ROUTING POLICY OPTIMIZATION FOR ONLINE DIFFUSION REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning can align visual generators with reward signals beyond their pretraining objectives, but existing score-function methods typically introduce stochasticity by perturbing the reverse denoising trajectory, leading to costly rollout-based optimization. We instead ask whether stochastic exploration can be introduced inside the generator while preserving its original sampler. *Routing Policy Optimization* (RPO) answers this question by formulating attention routing as a discrete policy. At a selected denoising level, each query block stochastically attends to a subset of candidate key blocks, with Bernoulli probabilities derived from the generator's native query–key affinities. The resulting routing masks define discrete actions that can be optimized with a score-function objective. RPO further uses a controlled two-rollout estimator: two rollouts share the same prompt, initial latent, routed denoising level, and routed layers, differing only in their sampled attention routes. Their reward difference provides an advantage signal for the routing actions. Because the routing policy reuses the generator's native attention parameters, the learned updates remain effective after routing is removed. At inference, RPO restores dense attention and retains the original deterministic sampler. Experiments on text-to-image and text-to-video generation show that RPO improves generation quality while achieving nearly $4\times$ higher training efficiency than Flow-GRPO-style methods. More details and results are available at `https://anonymous.4open.science/r/RoutePO-submission`.

1 INTRODUCTION

Diffusion and flow models now generate high-quality images and videos from natural-language prompts. DDPM establishes stochastic reverse denoising, while DDIM exposes a deterministic sampling path under the same training objective (Ho et al., 2020; Song et al., 2021). Flow matching and rectified flow instead learn vector fields for continuous transport (Lipman et al., 2023; Liu et al., 2023; Esser et al., 2024). In parallel, latent diffusion, Diffusion Transformers, and video diffusion models make high-resolution visual generation tractable at scale (Rombach et al., 2021; Peebles & Xie, 2022; Yang et al., 2024; Wan Team et al., 2025). Learned preference models then expose alignment objectives beyond likelihood and reconstruction (Xu et al., 2023; Kirstain et al., 2023; Ma et al., 2025).

Online reinforcement learning can optimize these objectives, but its central bottleneck is the policy. DDPO and DPOK treat stochastic denoising transitions as actions (Black et al., 2024; Fan et al., 2023). Flow-GRPO and DanceGRPO extend this construction to deterministic flow models by converting their ODE samplers into stochastic SDEs (Liu et al., 2025a; Xue et al., 2025b). Temporal windows, shared branches, and one-step updates reduce the resulting rollout cost (Li et al., 2025a; He et al., 2025; Li et al., 2025b; He et al., 2026), while dense credit and regulated clipping improve how terminal rewards shape sampler transitions (Deng et al., 2026; Shen et al., 2026; Yang et al., 2026; Wang et al., 2025a). Forward-matching objectives avoid reverse-trajectory likelihoods, but they also remove the explicit sampled action (Zheng et al., 2026; Luo et al., 2026; Xue et al., 2025a; Xu et al., 2026c; Choi et al., 2026). What remains missing is an explicit, tractable policy that preserves the original deterministic sampler for both image and video generators.

Our key insight is that stochastic exploration need not perturb the denoising trajectory. A diffusion transformer splits each latent into visual tokens, and its attention layers already compute query-key

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.

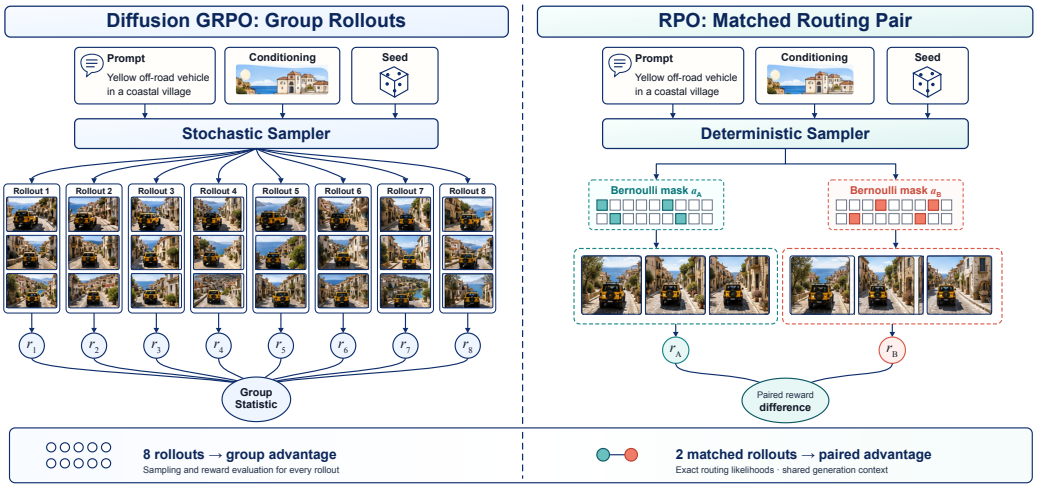


Figure 1: **A controlled routing pair replaces a large sampler-noise group.** Sampler-space GRPO injects continuous noise into the reverse process and estimates an advantage from many complete visual rollouts. RPO retains the original deterministic sampler and pairs two rollouts that share the prompt, initial visual latent, routed denoising level, and routed layers. They differ only in their Bernoulli attention masks, so their reward difference measures the aggregate effect of the two recorded routing actions. Generated visual outputs are schematic.

affinities between them. If we group neighboring tokens into blocks, each query block can attend to a sampled subset of key blocks instead of all of them as dense attention does. We call this choice of visible key blocks a *route*. Sampling routes exposes a discrete action without adding a policy head or modifying the sampler. We introduce *Routing Policy Optimization* (RPO), which samples an affinity-conditioned Bernoulli attention mask at one denoising level. The conditional likelihood of the recorded mask history supplies a score-function surrogate for the native attention parameters that produce the route probabilities.

The routing action also enables a controlled two-rollout estimator, as Figure 1 illustrates. Both members share every non-routing source of variation and differ only in their sampled mask histories. Their signed reward difference therefore provides a leave-one-out advantage for the two routing actions. During replay, later probabilities condition on earlier recorded masks, which follows the generator's forward execution order. A fixed logit offset sets the initial sparsity operating point, while a nominal keep budget normalizes the loss. Realized keep counts remain stochastic and can evolve with the attention affinities.

Routing serves only as a training interface. Its likelihood updates native query and key projections together with trainable upstream parameters that shape later route scores. These parameters also participate in ordinary dense attention. After training, we remove all masks and evaluate the updated generator with dense attention and its original deterministic sampler. The central empirical question is whether reward learned through temporary routing transfers to the dense image or video generator used at inference.

We evaluate this question on Stable Diffusion 3.5 Medium and Wan2.2-TI2V-5B (Wan Team et al., 2025), covering text-to-image and text-to-video generation. Across both settings, RPO improves reward-aligned generation quality while retaining dense inference. On Wan2.2, the dense evaluation gain reaches about $+3$ by step 15 and remains near that level through step 50. RPO also attains the strongest overall VBench score among the evaluated checkpoints while using 24.1 GPU-hours, compared with 75.4 for Flow-GRPO and 102.3 for DanceGRPO. This corresponds to $3.1\times$ and $4.2\times$ lower training cost, respectively. Our contributions can be summarized as following:

- We introduce a sampler-preserving formulation for online alignment of continuous visual generators. Its policy action is a training-time Bernoulli mask over native attention affinities, with no auxiliary policy head or SDE conversion.
- We formulate executed routes as a conditional mask-history policy and optimize them with a controlled two-rollout surrogate that holds the non-routing generation context fixed.

- We couple the routing policy to parameters reused by dense attention and remove routing at inference, preserving the original image or video sampler.
- We validate RPO on Stable Diffusion 3.5 and Wan2.2, demonstrating improved reward-aligned quality across image and video generation together with substantially lower training cost than sampler-space baselines.

2 Related work

Reward alignment methods for visual generators differ mainly in where the learning signal enters the generator. We organize prior work along this axis, from the sampler trajectory to the attention computation.

Sampler-space policy gradients. DDPO and DPOK first treated stochastic denoising transitions as policy actions (Black et al., 2024; Fan et al., 2023). Flow-GRPO and DanceGRPO carried this view to flow matching and video by converting the deterministic sampler into an SDE and estimating group-relative advantages (Liu et al., 2025a; Xue et al., 2025b). Follow-up work lowers the cost of these stochastic trajectories. Temporal windows, shared-prefix branches, and one-step updates reduce the number or cost of stochastic transitions (Li et al., 2025a;b; He et al., 2026). TempFlow-GRPO concentrates stochasticity at designated branch points for temporal credit assignment (He et al., 2025), and TAGRPO aligns trajectories that start from identical initial noise (Wang et al., 2026). Dense temporal and spatial credit or regulated clipping further improve how a terminal reward shapes each transition (Deng et al., 2026; Shen et al., 2026; Yang et al., 2026; Wang et al., 2025a). These methods keep the action inside the sampler, so exploration still perturbs the reverse trajectory.

Objectives without sampler actions. A second family avoids sampler actions. Direct reward fine-tuning backpropagates through differentiable image or video rewards (Clark et al., 2023; Prabhudesai et al., 2024). For black-box rewards, DiffusionNFT, DGPO, and advantage-weighted matching replace reverse-trajectory likelihoods with forward-process or preference objectives (Zheng et al., 2026; Luo et al., 2026; Xue et al., 2025a). Adjoint matching, ELBO surrogates, path-space value gradients, and velocity matching pursue the same goal through native regression targets (Bergmeister et al., 2026; Tang et al., 2026; Xu et al., 2026c; Choi et al., 2026). For few-step generators, a hybrid scheme combines distribution matching, GRPO on stochastic intermediate transitions, and reward backpropagation through the final step (Huang et al., 2026). Offline preference optimization removes online rollouts altogether, ranging from Diffusion-DPO to multidimensional and temporally dense video preferences (Wallace et al., 2024; Liu et al., 2024; 2025b; Wu et al., 2025b). These objectives leave inference untouched, but they give up an explicit sampled action. In contrast, RPO keeps a score-function policy with a recorded action and places that action inside attention.

Discrete actions and attention routing. Discrete policies already exist in visual generation. MaskGRPO, Mask-GRPO, and MaskFocus optimize token corruption, unmasking, or mask schedules in natively discrete image generators (Ma et al., 2026; Luo et al., 2025; Zhang et al., 2025a). Structured attention sparsity is also well studied, although mainly for efficiency. Q/K-derived block gates and training-free compression expose attention redundancy (Gao et al., 2024; Yuan et al., 2024; Xi et al., 2025; Chen et al., 2025). VSA, trainable dynamic routing, and hierarchical sparse attention learn such sparsity inside image and video generators (Zhang et al., 2025b; Zhao et al., 2025; Zhou et al., 2025; Lv et al., 2026; Xu et al., 2026a). Reinforcement learning has also selected sparse-attention levels or recomputation schedules for acceleration (Zhao et al., 2026; Aliev et al., 2026). KVPO comes closest to alignment. It explores through stochastic historical-key routing in autoregressive video, but it scores branches with a velocity-energy surrogate (Zhang et al., 2026). Other work optimizes attention allocation for multimodal reasoning or routes preference credit through diffusion attention (Li et al., 2026; Xu et al., 2026b). Dropout-GRPO samples Bernoulli masks to diversify deterministic latent reasoning (Jung, 2026). Our paired estimator also builds on common random numbers and group-relative baselines (Asmussen & Glynn, 2007; Shao et al., 2024), and 2-GRPO shows that two language-model rollouts can retain most of a larger group's performance (Wu et al., 2025a). RPO combines these ingredients through a new interface. Its action is a Bernoulli mask over native query-key affinities, its likelihood updates parameters shared with dense attention, and the mask disappears before inference.

3 PRELIMINARIES

RPO fine-tunes a pretrained visual generator that already has a deterministic sampler. This section fixes the notation for the sampler and the reward, introduces attention over token blocks, and defines where RPO routes attention.

Denoising trajectory and reward. Let c denote a text prompt, and let N denote both the number of denoising transitions and the largest noise level. Rollout i starts from Gaussian noise and follows the deterministic sampler:

$$\tau_i = (x_i^N, x_i^{N-1}, \ldots, x_i^0), \qquad x_i^N \sim \mathcal{N}(0, I), \qquad y_i = \mathrm{Dec}(x_i^0). \tag{1}$$

Here τ_i is the reverse trajectory, and the noise level $t \in \{N, \ldots, 0\}$ decreases during generation. The frozen decoder Dec maps the terminal latent x_i^0 to the visual output y_i, which is an image or a video. A black-box reward model R_d then scores the output on reward dimension d, giving $r_{i,d} = R_d(y_i, c)$. Training uses these reward values but never differentiates through R_d.

Attention over token blocks. The generator is a diffusion transformer that splits each latent into M visual tokens, one per spatial patch of an image or spatiotemporal patch of a video. In every self-attention layer and head, each token emits a query, a key, and a value vector. The attention weights of a query are a softmax over its scaled inner products with the keys, which we call query-key affinities. Dense attention lets every query attend to every key. Block-sparse attention coarsens this pattern. It arranges the tokens in a sequence, cuts the sequence into Q contiguous *token blocks* of B tokens, and applies a binary block mask $\mu \in \{0,1\}^{Q \times Q}$. A query token in block q attends only to the key tokens of blocks j with $\mu_{q,j} = 1$, and its softmax renormalizes over those keys. We order the tokens along a locality-preserving Gilbert curve, so each block is a compact image or video patch. We call the set of key blocks that query block q keeps its *route*, and the choice of routes *routing*. Sparse-attention methods choose routes to save computation. RPO instead samples routes as its policy action during training and restores dense attention at inference.

Routing sites and units. Routing acts at a single noise level for each prompt group:

$$t^\star \sim \mathrm{Uniform}(\mathcal{T}_{\mathrm{route}}), \qquad \mathcal{T}_{\mathrm{route}} \subseteq \{1, \ldots, N\}. \tag{2}$$

The set $\mathcal{T}_{\mathrm{route}}$ contains noise levels where a route can still change the final output. All rollouts of a prompt group share $t^\star$. At $t^\star$, a fixed subset of self-attention layers, the *routed layers*, apply block masks, while every other layer and noise level runs dense attention. A *routing unit* is one routing decision, namely the route of query block q in routed layer ℓ at level $t^\star$, written $u = (t^\star, \ell, q)$. When each attention head h routes independently, the unit becomes $u = (t^\star, \ell, h, q)$. Each query block keeps a deterministic safety set S_q, which contains the block itself and a local spatial or spatiotemporal neighborhood. The remaining blocks form the candidate set:

$$C_q = \{1, \ldots, Q\} \setminus S_q. \tag{3}$$

The route of every unit contains S_q, and the policy decides only which blocks of C_q it adds. Every query therefore keeps its local context, and its softmax never runs over an empty set. Appendix A details the traversal, the block sizes, and the construction of S_q. This setup confines all stochasticity to one routed transition, while the prompt, the sampler, and the safety neighborhood stay deterministic.

4 ROUTING POLICY OPTIMIZATION

Sampler-space methods explore by injecting noise into the reverse trajectory. RPO instead explores through a temporary attention mask at the routed level $t^\star$. We first define this mask policy from native query-key affinities (Section 4.1). We then write the likelihood of a recorded mask history (Section 4.2). Finally, we pair two rollouts that differ only in their masks and turn their reward difference into a policy-gradient update (Section 4.3). Figure 2 summarizes this path from routed training to dense inference.

4.1 AFFINITY-CONDITIONED BERNOULLI ROUTING

The central challenge is to expose a stochastic action without adding a policy head or perturbing the sampler. A routed attention layer already computes query-key affinities, so we reuse them as policy

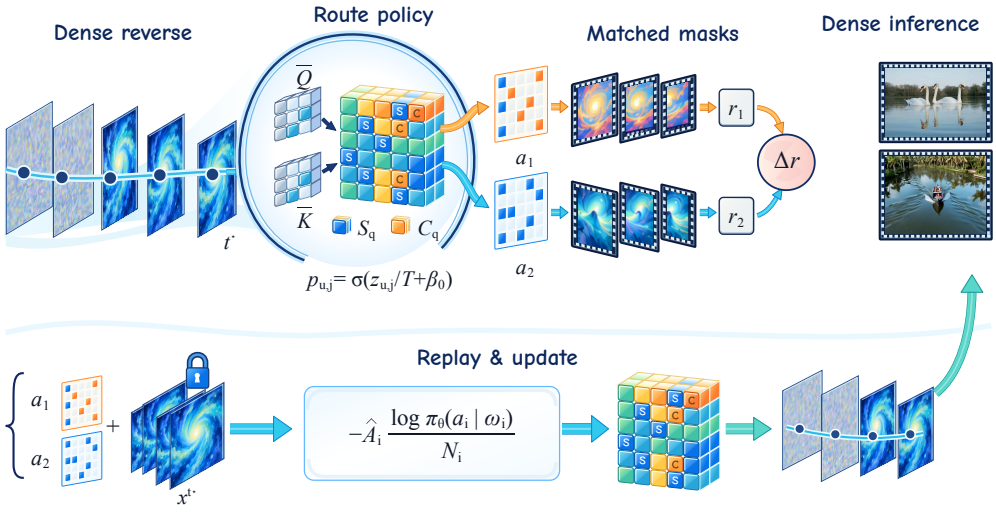


Figure 2: **A temporary route policy updates the dense generator.** At the shared denoising level $t^\star$, pooled native query-key affinities define Bernoulli keep probabilities over candidate blocks C_q, while safety blocks S_q remain active. Two matched rollouts record independent mask histories under the same prompt, content seed, routed level, and routed layers. Their reward difference weights the conditional replay likelihood, whose gradient updates attention parameters reused by the dense generator. At inference, RPO removes all masks and runs dense attention with the original deterministic sampler.

logits. For rollout i and routing unit u, let $\bar{Q}_{i,u}(\theta) \in \mathbb{R}^{d_\mathrm{h}}$ be the pooled query of block q, namely the mean of its token queries. Let $\bar{K}_{i,j}(\theta) \in \mathbb{R}^{d_\mathrm{h}}$ be the pooled key of candidate block $j \in C_q$, defined in the same way. The policy keeps each candidate block independently:

$$z_{i,u,j}(\theta) = \frac{\langle \bar{Q}_{i,u}(\theta), \bar{K}_{i,j}(\theta) \rangle}{\sqrt{d_\mathrm{h}}}, \qquad p_{i,u,j}(\theta) = \sigma\left(\frac{z_{i,u,j}(\theta)}{T} + \beta_0\right), \tag{4}$$
$$a_{i,u,j} \sim \mathrm{Bernoulli}(p_{i,u,j}(\theta)), \qquad j \in C_q.$$

Here θ contains the trainable generator parameters, d_h is the attention-head width, $T > 0$ is a temperature, σ is the logistic sigmoid, and β_0 is a fixed logit offset. The sampled entry $a_{i,u,j} \in \{0, 1\}$ keeps or drops block j for the query block of unit u. Under head-shared routing, we average the head-wise logits before the sigmoid. Intuitively, blocks that the query already attends to strongly are likely to stay visible, while the random draw still explores alternatives. The offset sets the initial sparsity. It is neither learned nor re-solved per row, so the number of kept blocks can change as the affinities evolve.

A configured keep ratio κ defines a nominal budget for loss normalization:

$$K_u = \lceil \kappa |C_q| \rceil, \qquad N_i = \sum_{u \in \mathcal{U}_{t^\star}} K_u. \tag{5}$$

Here $\mathcal{U}_{t^\star}$ is the set of routed units at the sampled noise level, and K_u is the nominal selection count of unit u. The normalizer N_i depends only on the routed units, their candidate-set sizes, and κ, not on the sampled keep count $\sum_{j \in C_q} a_{i,u,j}$. As a result, a denser sample cannot rescale its own loss. Native affinities thus decide which blocks stay visible, while the mask cardinality remains genuinely stochastic.

4.2 Conditional mask-history likelihood

Independent draws within one unit do not yet give the likelihood of a rollout. Routed units execute in a fixed forward order, and an earlier mask changes the hidden states behind every later logit. We therefore score each unit conditional on the recorded masks before it, denoted $a_{i,<u}$:

$$S_i(\theta) = \sum_{u \in \mathcal{U}_{t^\star}} \sum_{j \in C_q} \log \mathrm{Bern}\big(a_{i,u,j};\, p_{i,u,j}(\theta \mid a_{i,<u})\big) = \log \pi_\theta(a_i \mid \omega_i). \tag{6}$$

Here $\log \mathrm{Bern}(a; p) = a \log p + (1 - a) \log(1 - p)$, a_i is the complete mask history of rollout i, and ω_i is its shared context: the prompt, content seed, routed level, and routed layers. Replay restarts

from the stored latent $x_i^{t^\star}$, executes the recorded masks in order, and recomputes each probability. No mask is resampled. Therefore, Equation 6 is the exact likelihood of the recorded history under the current policy. It is not the exact gradient of the full generation and reward process, and Appendix E returns to this gap. With conditional replay, the routing action gains a precise log-likelihood that a policy gradient can weight.

4.3 Controlled two-rollout objective

A score-function update is only as informative as its advantage. A reward difference between unrelated rollouts mixes the effect of routing with changes in content. We therefore generate each prompt group as a matched pair with $G = 2$. Both members share the prompt c and a content seed that fixes the initial latent and every non-routing source of randomness. Only their routing seeds differ:

$$\xi^1_{\text{content}} = \xi^2_{\text{content}}, \qquad \xi^1_{\text{route}} \neq \xi^2_{\text{route}}. \tag{7}$$

Because the sampler is deterministic, the two members diverge only through their mask histories. For member $i \in \{1, 2\}$, let $\bar{i}$ denote the other member of the pair. The partner reward then serves as a leave-one-out baseline:

$$A^{\text{pair}}_{i,d} = r_{i,d} - r_{\bar{i},d}. \tag{8}$$

Given the shared context, $r_{\bar{i},d}$ does not depend on member i's action, so it is a valid score-function baseline. The two advantages have equal magnitude and opposite signs. Before replay, we winsorize the pair differences of each reward dimension, divide them by a robust interquartile scale, and sum the dimensions with weights w_d. A stop-gradient turns the result into a fixed replay weight $\widehat{A}_i$, and the processing preserves pair antisymmetry (Appendix C). The per-rollout loss and the pair objective are:

$$\mathcal{L}^{\text{pg}}_i(\theta) = -\widehat{A}_i \frac{S_i(\theta)}{N_i}, \qquad \mathcal{L}^{\text{RPO}}(\theta) = \frac{1}{2}\sum_{i=1}^{2} \mathcal{L}^{\text{pg}}_i(\theta). \tag{9}$$

Here S_i is the history likelihood from Equation 6, and N_i is the action-independent normalizer from Equation 5. When both members route the same units with the same candidate sets, $N_1 = N_2$. The update then raises the likelihood of the higher-reward mask history and lowers that of the lower-reward one, in proportion to their processed reward gap.

From routed training to dense inference. The route logits reuse native query and key projections and the upstream parameters that shape them. The routing loss therefore updates parameters that dense attention also uses. After training, we remove every mask and run the updated image or video generator with dense attention and its original deterministic sampler.

5 Experiments

Our experiments test whether reward learned through temporary routing transfers to the dense generator used at inference. After the setup, we evaluate text-to-image generation (Section 5.2), compare with sampler-space RL on video (Section 5.3), and measure VBench transfer and training cost (Section 5.4).

5.1 Setup

Image setting. We train Stable Diffusion 3.5 Medium (Esser et al., 2024) on prompts from the PickScore SFW split (Kirstain et al., 2023). Each rollout generates a 512×512 image with a deterministic 10-step flow sampler and no classifier-free guidance. The training reward is HPSv3 (Ma et al., 2025) scaled by 0.1. After training, we evaluate the dense model on DrawBench prompts (Saharia et al., 2022). An aesthetic score and DeQA-Score (You et al., 2025) measure image quality, while ImageReward (Xu et al., 2023), PickScore, and UnifiedReward (Wang et al., 2025b) measure preference.

Video setting. We train Wan2.2-TI2V-5B (Wan Team et al., 2025), a text-to-video flow-matching generator, on a curated creative prompt distribution. Each rollout renders 13 frames at 704×1280 with a fixed 50-step Euler sampler. The reward is HPSv3 scaled by 0.1 and is computed on uniformly sampled frames. During training, we probe the dense model on a fixed 61-prompt validation set with a 10-step sampler.

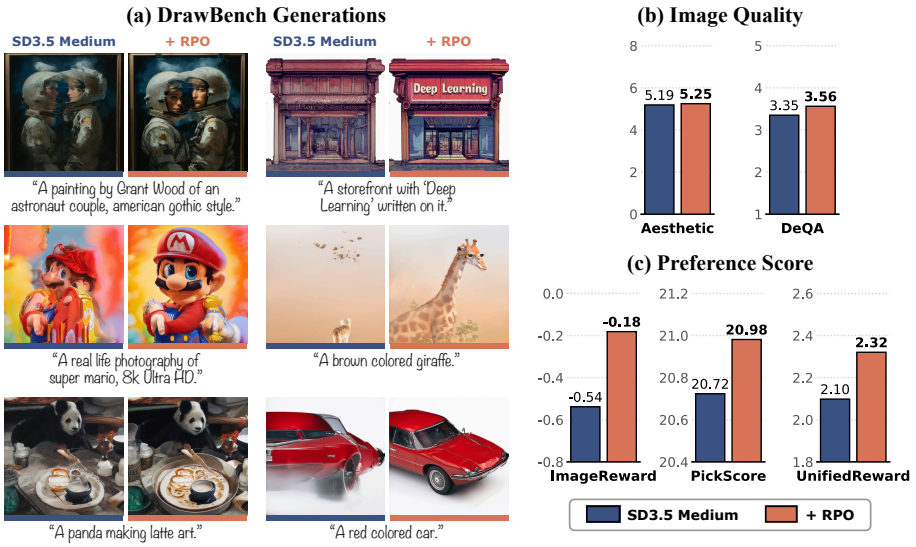
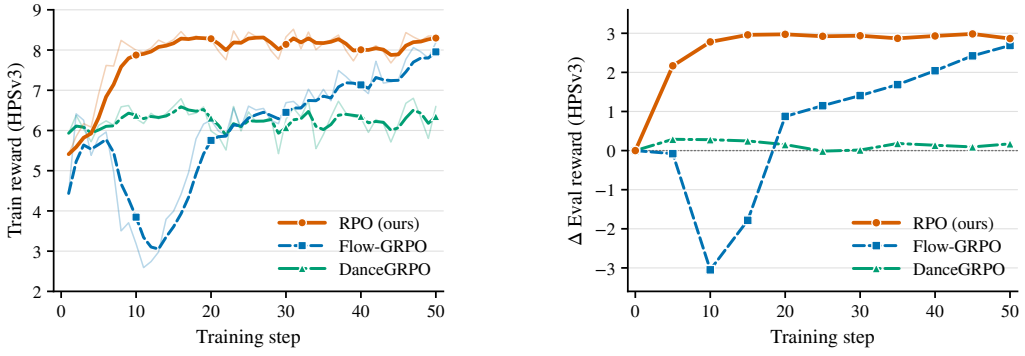


Figure 3: **RPO on Stable Diffusion 3.5 Medium.** (a) Samples before and after RPO on DrawBench prompts. Each pair shares the prompt and initial noise. (b, c) DrawBench scores over 1,000 images from 200 prompts with 5 seeds, generated without classifier-free guidance. RPO improves desired property across all five reported metrics.



(a) HPSv3 reward on the sampled training rollouts.

(b) Evaluation reward relative to initial score.

Figure 4: **RPO improves reward rapidly and preserves its evaluation gain.** We compare RPO, Flow-GRPO, and DanceGRPO over the total 50 training steps on Wan2.2-TI2V-5B with HPSv3 reward. (a) Per-step reward on each method's own training rollouts (faint) and its exponential moving average with factor 0.4 (bold). (b) Dense evaluation reward on the shared validation set, relative to each method's initial score.

Baselines and training. On Wan2.2, we compare against Flow-GRPO (Liu et al., 2025a) and DanceGRPO (Xue et al., 2025b). Both baselines train the same generator with the same HPSv3 reward, but they explore by sampling SDE trajectories. DanceGRPO uses groups of $G = 8$ and updates the full model. RPO instead collects 64 matched pairs ($G = 2$) per training step and routes a single denoising level. Appendix B lists the routing and optimization settings for both generators. Appendix D details the evaluation protocols.

5.2 Text-to-Image Generation

Figure 3 first compares SD3.5 Medium before and after RPO training on DrawBench. RPO improves all five metrics, although none of them is the training reward. The preference metrics show the clearest gains. ImageReward rises from -0.54 to -0.18, and UnifiedReward rises from 2.10 to 2.32. Image quality also improves, with DeQA-Score rising from 3.35 to 3.56. In the samples, the trained model writes the requested storefront text and renders a clear brown giraffe, whereas the base model misses both. Because each pair shares its prompt, initial noise, and sampler, these gains come from the updated weights of the dense generator alone.

Table 1: **Comparison on VBench across multiple dimensions with DanceGRPO and Flow-GRPO, plus overall score and training cost.** Scores are percentages. The Wan2.2 rows share one generation configuration. CogVideoX-5B and HunyuanVideo use the same prompts, frame count, denoising steps, and scoring, but keep their native resolution and guidance. Training GPU-hours are measured on H100 GPUs.

Method	Train GPU-h	Aesthetic quality	Imaging quality	Subject consistency	Motion smoothness	Object class	Multiple objects	Human action	Scene	Overall score
CogVideoX-5B	–	60.17	61.92	96.50	97.42	82.44	57.29	86.40	42.73	81.30
HunyuanVideo	–	63.05	63.78	98.76	99.42	65.79	56.46	83.20	27.01	78.69
Wan2.2-TI2V-5B	–	61.91	67.71	98.37	98.76	81.01	59.15	79.40	37.41	81.61
RPO	**24.1**	**62.64**	**68.65**	**98.69**	**99.06**	**85.41**	**61.30**	**84.20**	**40.84**	**82.00**
DanceGRPO	102.3	61.74	67.69	98.38	98.75	80.81	59.34	80.00	36.95	81.46
Flow-GRPO	75.4	60.09	67.43	97.59	97.88	79.10	59.98	81.80	36.73	81.19

5.3 Comparison with sampler-space RL on video

Video generation raises the cost of every rollout, so learning speed matters more. Figure 4 tracks the first 50 training steps of all three methods on Wan2.2. Figure 4a reports the reward of the rollouts each method samples during training. RPO raises this reward from about 5.4 to a band around 8.2. DanceGRPO moves from about 5.9 to a band around 6.2. Flow-GRPO first collapses and only returns to a comparable level near step 50. However, training reward scores routed or SDE rollouts rather than the dense model used at inference.

Figure 4b therefore reports the decisive endpoint. It shows the dense evaluation reward as a gain over each method's own step-zero score, which cancels a constant offset between rollout backends. RPO gains $+2.17$ by step 5 and about $+2.9$ by step 15, and it stays near $+3$ through step 50. Flow-GRPO follows its training reward. Its dense gain falls to about -3 at step 10 and then recovers gradually to $+2.7$ at step 50. DanceGRPO peaks at $+0.29$ at step 5 and then stays between about 0 and $+0.2$. RPO thus reaches its final dense gain within 15 steps and keeps it. In contrast, Flow-GRPO needs the whole window to approach that level, and DanceGRPO accumulates no gain. These comparison demonstrate RPO can achieve a faster convergence speed with a lower training cost.

5.4 VBench transfer and training cost

A single reward scalar compresses appearance and semantics, so reward gains alone do not establish broad video quality. We therefore evaluate the base model and the three trained checkpoints on VBench, which decomposes video generation into 16 fine-grained dimensions with tailored prompts and evaluation procedures (Huang et al., 2024). Each model generates $6{,}220$ 13-frame videos from shared prompts and initial noise with a deterministic sampler (Appendix D).

Table 1 lists the 8 dimensions where RPO exceeds both RL baselines, together with the overall score and training cost. The largest gains over the base model are semantic. Object class improves by $+4.40$, human action by $+4.80$, scene by $+3.43$, and multiple objects by $+2.15$. RPO also improves aesthetic quality, imaging quality, subject consistency, and motion smoothness over the base model. It is the only trained checkpoint above the base model overall. Moreover, it trains with 24.1 GPU-hours, which is $3.1\times$ less than Flow-GRPO and $4.2\times$ less than DanceGRPO, although the runs are not compute-matched.

The remaining dimensions reveal trade-offs. RPO lowers color and appearance style relative to the base model, although it still leads the trained baselines on color. Flow-GRPO raises dynamic degree by 12.50 points but loses consistency, smoothness, aesthetic quality, color, and object-class accuracy. RPO therefore transfers reward mainly through semantic alignment, whereas Flow-GRPO trades visual quality for motion. The four Wan2.2 overall scores span only 0.81 points on one seed set, so we read the aggregate ranking as directional. CogVideoX-5B (Yang et al., 2024) and HunyuanVideo (Kong et al., 2024) serve as external references under matched prompts and scoring.

Figure 5 makes these semantic gains visible on four held-out prompts, where the RPO videos follow the prompts more closely. Both checkpoints run dense attention with the same prompt, initial noise, and sampler, so they differ only in their weights. Across image and video, the gains learned through routing persist after the masks are removed.

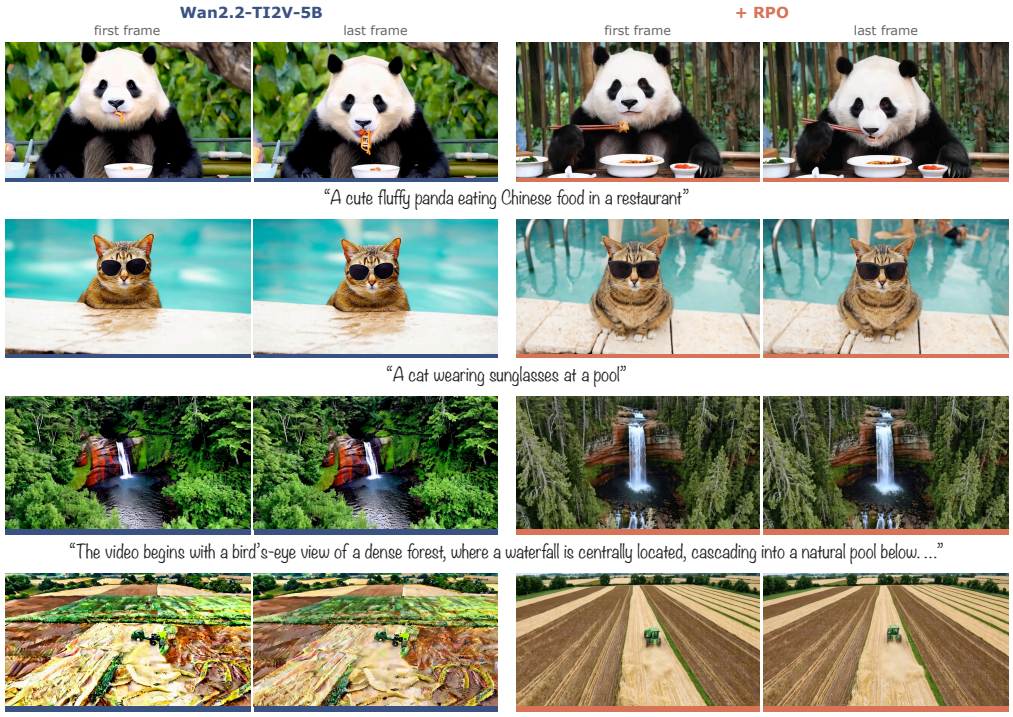


Figure 5: **Text-to-video samples from Wan2.2-TI2V-5B before and after RPO.** Each row shows one prompt. For each model we show the first and last of the 13 generated frames (1280×704, 50 sampling steps). Both models use the same prompt and initial noise. RPO follows the prompt more closely and replaces distorted, painterly textures with coherent, realistic scenes.

6 Discussion and Limitations

RPO places its policy action inside attention, and this design choice also defines its current limits. First, the framework is confined to attention. Its action is a mask over key-value blocks, so it requires a backbone that computes query-key affinities. We have not yet generalized routing to other architectures, where no such affinity exists to reuse as a policy logit. Second, the routing score is coarse. Each route logit is the inner product of a pooled query and a pooled key, and pooling averages away token-level structure within a block. The resulting score can therefore be noisy, and a more fine-grained score design could give the policy a more faithful view of which blocks matter. Finally, RPO currently adapts to an existing generator rather than shaping it. Co-designing the network architecture with RPO is a promising direction for achieving both higher efficiency and larger RL gains.

7 Conclusion

In this work, we addressed the cost of online reinforcement learning for visual generators without perturbing their reverse denoising trajectories. We introduced Routing Policy Optimization, a sampler-preserving method that draws block-wise Bernoulli routes from native query-key affinities at one selected denoising level. A controlled pair holds the non-routing generation context fixed, so its reward difference weights the conditional likelihood of the two recorded mask histories. Because the routing policy updates native attention parameters, inference removes the masks and restores dense attention under the original deterministic sampler. Experiments on Stable Diffusion 3.5 and Wan2.2 demonstrate improved reward-aligned quality across text-to-image and text-to-video generation while requiring substantially less training compute than sampler-space baselines. By moving stochastic exploration from the sampler into attention, RPO provides an efficient policy interface for reward-aligned visual generation without altering the inference process.

AI USE STATEMENT

We used generative AI tools to polish the writing of this paper, namely to edit the text for grammar, readability, and concision. We did not use generative AI tools for any task that requires disclosure under the ICLR 2027 AI policy. In particular, AI tools did not propose hypotheses, develop the method or its derivations, design experiments, implement code, generate or clean data, or interpret results. Writing proofs, translation, and qualitative data analysis are not applicable to this work. The authors reviewed every AI-assisted edit to confirm that it preserves the intended technical meaning. We take responsibility for the final content of this work, including all text, claims, and results.

ETHICS STATEMENT

Visual generative models can amplify reward-model biases and hallucinate plausible content, and RPO is a general policy-gradient tool. We use a public preference reward model on synthetic prompts and do not fine-tune on user data. Reward hacking is a known risk of online reinforcement learning for generation and is discussed in Appendix E.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 state the routing policy and the estimator in full. Section 5.1 and Appendix B record the models, prompt sets, reward model, group size, keep ratio, temperature, offset, and update budget for each generator. Appendix D describes every evaluation protocol.

REFERENCES

Mishan Aliev, Eva Neudachina, Ilya Bykov, Aleksandr Oganov, Kirill Struminsky, Aibek Alanov, and Denis Rakitin. ReCache: Learning Budget-Aware Caching Schedules for Diffusion Models via REINFORCE, 2026.

Søren Asmussen and Peter W. Glynn. *Stochastic Simulation: Algorithms and Analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer, 2007. ISBN 978-0-387-30679-7. doi: 10.1007/978-0-387-69033-9.

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.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Pengtao Chen, Xianfang Zeng, Maosen Zhao, Peng Ye, Mingzhu Shen, Wei Cheng, Gang Yu, and Tao Chen. Sparse-vDiT: Unleashing the Power of Sparse Attention to Accelerate Video Diffusion Transformers, 2025.

Jaemoo Choi, Wei Guo, Yuchen Zhu, Arash Vahdat, Molei Tao, Julius Berner, and Yongxin Chen. Scaling Reinforcement Learning for Diffusion Models via Velocity Matching, 2026.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly Fine-Tuning Diffusion Models on Differentiable Rewards, 2023.

Haoyou Deng, Keyu Yan, Chaojie Mao, Xiang Wang, Yu Liu, Changxin Gao, and Nong Sang. DenseGRPO: From Sparse to Dense Reward for Flow Matching Model Alignment, 2026.

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, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, 2024.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning Intrinsic Sparse Attention in Your LLMs, 2024.

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, 2025.

Xiaoxuan He, Siming Fu, Zeyue Xue, Weijie Wang, Ruizhe He, Yuming Li, Dacheng Yin, Shuai Dong, Haoyang Huang, Hongfa Wang, Nan Duan, and Bohan Zhuang. Flash-GRPO: Efficient Alignment for Video Diffusion via One-Step Policy Optimization, 2026.

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

Yushi Huang, Xiangxin Zhou, Ruoyu Wang, Chi Zhang, Jun Zhang, and Tianyu Pang. Reinforcing Few-step Generators via Reward-Tilted Distribution Matching, 2026.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818, 2024. doi: 10.1109/CVPR52733.2024.02060.

Wooil Jung. Dropout-GRPO: Variational stochasticity for continuous latent reasoning, 2026.

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

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. HunyuanVideo: A systematic framework for large video generative models, 2024.

Bangzheng Li, Jianmo Ni, Chen Qu, Ian Miao, Liu Yang, Xingyu Fu, Muhao Chen, and Derek Zhiyuan Cheng. Reinforced Attention Learning, 2026.

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, 2025a.

Yuming Li, Yikai Wang, Yuying Zhu, Zhongyu Zhao, Ming Lu, Qi She, and Shanghang Zhang. BranchGRPO: Stable and Efficient GRPO with Structured Branching in Diffusion Models, 2025b.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, Xiaohong Liu, Fei Yang, Pengfei Wan, Di Zhang, Kun Gai, Yujiu Yang, and Wanli Ouyang. Improving video generation with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Runtao Liu, Haoyu Wu, Ziqiang Zheng, Chen Wei, Yingqing He, Renjie Pi, and Qifeng Chen. VideoDPO: Omni-Preference Alignment for Video Diffusion Generation, 2024.

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

Yifu Luo, Xinhao Hu, Keyu Fan, Haoyuan Sun, Zeyu Chen, Bo Xia, Tiantian Zhang, Yongzhe Chang, and Xueqian Wang. Reinforcement Learning Meets Masked Generative Models: Mask-GRPO for Text-to-Image Generation, 2025.

Yihong Luo, Tianyang Hu, and Jing Tang. Reinforcing diffusion models by direct group preference optimization. In *International Conference on Learning Representations (ICLR)*, 2026.

Chengtao Lv, Yumeng Shi, Yushi Huang, Ruihao Gong, Shen Ren, and Wenya Wang. Light Forcing: Accelerating Autoregressive Video Diffusion via Sparse Attention, 2026.

Tianren Ma, Mu Zhang, Yibing Wang, and Qixiang Ye. Consolidating reinforcement learning for multimodal discrete diffusion models. In *International Conference on Learning Representations (ICLR)*, 2026.

Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. HPSv3: Towards wide-spectrum human preference score. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 15086–15095. IEEE, 2025. doi: 10.1109/ICCV51701.2025.01400. URL https://doi.org/10.1109/ICCV51701.2025.01400.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2022.

Mihir Prabhudesai, Russell Mendonca, Zheyang Qin, Katerina Fragkiadaki, and Deepak Pathak. Video Diffusion Alignment via Reward Gradients, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2021.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 36479–36494, 2022. doi: 10.52202/068431-2643.

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, 2024.

Jinjie Shen, Wei Deng, Xian Hu, Daiguo Zhou, and Jian Luan. STAR: SpatioTemporal Adaptive Reward Allocation for Text-to-Image RL Post-Training, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

Bingda Tang, Yuhui Zhang, Xiaohan Wang, Jiayuan Mao, Ludwig Schmidt, and Serena Yeung-Levy. V-GRPO: Online Reinforcement Learning for Denoising Generative Models Is Easier than You Think, 2026.

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

Wan Team, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng

Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025.

Jin Wang, Jianxiang Lu, Guangzheng Xu, Comi Chen, Haoyu Yang, Linqing Wang, Peng Chen, Mingtao Chen, Zhichao Hu, Longhuang Wu, Shuai Shao, Qinglin Lu, and Ping Luo. TAGRPO: Boosting GRPO on Image-to-Video Generation with Direct Trajectory Alignment, 2026.

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, 2025a.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation, 2025b.

Yihong Wu, Liheng Ma, Lei Ding, Muzhi Li, Xinyu Wang, Kejia Chen, Zhan Su, Zhanguang Zhang, Chenyang Huang, Yingxue Zhang, Mark Coates, and Jian-Yun Nie. It takes two: Your GRPO is secretly DPO, 2025a.

Ziyi Wu, Anil Kag, Ivan Skorokhodov, Willi Menapace, Ashkan Mirzaei, Igor Gilitschenski, Sergey Tulyakov, and Aliaksandr Siarohin. DenseDPO: Fine-Grained Temporal Preference Optimization for Video Diffusion Models, 2025b.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse VideoGen: Accelerating Video Diffusion Transformers with Spatial-Temporal Sparsity, 2025.

Boxun Xu, Yuming Du, Zichang Liu, Siyu Yang, Ziyang Jiang, Siqi Yan, Rajasi Saha, Albert Pumarola, Wenchen Wang, and Peng Li. Sparse Forcing: Native Trainable Sparse Attention for Real-time Autoregressive Diffusion Video Generation, 2026a.

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, 2023.

Juntao Xu, Shihong Li, Hoi Fan Au, and Ning Zhu. ToPO: Token-Conditioned Preference Routing for Attention-Based Latent Diffusion Models, 2026b.

Yixian Xu, Yuanrui Zhang, Shengjie Luo, Liwei Wang, and Di He. Designing Reinforcement Learning for Diffusion Models: A Unified Path-Space View, 2026c.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage Weighted Matching: Aligning RL with Pretraining in Diffusion Models, 2025a.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. Dancegrpo: Unleashing grpo on visual generation, 2025b.

Yankai Yang, Yancheng Long, Wei Chen, Xingyu Lu, Hongyang Wei, Bin Wen, Fan Yang, Tingting Gao, Han Li, and Shuo Yang. SpatialFlow-GRPO: Where Spatial Credit Drives Image Editing, 2026.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2024.

Zhiyuan You, Xin Cai, Jinjin Gu, Tianfan Xue, and Chao Dong. Teaching large language models to regress accurate image quality scores using score distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14483–14494, 2025. doi: 10.1109/CVPR52734.2025.01350.

Zhihang Yuan, Hanling Zhang, Pu Lu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. DiTFastAttn: Attention Compression for Diffusion Transformer Models, 2024.

Guohui Zhang, Hu Yu, Xiaoxiao Ma, Yaning Pan, Hang Xu, and Feng Zhao. MaskFocus: Focusing Policy Optimization on Critical Steps for Masked Image Generation, 2025a.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. VSA: Faster Video Diffusion with Trainable Sparse Attention, 2025b.

Ruicheng Zhang, Kaixi Cong, Jun Zhou, Zhizhou Zhong, Zunnan Xu, Shuiyang Mao, Wei Liu, and Xiu Li. KVPO: ODE-native GRPO for autoregressive video alignment via KV semantic exploration, 2026.

Yuechen Zhang, Jinbo Xing, Bin Xia, Shaoteng Liu, Bohao Peng, Xin Tao, Pengfei Wan, Eric Lo, and Jiaya Jia. Training-free efficient video generation via dynamic token carving. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, pp. 90947–90980, 2025c. doi: 10.52202/085713-2738.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Hao Luo, Yibing Song, Gao Huang, Fan Wang, and Yang You. DyDiT++: Diffusion Transformers with Timestep and Spatial Dynamics for Efficient Visual Generation, 2025.

Wangbo Zhao, Yizeng Han, Zhiwei Tang, Jiasheng Tang, Pengfei Zhou, Kai Wang, Bohan Zhuang, Zhangyang Wang, Fan Wang, and Yang You. RAPID^3: Tri-level reinforced acceleration policies for diffusion transformer. In *International Conference on Learning Representations (ICLR)*, 2026.

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 *International Conference on Learning Representations (ICLR)*, 2026.

Yifan Zhou, Zeqi Xiao, Tianyi Wei, Shuai Yang, and Xingang Pan. Trainable Log-linear Sparse Attention for Efficient Diffusion Transformers, 2025.

A GILBERT BLOCKS AND THE SAFETY SET

Section 3 places every routing decision on a block of visual tokens and protects a local safety set S_q. It states both constructions briefly and assumes a single block size B. This appendix gives the full construction that both generators use. We first explain why the token order matters and define the frame-sliced Gilbert order. We then decouple the block size into query, key, and safety partitions. Finally, we define S_q and the executed attention mask, and we report the resulting block and candidate counts for Wan2.2 and Stable Diffusion 3.5.

Why the token order matters. A routing unit pools a contiguous run of tokens into one query or key vector, and one Bernoulli draw keeps or drops a whole run. Both generators store visual tokens in raster order, which scans each latent frame row by row. A contiguous raster run is therefore a thin horizontal strip. On the 32×32 token grid of Stable Diffusion 3.5, a 32-token run is one entire image row. Its pooled key averages tokens that lie far apart, while its vertical neighbors lie a full row away in the sequence. As a result, a route decision on a raster run couples unrelated content, and a small spatial neighborhood cannot be expressed as a few runs. The token order should instead turn every contiguous run into a compact spatial patch.

A.1 FRAME-SLICED GILBERT ORDER

A generalized Hilbert curve, or Gilbert curve, visits every cell of a rectangle of arbitrary size exactly once and moves to an adjacent cell at every step. Jenga (Zhang et al., 2025c) partitions video tokens along such a curve for block-sparse attention, and we adopt the frame-sliced variant of its released implementation. Let the visual latent contain F frames of $H \times W$ tokens, so it holds $M = FHW$

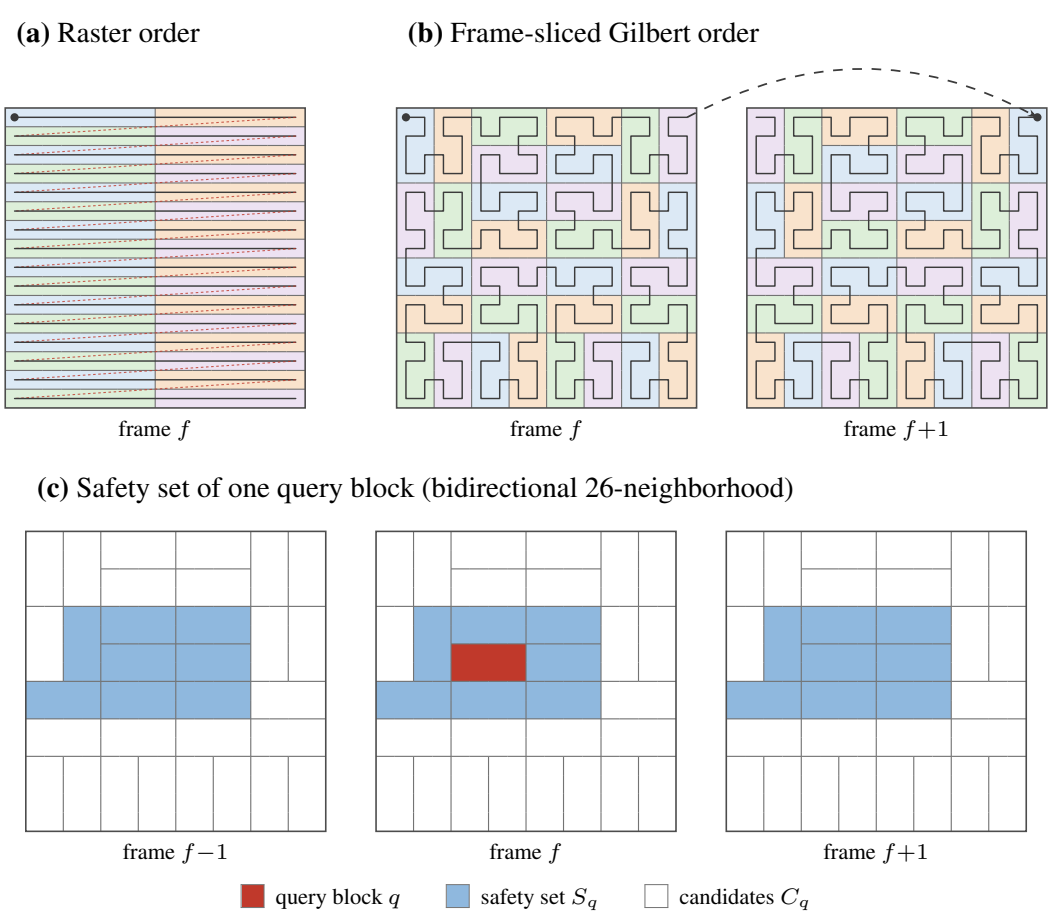


Figure 6: **Gilbert blocks and the safety set on toy** 16×16 **frames with 8-token blocks.** (a) Raster order turns each block into a horizontal strip and jumps back at every row end (red dotted segments). (b) The frame-sliced Gilbert order turns each block into a compact 2×4 or 4×2 patch. The curve continues from the end of frame f to the same position in frame $f+1$ (dashed arrow) and traverses that frame in a reflected direction. (c) The safety set S_q of one query block q when $B_{\mathrm{q}} = B_{\mathrm{k}} = B_{\mathrm{s}} = 8$. Because attention is bidirectional, S_q extends to both the previous and the next frame. All remaining blocks form the candidate set C_q.

tokens. Token $p = (f, m, n)$ lies in frame $f \in \{0, \ldots, F-1\}$, row m, and column n. The order assigns each token a rank:

$$\gamma(f, m, n) = fHW + g_{H,W}(\phi_f(m, n)) \,. \tag{10}$$

Here $g_{H,W}$ assigns the ranks $1, \ldots, HW$ along the two-dimensional Gilbert curve of an $H \times W$ grid, which starts and ends at the two corners of one longer side. The map ϕ_f reflects the rows, the columns, or both, so that frame f starts at the corner nearest to the end of frame $f-1$. The first map ϕ_0 is the identity, and the offset fHW keeps every frame contiguous. Intuitively, the curve fills one frame, steps to the same spatial position in the next frame, and fills that frame in a reflected direction. Every step therefore moves to a neighboring token, including the step across a frame boundary. Figure 6 contrasts this order with the raster order.

Contiguous runs of this order are compact patches. On the 32×32 grid of Stable Diffusion 3.5, every 32-token run is a 4×8 or 8×4 rectangle, and every 128-token run is a 16×8 or 8×16 rectangle. On the 22×40 frames of Wan2.2, every 16-token run stays inside one frame, and the longer side of its bounding box spans 5.7 tokens on average and 8 tokens at most. The same run in raster order would be a 1×16 strip.

Bidirectional attention. Both generators denoise all visual tokens jointly with bidirectional self-attention, so every query can attend to every frame. The order therefore depends only on the latent

grid and not on a generation schedule. We compute it once per resolution and reuse it for all routed layers, heads, noise levels, and rollouts.

Reordering inside attention. The order acts only inside each routed self-attention layer. After the native projections, query-key normalization, and rotary position embedding where the generator uses one, we permute the queries, keys, and values into Gilbert order. We then apply the routed mask and permute the output back to raster order. Dense attention is equivariant to a shared permutation of its queries, keys, and values, and every positional signal is attached before the permutation. The reordering thus changes which token pairs share a routing decision, but it leaves positions and the dense computation unchanged. At inference, RPO runs the stock dense attention without any reordering.

A.2 Query, key, and safety blocks

A single block size ties three different roles together. The query block sets the number of routing units, the key block sets the resolution of the action, and the safety block sets the extent of the protected neighborhood. The implementation therefore partitions the Gilbert sequence at three independent sizes:

$$\mathcal{Q}_q = \{p : \lceil \gamma(p)/B_{\mathrm{q}} \rceil = q\}, \quad \mathcal{K}_j = \{p : \lceil \gamma(p)/B_{\mathrm{k}} \rceil = j\}, \quad \mathcal{R}_b = \{p : \lceil \gamma(p)/B_{\mathrm{s}} \rceil = b\}. \tag{11}$$

Here $\mathcal{Q}_q$ is query block $q \in \{1, \dots, Q\}$ with $Q = \lceil M/B_{\mathrm{q}} \rceil$, $\mathcal{K}_j$ is key block $j \in \{1, \dots, J\}$ with $J = \lceil M/B_{\mathrm{k}} \rceil$, and $\mathcal{R}_b$ is safety block b. The sizes B_{q}, B_{k}, and B_{s} count Gilbert-ordered tokens, and the last block of each partition may be shorter. For routing unit $u = (t^\star, \ell, h, q)$, the pooled query $\bar{Q}_{i,u}$ in Equation 4 is the mean of the head-h queries over $\mathcal{Q}_q$, and the pooled key $\bar{K}_{i,j}$ is the mean of the head-h keys over $\mathcal{K}_j$. One Bernoulli entry $a_{i,u,j}$ then keeps or drops key block $\mathcal{K}_j$ for every query in $\mathcal{Q}_q$. A smaller B_{k} yields more candidates and finer masks, whereas B_{s} alone controls how much local context stays protected. When $B_{\mathrm{q}} = B_{\mathrm{k}} = B_{\mathrm{s}} = B$, the three partitions coincide, $J = Q$, and the construction reduces to Section 3. This decoupling refines the routing action without shrinking the protected neighborhood.

A.3 Safety set and executed mask

Bernoulli routing makes the keep count random, and a single draw may drop every candidate of a unit. Without protection, a query could lose all of its keys, which leaves the attention softmax undefined, or lose its local context at random. The safety set removes both failure modes by construction. Two tokens $p = (f, m, n)$ and $p' = (f', m', n')$ are neighbors when $\max(|f - f'|, |m - m'|, |n - n'|) \le 1$. This is the 26-neighborhood in the spatiotemporal grid, and it reduces to the 8-neighborhood for images, where $F = 1$. Following the block adjacency mask of Jenga (Zhang et al., 2025c), safety blocks b and b' are adjacent, written $b \approx b'$, when some token of $\mathcal{R}_b$ neighbors some token of $\mathcal{R}_{b'}$. Every safety block is adjacent to itself. Let $s(p)$ denote the safety block that contains token p. We define:

$$\mathcal{E}_{\mathrm{safe}} = \{(p, p') : s(p) \approx s(p')\}, \quad S_q = \{j : \mathcal{Q}_q \times \mathcal{K}_j \subseteq \mathcal{E}_{\mathrm{safe}}\}, \quad C_q = \{1, \dots, J\} \setminus S_q. \tag{12}$$

Here $\mathcal{E}_{\mathrm{safe}}$ is the set of query-key token pairs that are always attended. A key block enters S_q only when every pair between it and query block q is safe. The candidate set C_q collects all other key blocks, which matches Equation 3 with Q replaced by J. Bernoulli entries exist only for $j \in C_q$, so safety blocks carry no probability and add no term to $S_i(\theta)$ in Equation 6. The relation depends only on the grid and the block sizes, so every head, noise level, and rollout shares it.

The executed mask combines the safety relation with the sampled action. For query token $p \in \mathcal{Q}_q$ and key token $p' \in \mathcal{K}_j$ with $j \in C_q$, the mask of rollout i at unit u is

$$\mu_{i,u}(p, p') = \max\Big(\mathbf{1}\big[(p, p') \in \mathcal{E}_{\mathrm{safe}}\big],\, a_{i,u,j}\Big). \tag{13}$$

Here $\mu_{i,u}(p, p') = 1$ means that query token p attends to key token p', and $\mathbf{1}[\cdot]$ is the indicator function. Pairs with $j \in S_q$ are always attended. The sampled entry $a_{i,u,j}$ therefore decides only the unsafe pairs of a candidate block. A candidate that the safety relation covers in part remains a meaningful action, because dropping it removes only its unsafe pairs while the safe part stays attended. Every query token keeps at least its own safety block, so the softmax stays defined even when a draw keeps no candidate. Intuitively, the route action decides which non-local context each query block sees, while its local neighborhood stays deterministic.

Table 2: **Token grids, blocks, and candidate sets of the reference configurations.** Counts refer to one routed layer and head. Block counts and the safe-pair fraction cover visual tokens only. The safe-pair fraction is the share of a query block's visual query-key pairs that lie in $\mathcal{E}_{\text{safe}}$. Means and ranges are taken over query blocks.

	Wan2.2	Stable Diffusion 3.5
Output size	$13 \times 704 \times 1280$	512×512
Token grid $F \times H \times W$	$4 \times 22 \times 40$	$1 \times 32 \times 32$
Visual tokens M	3,520	1,024
Block sizes B_{q} / B_{k} / B_{s}	64 / 16 / 32	128 / 32 / 32
Query blocks Q / key blocks J	55 / 220	8 / 32
Safety set $\lvert S_q \rvert$, mean	18.8	1
Candidate set $\lvert C_q \rvert$, mean [range]	201.2 [184, 210]	31 [31, 31]
Candidates covered in part, mean	19.3	10.5
Safe-pair fraction, mean [range]	12.9% [6.8%, 21.4%]	20.5% [17.2%, 23.4%]
Nominal budget K_u at $\kappa = 0.5$, mean	100.6	16

Symmetric neighborhood in time. Because attention is bidirectional, the neighborhood is symmetric in time as well as in space, and so is $\mathcal{E}_{\text{safe}}$. A query block in frame f keeps safe context in frames $f-1$, f, and $f+1$, as Figure 6(c) shows. No causal constraint removes the following frame from the neighborhood.

Text conditioning. Routing acts only on visual self-attention. Stable Diffusion 3.5 runs a joint attention over image and text tokens, and we route only the image-to-image pairs. Every pair that involves a text token stays dense, so each image query keeps the full text band in addition to its safety neighborhood. The second, image-only self-attention of Stable Diffusion 3.5 has no text band and relies on the safety set alone. Wan2.2 injects text through cross-attention, which is never routed.

A.4 Block counts in the reference configurations

Table 2 reports the token grid, the block counts, and the candidate-set sizes of both reference configurations. For Wan2.2, the video autoencoder compresses 13 frames at 704×1280 by a factor of 4 in time and 16 in space, and patchification halves each spatial side, which gives a $4 \times 22 \times 40$ token grid. For Stable Diffusion 3.5, a 512×512 image gives a 32×32 image grid. The text tokens are never routed, so the counts exclude them. In Wan2.2, the safety relation covers about one eighth of each query block's visual pairs and leaves about 200 candidate key blocks. In Stable Diffusion 3.5, each 128-token query block spans four safety blocks, so only one key block is covered in full, while about a third of the candidates are covered in part. In both generators, roughly four fifths or more of the visual query-key pairs remain under the routing policy. The safety set thus guarantees valid local attention while leaving most of the attention pattern to the learned route.

B Implementation details

This appendix lists the settings behind Section 5.1. The routing policy and objective are identical for both generators. Only the routing hyperparameters and optimization settings differ.

Stable Diffusion 3.5. The reference configuration routes denoising slot 2 with temperature $T = 1$, fixed offset $\beta_0 = 1$, and nominal keep ratio $\kappa = 0.5$. Each keep probability is therefore $p_{i,u,j} = \sigma(z_{i,u,j} + 1)$, and the observed keep ratio stays near 0.58. Per-head routing pools 128 query and 32 key tokens per block and builds the safety set from 32-token blocks. It covers half of the 24 transformer blocks plus the final block, and it routes both the joint attention and the second image-only self-attention. Each wave contains 64 prompt groups at $G = 2$, split into two optimizer minibatches of 32 groups. We train rank-32 LoRA adapters with $\alpha = 64$, learning rate 5×10^{-5}, weight decay 10^{-4}, and no warmup. Advantage processing uses 5% winsorization, batch-IQR scaling with floor 0.03, and a near-tie gate of 0.01. Training runs for 80 steps on one node with eight H100 GPUs, with evaluation every 5 steps.

Wan2.2. The reference configuration uses temperature $T = 2$ and fixed offset $\beta_0 = -1$, so each keep probability is $p_{i,u,j} = \sigma(z_{i,u,j}/2 - 1)$. We selected this offset as the initial sparsity operating point. No row-wise root is solved, and the expected number of retained blocks is not constrained to equal K_u. Routing pools 64 query tokens and 16 key tokens per block, routes each head independently, and builds the safety set from 32-token blocks (Appendix A). The configured ratio $\kappa = 0.5$ sets the nominal normalizer in Equation 5, while the realized keep count is recorded separately. We optimize the trainable generator parameters with learning rate 2×10^{-5}. Each wave contains 64 prompt groups at $G = 2$ and is divided into two disjoint optimizer minibatches of 32 groups. The first minibatch is fresh, and the second contains different pairs from the same wave. An advantage-spread gate with threshold $\tau_{\text{spread}} = 4 \times 10^{-3}$ zeros the advantage of near-tie pairs, so those pairs contribute no update.

Optimizer safeguards. Three generic safeguards stabilize optimization. They are not components of RPO. First, the second minibatch uses standard PPO ratio clipping with width 10^{-4} for both generators. Let S_i^{old} be the behavior log-likelihood stored during collection. We use the normalized ratio $\rho_i(\theta) = \exp((S_i(\theta) - S_i^{\text{old}})/N_i)$, which leaves the Bernoulli likelihood of Section 4.2 unchanged. Second, on Wan2.2 we cap the processed advantage at $A_{\max}^{\text{loss}} = 5$. Third, Wan2.2 training applies a global gradient-norm cap $\Gamma_{\text{grad}} = 1$, which treats every trainable gradient as one concatenated vector and rescales its accumulated norm before AdamW consumes it. In the reference Wan2.2 training window, this cap does not fire on the fresh optimizer step and fires on roughly one third of the stale second steps.

C Robust advantage processing

Section 4.3 converts raw pair differences into a fixed replay weight. For each reward dimension d, we apply symmetric winsorization and an interquartile-range scale:

$$\widetilde{A}_{i,d} = \text{clip}\Big(A_{i,d}^{\text{pair}}, L_d, U_d\Big), \qquad \text{Scale}_d = \max\left\{\frac{\text{Quantile}_{0.75}(\widetilde{A}_{\cdot,d}) - \text{Quantile}_{0.25}(\widetilde{A}_{\cdot,d})}{1.349}, \epsilon_{\text{s}}\right\},$$
$$\widehat{A}_i = \text{sg}\left(\sum_{d \in \mathcal{D}} w_d \frac{\widetilde{A}_{i,d}}{\text{Scale}_d}\right). \tag{14}$$

Here L_d and U_d are the symmetric winsorization thresholds, Quantile_α is the empirical quantile at probability α, $\epsilon_{\text{s}} > 0$ is a numerical floor, and w_d weights reward dimension $d \in \mathcal{D}$. The constant 1.349 converts the interquartile range into a standard-deviation scale under a Gaussian reference. The stop-gradient operator $\text{sg}(\cdot)$ makes $\widehat{A}_i$ a fixed replay weight. Because the thresholds are symmetric and the processing is shared across both signs of every pair, it preserves pair antisymmetry up to numerical precision. Our experiments use a single HPSv3 dimension, so $\mathcal{D}$ contains one element.

D Evaluation protocols

Wan2.2 dense probe. The reward scores the rendered video rather than the weights. The same pretraining checkpoint can therefore produce different step-zero scores under different rollout backends. On the 61-prompt probe, the RPO backend starts at 4.92 raw HPSv3, whereas the DanceGRPO backend starts at 5.53. Figure 4b therefore plots every dense-model number as a gain over the run's own step-zero score, which cancels this constant renderer offset. Subtler backend effects remain possible.

Comparison with DanceGRPO. The DanceGRPO run uses the same generator, reward, and probe prompts as RPO. Four differences remain. First, DanceGRPO is full-model GRPO with $G = 8$ and a stochastic SDE sampler, whereas RPO routes a single denoising transition with $G = 2$. Second, the two runs use different rollout backends. Third, the two runs record dense evaluation through different evaluation code, although both measure raw HPSv3 under dense inference on the same prompts. Fourth, the DanceGRPO run follows a longer 120-step schedule, so Figure 4 shows only its early phase.

VBench. We evaluate all 16 VBench dimensions on the standard prompt suite. Each Wan2.2 model generates 6,220 videos with 13 frames at 704×1280 and 24 fps, using a deterministic 50-step ODE sampler with classifier-free guidance scale 5.0. The four Wan2.2 models share prompts and initial noise. CogVideoX-5B and HunyuanVideo use the same prompts, frame count, denoising steps, and scoring, but they keep their native resolution and guidance. Their rows therefore provide a reference under matched content and scoring, although their generation settings and training compute differ. Training GPU-hours use a common accounting scope and exclude VBench evaluation.

Stable Diffusion 3.5. During training, dense evaluation uses a 28-step sampler and reports raw HPSv3 on 64 held-out prompts. Early runs draw this subset with the training seed. For controlled seed comparisons, we instead use a fixed 64-prompt file and a fixed validation noise seed. The DrawBench evaluation in Figure 3 uses 1,000 images from 200 prompts and 5 seeds, generated without classifier-free guidance.

E EXTENDED DISCUSSION

Known failure modes. Two failure modes remain open. The first is a dense-gain fade under certain routing schedules. In these runs, the dense-inference gain peaks early and then decays over later training steps. Diagnostic runs with constant-offset Bernoulli configurations show that the fade begins at a sharp drop in stale policy displacement rather than at reward degradation. However, the mechanism is not fully isolated. The second is reward hacking against the frozen preference model. This risk is general to online RL for generative models and is not specific to RPO. Both failure modes suggest that a scalar terminal reward is a coarse supervisor for a discrete routing action.

Estimator scope. The recorded-action likelihood in Equation 6 is exact, but the resulting update is a route-channel semi-gradient. It omits the pathwise gradient of the generator through later denoising transitions and through the frozen reward model. The fixed offset also sets only an initial sparsity operating point. It does not preserve the expected keep count as the attention affinities evolve, so reward improvement can coincide with a change in mask density.

Extensions. Two directions extend the method naturally. Multi-step routing would route more than one denoising transition per rollout. It needs a coupled-pair estimator to keep the matched-pair advantage well defined. Larger groups would give a leave-one-out baseline with more members. They trade part of the pair's variance cancellation for a more stable advantage scale. Both extensions are compatible with the estimator as stated.