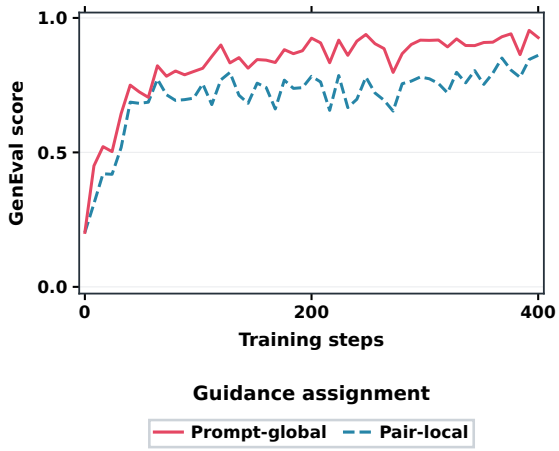

Figure 17: Prompt-global versus pair-local pairwise guidance on GenEval.