DATA-REGULARIZED REINFORCEMENT LEARNING FOR DIFFUSION MODELS AT SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

Aligning generative diffusion models with human preferences via reinforcement learning (RL) is critical yet challenging. Most existing algorithms are often vulnerable to reward hacking, such as quality degradation, over-stylization, or reduced diversity. Our analysis demonstrates that this can be attributed to the inherent limitations of their regularization, which provides unreliable penalties. We introduce Data-regularized Diffusion Reinforcement Learning (DDRL), a novel framework that uses the *forward* KL divergence to anchor the policy to an off-policy data distribution. Theoretically, DDRL enables robust, unbiased integration of RL with standard diffusion training. Empirically, this translates into a simple yet effective algorithm that combines reward maximization with diffusion loss minimization. With over a million GPU hours of experiments and ten thousand double-blind human evaluations, we demonstrate on high-resolution video generation tasks that DDRL significantly improves rewards while alleviating the reward hacking seen in baselines, achieving the highest human preference and establishing a robust and scalable paradigm for diffusion post-training.

1 INTRODUCTION

Recent developments in generative models, including denoising diffusion models (Nichol & Dhariwal, 2021; Song et al., 2020b;a; Esser et al., 2024) and the flow-based counterparts (Lipman et al., 2022; Liu et al., 2022; Song et al., 2023), have demonstrated their remarkable capabilities in high-quality images and videos generation (Agarwal et al., 2025; Ali et al., 2025). With impressive scalability to billions of data points (Schuhmann et al., 2022), diffusion models have the potential to serve as foundational generative models across a wide range of applications (Song & Ermon, 2019; Nichol & Dhariwal, 2021; Rombach et al., 2022; Ye et al., 2024). To align these models with human preferences, reinforcement learning (RL) has emerged as a powerful post-training paradigm. Algorithms like RLHF and GRPO (Achiam et al., 2023; Ouyang et al., 2022; Shao et al., 2024) post-train a generative model p_θ from a reference model p_{ref} to maximize the rewards that represent human preferences, while regularizing the *reverse* KL divergence $D_{\text{KL}}(p_\theta \| p_{\text{ref}})$ to control their difference. Thereafter, they have been adopted to post-train diffusion models as well (Black et al., 2023; Wallace et al., 2024; Liu et al., 2025a; Xue et al., 2025; Gao et al., 2025).

While critical, applying RL to diffusion models at scale is challenging. Most vision reward models are learned from human preferences data and are non-verifiable: they accurately reflect human preferences only in a neighborhood of the training data manifold. As p_θ is optimized for reward maximization, its multi-step, Markovian sampling procedure can easily exploit out-of-distribution (OOD) regions away from the data manifold, where the reference model is barely trained and fails to provide accurate regularization. This issue lies in the adoption of on-policy samples for regularization, and makes existing algorithms vulnerable to reward hacking (Skalse et al., 2022; Zhai et al., 2025), a phenomenon where models generate human-unpreferred samples that nevertheless receive high rewards, resulting in issues like diversity decline, quality degradation, and over-stylization. We illustrate in Fig. 1 the low-quality, unrealistic video generations that nevertheless achieve a high reward, despite the KL divergence remaining controlled (Fig. 2). Similarly, Fig. 4 illustrates the over-simplified images generated by existing methods.

We are motivated by the fundamental question: *can we design a more principled RL algorithm for diffusion models that is effective and inherently robust to hacking?* To address the issue caused

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: Generated videos after RL with different algorithms from the same checkpoint. DDRL improves rewards by generating prompt-aligned, realistic videos, whereas baselines increase rewards with over-colorized, over-stylized, and unrealistic videos.

by on-policy regularization, we introduce **Data-regularized Diffusion Reinforcement Learning** (DDRL), a novel framework that anchors to off-policy data distributions. Intuitively, DDRL maximizes relative rewards while regularizing the *diffusion loss* on a dataset that is either real or synthetic to prevent unreasonable generations. Theoretically, DDRL stems from the *forward* KL divergence, which is equivalent to the diffusion training objective and guarantees unbiased RL optimization. It is robust to reward hacking and seamlessly integrates RL with supervised learning (SFT), scaling effectively by balancing on-policy exploration with off-policy data quality.

We conduct extensive experiments on large-scale video and image generation tasks, utilizing over one million H100 GPU hours and ten thousand double-blind human voting results. Under all settings with different rewards, base models, and conditions, DDRL consistently improves rewards while also maintaining the highest human voting preferences, while existing baselines either fail to increase rewards (with large regularization and early stopping) or provide hacked generations that are consistently not favored by voters. Our empirical analysis shows that they hack by trading off one reward score with another, while DDRL can bring Pareto improvement. Furthermore, we ablate DDRL and demonstrate its potential to integrate SFT and RL into a single stage, achieving comparable quality with significantly higher data efficiency. We also justify the effectiveness of DDRL when using synthetic data for regularization.

2 Backgrounds & Preliminaries

Diffusion generative models learn a conditional data distribution $p_{\text{data}}(\mathbf{x}|c)$ by reversing a fixed noise-adding process[1], where c is a given condition (Nichol & Dhariwal, 2021; Song et al., 2020b;a; Esser et al., 2024). Given a noise schedule $\{\beta_t\}_{t=1}^{T} \in (0,1)$, the forward process is defined as

$$\begin{aligned} q(\mathbf{x}_0|c) &= p_{\text{data}}(\mathbf{x}|c), \\ q(\mathbf{x}_t|\mathbf{x}_{t-1}) &= \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}). \end{aligned} \tag{1}$$

[1]For simplicity, we formulate under the diffusion models, while our method applies directly to other variants such as flow models (Lipman et al., 2022; Liu et al., 2022) and consistency models (Song et al., 2023).

A diffusion model $\boldsymbol{\epsilon}_\theta$ learns the distribution by

$$p_\theta(\mathbf{x}_{0:T}|c) = p_\theta(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, c), \tag{2}$$

where $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{x}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t, t, c), \sigma_t^2\mathbf{I})$. Here $\boldsymbol{\mu}_\theta$ can be computed from $\mathbf{x}_t$ and the model-predicted noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c)$. The goal is to minimize the KL divergence between $q(\mathbf{x}_{0:T}|c)$ and $p_\theta(\mathbf{x}_{0:T}|c)$, i.e.

$$D_{\mathrm{KL}}(q(\cdot|c)\|p_\theta(\cdot|c)) \triangleq \int_{\mathbf{x}_{0:T}} q(\mathbf{x}_{0:T}|c)\log\frac{q(\mathbf{x}_{0:T}|c)}{p_\theta(\mathbf{x}_{0:T}|c)}\mathrm{d}\mathbf{x}. \tag{3}$$

To this end, the model is trained by minimizing:

$$\mathcal{L}(\theta; p_{\text{data}}) = \mathbb{E}_{t,\mathbf{x}_t\sim q(\mathbf{x}_t|c)}\left[w_t\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c) - \boldsymbol{\epsilon}\|^2\right], \tag{4}$$

where q is constructed from p_{data} as defined above, $\boldsymbol{\epsilon}$ is the Gaussian noise used to sample $\mathbf{x}_t$ from q, and w_t is θ-independent coefficient calculated from β_t. As a classical result, Eq. (4) can be proven equivalent to Eq. (3) up to a θ-independent constant, and thus both objectives share the same optimum at which $p_\theta(\mathbf{x}_0|c)$ equals $p_{\text{data}}(\mathbf{x}|c)$.

Reward Models. Similar to RL for language models (Ouyang et al., 2022; Shao et al., 2024; Rafailov et al., 2023), we are given a reward model $r(\mathbf{x}_0, c)$, which represents the user's preference to a sample $\mathbf{x}_0$. Remarkably, r can be any rule-based function or neural network, and it can represent a variety of properties such as a video's smoothness, an image's physical reasonability, as well as a sample's prompt adherence (Liu et al., 2025a; Xue et al., 2025; Wallace et al., 2024). In the vision domain, most rewards are *non-verifiable*, i.e., they are an approximation of the underlying true rewards $r^*(\mathbf{x}_0, c)$ that we have no access to, and are only accurate in a neighborhood of the real data manifold. In other words, a higher reward does not necessarily imply a better sample.

Reinforcement learning for diffusion. The objective is to post-train p_θ to generate samples from

$$p_\theta(\mathbf{x}_0|c) \propto p_{\text{data}}(\mathbf{x}_0|c)\exp\left(r(\mathbf{x}_0, c)/\beta\right), \tag{5}$$

where β is a temperature parameter. Existing literature shows that the denoising process can be formulated as a Markov Decision Process (MDP) (Black et al., 2023), making it possible to post-train the diffusion model via RL algorithms. Specifically, the diffusion model $\boldsymbol{\epsilon}_\theta$ is the policy whose action is to sample $\mathbf{x}_{t-1}$ from current state $(\mathbf{x}_t, t, c)$, reaching state $(\mathbf{x}_{t-1}, t-1, c)$. The reward $r(\mathbf{x}_0, c)$ is provided at the terminal step $t = 0$. The diffusion policy is post-trained by maximizing

$$\mathcal{J}_{\text{RL}}(p_\theta) = \mathbb{E}_{p_\theta(\mathbf{x}_0|c)}\Big[r(\mathbf{x}_0, c)/\beta\Big] - D_{\mathrm{KL}}(p_\theta||p_{\text{ref}}). \tag{6}$$

Notice that $p_{\text{ref}}(\mathbf{x}_{0:T}|c)$ in the formula typically represents a reference diffusion model that is trained on p_{data}. Classical RL theory show that its optimum satisfies $p_\theta(\mathbf{x}_0|c) \propto p_{\text{ref}}(\mathbf{x}_0|c)\exp(r(\mathbf{x}_0, c)/\beta)$ (Sutton et al., 1998), which approximates Eq. (5) if p_{ref} is close to p_{data}.

Optimization. Eq. (6) leverages *reverse* KL divergence to constrain the optimization. Due to the Markovian property, it can be estimated by

$$D_{\mathrm{KL}}(p_\theta||p_{\text{ref}}) = \mathbb{E}_{\mathbf{x}_{0:T}\sim p_\theta}\sum_t D_{\mathrm{KL}}(p_\theta(\cdot|\mathbf{x}_t, c)\|p_{\text{ref}}(\cdot|\mathbf{x}_t, c)), \tag{7}$$

where $p(\cdot|\mathbf{x}_t, c)$ is the marginal distribution over $\mathbf{x}_{t-1}$. When both p_θ and p_{ref} are diffusion models, the divergence can be analytically calculated by $\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c) - \boldsymbol{\epsilon}_{\text{ref}}(\mathbf{x}_t, t, c)\|^2$. For the reward maximization term, classical approaches such as REINFORCE (Sutton et al., 1999) points out that

$$\begin{aligned}\nabla_\theta\mathbb{E}_{p_\theta}[r(\mathbf{x}_0, c)] &= \int_{\mathbf{x}_{0:T}} \nabla_\theta p_\theta(\mathbf{x}_{0:T}|c)r(\mathbf{x}_0, c)\mathrm{d}\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x}_{0:T}\sim p_\theta(\cdot|c)}[\nabla_\theta\log p_\theta(\mathbf{x}_{0:T}|c)r(\mathbf{x}_0, c)].\end{aligned}$$

To optimize $\mathbb{E}[r(\mathbf{x}_0, c)]$, we can compute its gradient from the log probability of samples from p_θ. In the context of diffusion, $\log p_\theta(\mathbf{x}_{0:T}|c)$ can be calculated by decomposing it into the sum of log probabilities at each step t. Empirically, GRPO (Shao et al., 2024) introduces relative advantages to stabilize the training. Specifically, for each condition c, the algorithm will sample a *rollout group* of $\mathbf{x}_0^1, \cdots, \mathbf{x}_0^N$ from $p_\theta(\mathbf{x}_0|c)$ and replace $r(\mathbf{x}_0^n)$ by the advantage term $A(\mathbf{x}_0^n, c) \triangleq \frac{r(\mathbf{x}_0^n, c) - \mathrm{mean}}{\mathrm{std}}$, where $\mathrm{mean}, \mathrm{std}$ are within the group. While not theoretically grounded, advantages enable models to focus on comparative qualities rather than on absolute reward values, and are widely adopted.

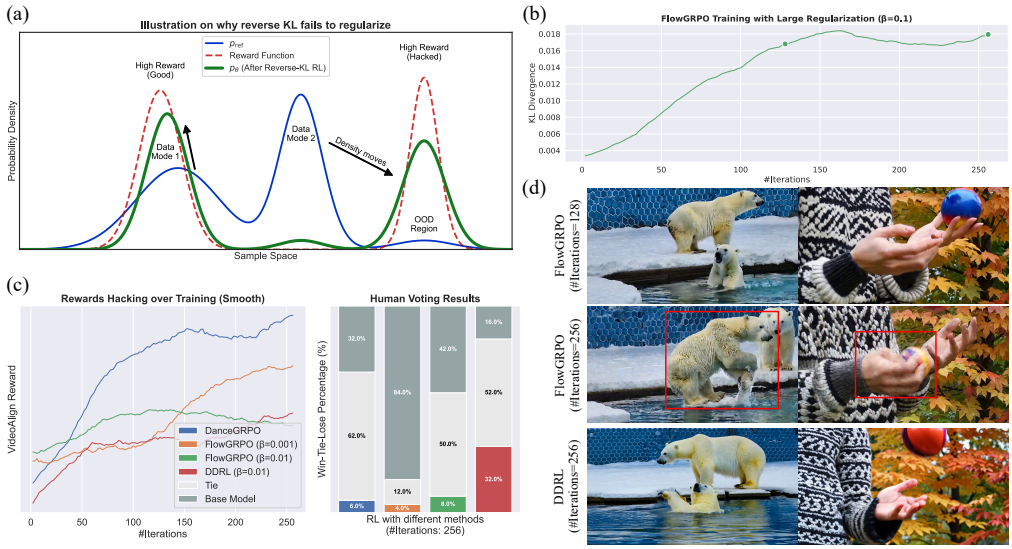

Figure 2: Why does reverse KL fail to regularize diffusion RL? Illustration: (a) high-reward regions exist outside the data manifold due to imperfect rewards; reverse KL, computed on on-policy samples, fails to penalize the policy in these OOD regions where p_{ref} is unreliable. Evidence: (b) Using FlowGRPO with a large KL coefficient $\beta = 0.1$, the divergence remains stable after 128 iterations; However, the generated videos progressively exhibit noise textures and artifacts (red boxes in (d)) as the training continues to 256 iterations. Quantitatively, (c) double-blind human voting reveals that voters consistently prefer the base model over these post-trained baselines, despite the latter having higher rewards. By contrast, DDRL improves both rewards and human preference.

3 Methodology

3.1 Motivation

In generative modeling, reward maximization is not the purpose, but rather the means to learn p_θ that reweighs the original p_{ref} towards human preferences. Existing algorithms anchor p_θ to p_{ref} by either leveraging the reverse KL divergence (FlowGRPO (Liu et al., 2025a)) or by adopting heuristics such as a fixed initial random seed (DanceGRPO (Xue et al., 2025)). Unfortunately, they exhibit various types of hacking behaviors due to their intrinsically ineffective regularization. To illustrate, we post-train Cosmos2.5 (Agarwal et al., 2025), a physical world model that aims to generate physically reasonable videos, using different methods. As shown in Fig. 1, DanceGRPO optimizes rewards by generating over-colorized, over-stylized videos that are unrealistic, ignoring the prompt requirements, while FlowGRPO generates blurry, jittering videos that are visually unacceptable. A similar phenomenon is observed in image generation (Fig. 4), where existing methods exploit text-rendering rewards by generating overly simplified images that lose realism.

Why reverse KL fails to alleviate reward hacking? We illustrate in Fig. 2(a) that this is a joint effect of non-verifiable rewards, imperfect reference models, and ineffective on-policy regularization. Since reward models and p_{ref} are trained networks, high-reward regions can exist outside the data manifold, and the reverse KL computed on on-policy samples from p_θ (Eq. (7)) cannot penalize the policy in these OOD regions where p_{ref} is barely trained and is unreliable. When p_θ is driven toward reward maximization, it gradually shifts intermediate states $\mathbf{x}_t$ away from the data manifold that nevertheless could achieve higher rewards. The multi-step Markovian sampling exacerbates this: per-step divergence may appear small while the accumulated drift is significant.

Empirical evidence? Fig. 2(b) shows the change of KL divergence during training when using a large coefficient $\beta = 0.1$. While it remains stable after 128 iterations, Fig. 2(d) reveals that generated videos progressively exhibit noise textures and artifacts (highlighted in red boxes) as the training continues to 256 iterations, confirming that controlling the reverse KL provides a false sense of security. Quantitatively, Fig. 2(c) shows that while baseline methods increase rewards, this improvement comes at the cost of human preference: double-blind human voters consistently prefer the base model's generations despite the post-trained baselines achieving higher rewards.

Easy fix? While heuristics such as early stopping, delayed samplers, or conservative hyperparameters can partially mitigate reward hacking, they inevitably constrain legitimate reward gains. For instance, training FlowGRPO for only 128 iterations yields comparable human preference to the base model, but the reward improvement becomes marginal. Our main experiments (Tab. 2) adopt this conservative setting to limit baselines' hacking, yet the dilemma persists. In addition, Tab. 4 also indicates that simply maintaining an old policy p_{old} as the sampler could not address the problem. Fundamentally, we call for a more principled post-training framework for diffusion models.

3.2 THE DDRL FRAMEWORK

A straightforward solution is to regularize with samples from the reference model (off-policy samples), i.e., to replace $D_{\text{KL}}(p_\theta \| p_{\text{ref}})$ with the forward KL divergence $D_{\text{KL}}(p_{\text{ref}} \| p_\theta)$, which is computed using $\mathbf{x}_t$ sampled from p_{ref}. This naive replacement will lead to an incorrect optimum different from Eq. (5), with more details in GX-Chen et al. (2025). To effectively leverage the off-policy regularization while maintaining the precise objective, this paper proposes a novel, principled, and robust forward KL divergence-based RL framework, named **Data-regularized Diffusion Reinforcement Learning** (DDRL). Specifically, given a p_{ref}, DDRL maximizes $\mathcal{J}_{\text{DDRL}}(p_\theta)$ defined as:

$$\mathbb{E}_{p_\theta(\mathbf{x}_0|c)}\Big[\lambda\Big(\frac{r(\mathbf{x}_0, c) - Z}{\beta}\Big)\Big] - D_{\text{KL}}(\tilde{p}_{\text{ref}} || p_\theta), \tag{8}$$

where $\lambda(x) = -\exp(-x)$ is a monotonic transform function, and constant $Z = \beta \log \mathbb{E}_{p_{\text{ref}}(\mathbf{x}_0|c)}[\exp\{r(\mathbf{x}, c)/\beta\}]$ is an "average reward" that converts absolute rewards to relative advantages as inspired by GRPO. The core difference lies in $\tilde{p}_{\text{ref}}$, which is defined as the *forward distribution* specified in Eq. (1):

$$\tilde{p}_{\text{ref}}(\mathbf{x}_0|c) = p_{\text{ref}}(\mathbf{x}_0|c).$$
$$\tilde{p}_{\text{ref}}(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}).$$

Notably, Eq. (8) differs from Eq. (6) in two fundamental ways. First, it adopts the forward KL divergence, enabling regularization with off-policy samples from p_{ref}; Second, it is based on the forward process $\tilde{p}_{\text{ref}}$, which is different from the denoised process p_{ref}. Below, we show that this formulation elegantly connects to diffusion training, post-train integration, and classical RL objectives.

Further transformation. Using $D_{\text{KL}}(\tilde{p}_{\text{ref}} \| p_\theta)$ not only targets the hacking phenomenon brought by on-policy sampling, but also bridges to Eq. (3), where q is similarly defined from the data distribution. This connection enables us to further transform $D_{\text{KL}}(\tilde{p}_{\text{ref}} || p_\theta)$ to the standard diffusion loss $\mathcal{L}(\theta; \tilde{p}_{\text{ref}}(\mathbf{x}_0|c))$, similar to the conversion from Eq. (3) to Eq. (4). Thereafter, we only use the reference model for *sampling*, instead of for computing intermediate variables as required in classical RL objective Eq. (6), and it can flexibly be *any* sampler. Specifically, when the reference model is fine-tuned with a high-quality, accessible dataset p_{data}, users can directly compute the diffusion loss on this dataset. In contrast, when the dataset used to trained the reference model is not available, users can generate synthetic data upfront and post-train with this data. This leads to an equivalent objective $\tilde{\mathcal{J}}_{\text{DDRL}}$, where we replace $\tilde{p}_{\text{ref}}$ with $\tilde{p}_{\text{data}}$ to emphasize its nature as a sampler:

$$\mathbb{E}_{p_\theta(\mathbf{x}_0|c)}\Big[\lambda\Big(\frac{r(\mathbf{x}_0, c) - Z}{\beta}\Big)\Big] - \mathcal{L}(\theta; \tilde{p}_{\text{data}}(\mathbf{x}_0|c)). \tag{9}$$

Theorem 1. *Maximizing Eq.* (9) *is equivalent to maximizing Eq.* (8)*, with optimal policy*

$$p^*_\theta(\mathbf{x}_0|c) \propto \tilde{p}_{data}(\mathbf{x}_0|c) \exp\left(r(\mathbf{x}_0, c)/\beta\right). \tag{10}$$

The proof is in the appendix. Theorem 1 demonstrates that DDRL enables direct regularization with data while maintaining the correct optimal distribution (Eq. (6)). Notice that $\tilde{p}_{\text{data}}$ can generally be a real data distribution or a synthetic distribution such as $p_{\text{ref}}(\mathbf{x}_0|c)$, and the optimum becomes $p_{\text{ref}}(\mathbf{x}_0|c) \exp(r(\mathbf{x}_0, c)/\beta)$ as in traditional RL. It enables direct utilization of real data in RL through diffusion loss, which is not possible in reverse KL-based approaches.

DDRL as a post-train integration. Eq. (9) points out the intrinsic nature of DDRL as an elegant integration of supervised fine-tuning (via diffusion loss minimization) and reinforcement learning (via reward maximization). Indeed, this combination has also been studied in recent LLM post-training works (Chen et al., 2025; Lv et al., 2025), offering cross-justification of over methodology. Theorem 1 theoretically justifies for the first time the merit of combining two post-training stages, and its proof can be transferred to auto-regressive generative models as well. In short, DDRL not only offers a theoretical formulation but also leads to an empirically robust and unified implementation.

Algorithm 1 Data-regularized Diffusion Reinforce Learning (**DDRL**)

Input: Initial policy θ, data sampler $\tilde{p}_{\text{data}}$, reward model r, training steps $\mathcal{T} \subset [0, \cdots, T-1]$, coefficient β, rollout size N.

for each training iteration **do**
  Sample c, sample $\tilde{\mathbf{x}}_0 \sim \tilde{p}_{\text{data}}(\cdot|c)$
  Rollout $\{\mathbf{x}_t^n\}_{n=1}^N$ with p_θ, get reward $r^n = r(\mathbf{x}_0^n, c)$, compute $A^n = \frac{r^n - \text{mean}}{\beta \cdot \text{std}}$.
  for $t \in \mathcal{T}$ **do**
    Sample $\boldsymbol{\epsilon}^n$, get noisy $\tilde{\mathbf{x}}_t^n$ based on $(\tilde{\mathbf{x}}_0, \boldsymbol{\epsilon}^n, t)$.
    Optimize $L_t^n = \|\boldsymbol{\epsilon}_\theta(\tilde{\mathbf{x}}_t^n, t, c) - \boldsymbol{\epsilon}^n\| - A^n \log p_\theta(\mathbf{x}_t^n | \mathbf{x}_{t+1}^n, c)$ ▷ From $Eq.$ (9)
  end for
  Update θ via gradient descent w.r.t. all L_t^n.
end for

3.3 Practical Implementation

We present the pseudo-code of our practical DDRL implementation in Algorithm 1. For diffusion loss minimization, similar to standard diffusion training, we omit the coefficient w_t for stability. For reward maximization, we omit the monotonic transformation $\lambda(\cdot)$ such that the training reward scales consistently and comparably with baselines. We compute Z using the average reward within the rollout group, and additionally divide the standard deviation.

Timesteps & Guidance. We use $\mathcal{T}$ to denote the timesteps subset where DDRL optimizes the objective. Similar to the observation in Xue et al. (2025), we find that optimizing half of the steps is sufficient, and DDRL simply optimizes once every two steps. In terms of guidance, FlowGRPO utilizes classifier-free guidance (CFG) while DanceGRPO does not use CFG whenever possible. Following the diffusion training standard and recent studies (Zheng et al., 2025), DDRL does not adopt CFG, and drops the condition c when calculating the diffusion loss with 20% probability.

Memory & Efficiency. Both real and synthetic data can be collected upfront, so DDRL needs neither a reference model nor an old model during training, thus saving memory by a large margin. That said, all importance sampling-based reward maximization alternatives are naturally compatible with DDRL, as the core improvement lies in adopting the diffusion loss. In terms of efficiency, the vanilla version of Algorithm 1 optimizes the diffusion loss of each single data point $N|\mathcal{T}|$ times (N rollouts, $|\mathcal{T}|$ steps). We show in Tab. 3 that this can be substantially reduced to N times by sampling t independently and optimizing the diffusion loss only once. As a result, the number of network evaluations (NFEs) per data of DDRL is similar to DanceGRPO and half compared to FlowGRPO.

4 Experiments

4.1 Setting

Models. For video generation, we use Cosmos2.5 (Ali et al., 2025; Agarwal et al., 2025) 2B and 14B as the base models due to their capability in generating physically reasonable scenarios. We use two reward models: VideoAlign (Liu et al., 2025b) finetunes Qwen2-vl (Wang et al., 2024) to evaluate the visual quality, motion quality, and text alignment of a video, and we use the average of three scores as our reward. VBench (Huang et al., 2024) contains a wide range of model-based metrics. We use the average of subject consistency, background consistency, motion smoothness, aesthetic quality, and imaging quality (five scores).

Training. We train with 256 H100 GPUs for Cosmos2.5-2B and 1024 GPUs for Cosmos2.5-14B. We implement baselines based on original papers and codebases, with hyperparameters specified in Tab. 1. Notice that a larger $|\mathcal{T}|$ naturally results in better optimization. For all methods, we accumulate the gradient over all $t \in \mathcal{T}$ and conduct one full parameter update at the end of each iteration. We consume approximately one million GPU hours. More details are presented in the appendix.

Table 1: Hyperparameters table.

Method	T	$\lvert\mathcal{T}\rvert$	β	use p_{ref}?
DanceGRPO	20	12	0	×
FlowGRPO	10	10	$[0.001, 0.1]$	✓
DDRL	20	10	0.01	×

Table 2: DDRL aligns model with human preference better without severe reward hacking. "T2V" and "I2V" stand for "Text to Video" and "Image to Video". Δ-Vote denotes the difference in human preference from pairwise video comparisons, calculated as the win percentage of the corresponding method minus the win percentage of DDRL. We mark a reward if its generated videos are less preferred by humans but the reward is higher in this setting, which generally indicates reward hacking.

Base Model	Method	VideoAlign T2V		VideoAlign I2V		VBench T2V		VBench I2V	
		Δ-Vote↑ (%)	**Reward**	**Δ-Vote↑ (%)**	**Reward**	**Δ-Vote↑ (%)**	**Reward**	**Δ-Vote↑ (%)**	**Reward**
Cosmos2.5-2B	w/o RL	-22.9	0.408	-4.2	0.079	-8.9	0.830	-15.7	0.810
	DanceGRPO	-10.5	0.715	-6.4	0.254	-6.9	0.849	-8.0	0.822
	FlowGRPO	-6.7	0.408	-13.0	0.069	-4.2	0.830	-9.6	0.807
	DDRL	0	0.604	0	0.177	0	0.842	0	0.819
Cosmos2.5-14B	w/o RL	-8.4	0.359	-12.5	0.058	-4.5	0.820	-4.5	0.813
	DanceGRPO	-4.9	0.494	-5.9	0.160	-3.1	0.834	-4.2	0.818
	FlowGRPO	-6.7	0.476	-11.6	0.128	-7.3	0.829	-5.4	0.808
	DDRL	0	0.555	0	0.134	0	0.835	0	0.819

Evaluation. All reported measurements are tested on a PAI-Bench (Zhou et al., 2025), a high-quality test dataset containing five categories (human, industry, robotics, driving, others) and 1044 prompts. For all methods, we generate videos using the trained models under the T2V and I2V (same conditional images) settings with classifier-free guidance. Since we aim to analyze the hacking phenomenon, comparing only the reward values becomes misleading, and we conduct large-scale double-blind **human voting** across fifteen researchers. For each experimental setting, DDRL is compared separately against FlowGRPO, DanceGRPO, and the base model. We collect human preference data via a side-by-side comparison website, where participants are asked to choose the better-quality video (or tie). We report the difference in the preference percentages (denoted as **Δ-Vote**): if DDRL wins on 40.0% videos and loses on 18.0%, then the reported Δ-Vote is 22.0. In total, we gather ∼9600 comparison data as the ground-truth to reflect method quality (Skalse et al., 2022): a good method to increase rewards and improve/maintain human preference simultaneously.

4.2 Superiority of DDRL

Over 15% of the computation is spent on selecting the best hyperparameters for baseline methods to ensure that large-scale human voting is informative. This includes the number of training iterations and the coefficient β. As we demonstrate in Fig. 2(b), both DanceGRPO and FlowGRPO (regardless of β ranging across two magnitudes) hack severely and become worse than the base model after training for 256 iterations. To reduce the phenomenon, we train all methods only for 128 iterations and use the exponential moving average (EMA) model weights to generate test videos. In addition, we also carefully sweep the regularization parameter $\beta \in [0.001, 0.1]$ for FlowGRPO, and select $\beta = 0.01$, consistent with their paper. While this value does not prevent reward hacking, it mitigates the most severe artifacts, despite stifling legitimate reward growth. Lastly, for 14B post-training, we reduce the learning rate to 3e-6 for all methods to further reduce baseline hacking.

We present a summary of voting results and rewards under two reward models, two generation settings, and two base models in Tab. 2, with details provided in the appendix. Simultaneously across all settings, DDRL not only improves rewards over the base model, but is also more preferred than any other baselines or the base model, with Δ-Vote always being negative indicating that DDRL-generated videos are consistently more favorable. Importantly, DDRL is the only method that improves rewards in a way that voters prefer, whereas DanceGRPO often hacks the reward models by generating unpreferred videos, which nevertheless receive higher rewards. In addition, the advantage in 14B experiments is smaller due to the use of a smaller learning rate to mitigate baselines' hacking behaviors, and the performance gap will enlarge as we continue post-training.

Reward hacking analysis. We carefully analyze why existing methods do not work well by presenting a breakdown of per-score-increase after RL in Fig. 3. DanceGRPO achieves its high average reward at the cost of sacrificing one score, with text alignment increase being negative after post-training. This trade-off is a signal of reward hacking, indicating that one property is being compromised to exploit the reward. DDRL, by regularizing directly with the data-anchored diffusion loss, achieves a Pareto improvement across all scores. We believe that the ability to enhance alignment without sacrificing other generative properties is a highly appealing and robust feature in RL.

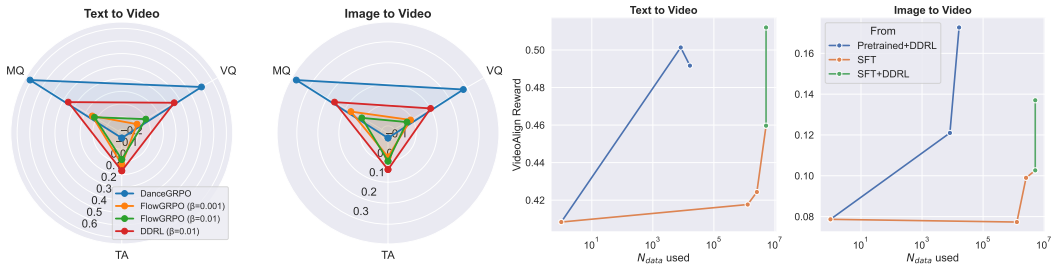

Figure 3: (Left) Breakdown of the increase/decline of text alignment (TA), visual quality (VQ), and motion quality (MQ), after post-training the 2B model with different methods. DanceGRPO shows a 16% (T2V) and 28% (I2V) decrease in TA, despite its average reward being the highest. DDRL is the only algorithm that brings "Pareto improvement". (Right) VideoAlign reward after post-training with DDRL from a pretrained model vs. an SFT model. The former achieves comparably high rewards with significantly higher data efficiency.

Notably, our large-scale human voting is labor-intensive and not always be adoptable. A reliable way to automatically detect reward hacking is highly beneficial. Although we have yet to figure it out, several metrics could be indicative. For instance, if the diffusion loss of a post-trained checkpoint increases by more than 10%, its outputs are mostly hacked; a steep increase in rewards or a sharp decrease in standard deviation are also potential signals.

Ablations on DDRL. We ablate DDRL in Tab. 3. First, while the main comparison uses 128 iterations to favor baselines, DDRL can double training iterations to 256 without reward hacking, continuing to improve rewards (additional human voting confirms no preference degradation). Second, the per-sample overhead of DDRL over FlowGRPO comes from computing the diffusion loss at each timestep in $\mathcal{T}$ ($N|\mathcal{T}$ more where N is the rollout size). By computing diffusion loss only once by randomly sampling $t \in \mathcal{T}$ ("DDRL-Reduced"), the per-sample wall-time drops to 328s (30% faster than FlowGRPO) while achieving rewards comparable to DDRL.

Table 3: Performance, per-sample wall-time (in seconds), and per-sample number of network evaluations (NFEs) with gradients under different algorithms, tested with Cosmos2.5-2B on VideoAlign. Gray rows are copied from Tab. 6.

Method	#Iter	NFEs	Wall-Time	T2V	I2V
DanceGRPO	128	$N\|\mathcal{T}\|$	296	0.715	0.254
FlowGRPO	128	$2N\|\mathcal{T}\|$	467	0.408	0.069
DDRL	128	$2N\|\mathcal{T}\|$	582	0.604	0.177
DDRL	256	$2N\|\mathcal{T}\|$	582	0.627	0.180
DDRL-Reduced	256	$N(1+\|\mathcal{T}\|)$	328	0.636	0.143

4.3 DDRL AS POST-TRAINING INTEGRATION

We have pointed out that DDRL essentially combines RL with SFT as its objective. This raises a natural question: can it be used to directly post-train a model, replacing the standard SFT-then-RL pipeline? We compare two training strategies: (1) a conventional approach where we SFT a vanilla pretrained model on a high-quality dataset for 20K iterations, then post-train with DDRL; (2) we apply DDRL directly to the vanilla pretrained checkpoint using the same dataset. Interestingly, as shown in Fig. 3 (Right), the direct DDRL approach achieves comparably high rewards and low diffusion loss (0.119 v.s. 0.121) to the SFT-then-RL pipeline. While the direct approach saves the entire 20K iteration SFT step, we note that the on-policy RL rollouts—which reuse data multiple times—remain the dominant computational cost in both settings. Nevertheless, this result demonstrates a strong potential to simplify and unify the diffusion post-training paradigm, and we encourage future research to examine this possibility more closely.

4.4 DDRL WITH SYNTHETIC DATA

Below, we show that DDRL works effectively in the absence of real data. Specifically, we apply DDRL to text-to-image (T2I) tasks based on SD3.5-Medium (Esser et al., 2024), with settings aligned with FlowGRPO. We focus on improving the OCR reward for visual text rendering, which is vulnerable to reward hacking. We finetune with LoRA ($\alpha = 64$, $r = 32$) and use synthetic data sampled from the base model for diffusion loss regularization. In addition, we include an off-policy variant of FlowGRPO that maintains an old policy p_{old} for sampling (updated every 4 iterations), and Diffusion-DPO (Wallace et al., 2024) trained on preference pairs to examine whether alternative RL paradigms can mitigate hacking.



Figure 4: Generated images after RL on the OCR reward. DDRL achieves high rewards and maintains fidelity to the base model in terms of style and realism, while others increase rewards by producing cartoonish outputs.

As shown in Figure 4 and Table 4, DDRL exhibits both improved OCR accuracy and over 20% human preference due to its fidelity to the base model's style and realism, even when we purely use synthetic data for regularization. In contrast, baseline methods tend to simplify images by using large, centralized captions. Notably, the off-policy method still produces under-preferred outputs similar to its on-policy counterpart, indicating that simply decoupling the sampling policy does not address the fundamental hacking issue. DPO avoids severe hacking but fails to substantially improve OCR accuracy, likely due to its reliance on offline preference pairs that limit reward optimization. We also demonstrate that DDRL achieves high OOD rewards (Hessel et al., 2021; Kirstain et al., 2023; Xu et al., 2023), which shows its robustness in preserving generation diversity.

Table 4: Human voting results, OCR reward, and three OOD rewards after post-training SD3.5-Medium. We similarly mark the OCR reward if it is achieved by sacrificing human preference, *e.g.*, generations are over-simplified or cartoonish.

Method	Δ-Vote↑	OCR↑	Clip↑	Pick↑	ImgRe↑
w/o RL	-	0.566	0.321	0.865	1.13
DanceGRPO	-53.0	0.846	0.309	0.837	0.80
FlowGRPO	-22.1	0.845	0.313	0.855	1.03
FlowGRPO (off-policy)	-20.9	0.843	0.313	0.857	1.06
Diffusion-DPO	-3.6	0.707	0.318	0.865	1.17
DDRL	0	0.823	0.320	0.865	1.14

5 Related Works & Conclusions

More related works can be found in Section A.

Reward hacking. Reward hacking, or reward overoptimization, is a long-standing challenge closely related to Goodhart's Law (Chrystal et al., 2003)—that a measure ceases to be reliable once it becomes a target. This phenomenon is well-documented in both general AI (Skalse et al., 2022; Clark & Amodei, 2016; Amodei et al., 2016) and RL settings (Karwowski et al., 2023), and remains a primary obstacle for aligning generative models. In diffusion RL, reward hacking manifests as quality degradation, over-stylization, or diversity collapse when naively optimizing imperfect reward models. Several recent works explicitly target this issue: Zhang et al. (2024) identify inductive and primacy biases as contributors to reward over-optimization in diffusion RL; Zhang et al. (2025) propose constrained RL formulations to bound reward exploitation; Kim et al. (2025) and Zhai et al. (2025) develop inference-time alignment strategies that mitigate over-optimization without additional training; and GX-Chen et al. (2025) further demonstrate that standard reverse-KL regularization is inherently prone to mode collapse.

Conclusion. Reward hacking in diffusion RL is an intrinsic issue of on-policy regularization. This paper proposes DDRL, a novel framework with a theoretically robust objective and an empirically simple, stable, and highly scalable algorithm that combines on-policy reward maximization with diffusion loss minimization. Our extensive experiments demonstrate that DDRL improves rewards and aligns with human preference by effectively alleviating reward hacking, establishing a principled and effective foundation for the future of post-training diffusion models.

AI USE STATEMENT

We used SD3.5-Medium to generate synthetic reference images for diffusion-loss regularization experiments. In terms of large language models, we use them to format LaTeX and polish grammar. The authors take responsibility for the final content.

ETHICS STATEMENT

Our human evaluation compares generated videos using the protocol described in the appendix. All participants provided informed consent. Despite improved reward alignment, generated content may still exhibit biases or be misused to create misleading media.

REPRODUCIBILITY STATEMENT

We provide the training procedure in Algorithm 1, experimental settings in Section 4, and the theoretical proof, additional implementation details, and results in the appendices to support reproducibility. In supplementary materials we provide the code to reproduce the results, and the videos related to figures in the main paper.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, et al. Cosmos-reason1: From physical common sense to embodied reasoning. *arXiv preprint arXiv:2503.15558*, 2025.

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

Liang Chen, Xueting Han, Li Shen, Jing Bai, and Kam-Fai Wong. Beyond two-stage training: Cooperative sft and rl for llm reasoning. *arXiv preprint arXiv:2509.06948*, 2025.

K Alec Chrystal, Paul D Mizen, and PD Mizen. Goodhart's law: its origins, meaning and implications for monetary policy. *Central banking, monetary theory and practice: Essays in honour of Charles Goodhart*, 1:221–243, 2003.

Jack Clark and Dario Amodei. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

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

Anthony GX-Chen, Jatin Prakash, Jeff Guo, Rob Fergus, and Rajesh Ranganath. Kl-regularized reinforcement learning is designed to mode collapse. *arXiv preprint arXiv:2510.20817*, 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.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar Skalse. Goodhart's law in reinforcement learning. *arXiv preprint arXiv:2310.09144*, 2023.

Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. *arXiv preprint arXiv:2501.05803*, 2025.

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. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Yaron Lipman, Ricky TQ 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, Mingwu Zheng, Xiele Wu, Qiulin Wang, Wenyu Qin, Menghan Xia, et al. Improving video generation with human feedback, 2025. *URL https://arxiv. org/abs/2501.13918*, 2025b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Lixuan He, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, et al. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*, 2025.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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*, pp. 8228–8238, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Y Zou, and Stefano Ermon. Tfg: Unified training-free guidance for diffusion models. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024.

Kevin Zhai, Utsav Singh, Anirudh Thatipelli, Souradip Chakraborty, Anit Kumar Sahu, Furong Huang, Amrit Singh Bedi, and Mubarak Shah. Mira: Towards mitigating reward hacking in inference-time alignment of t2i diffusion models. *arXiv preprint arXiv:2510.01549*, 2025.

Ziyi Zhang, Sen Zhang, Yibing Zhan, Yong Luo, Yonggang Wen, and Dacheng Tao. Confronting reward overoptimization for diffusion models: A perspective of inductive and primacy biases. *arXiv preprint arXiv:2402.08552*, 2024.

Ziyi Zhang, Sen Zhang, Li Shen, Yibing Zhan, Yong Luo, Han Hu, Bo Du, Yonggang Wen, and Dacheng Tao. Aligning text-to-image diffusion models with constrained reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 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.

Fengzhe Zhou, Jiannan Huang, Jialuo Li, and Humphrey Shi. Physical ai bench: A comprehensive benchmark for physical ai generation and understanding, 2025. URL `https://github.com/SHI-Labs/physical-ai-bench`.

A More Related Works

RL and guidance for diffusion models. Both RL and guidance align generative models with human preferences by targeting $p_{\text{ref}}(\mathbf{x}_0|c)\exp(r(\mathbf{x}_0,c)/\beta)$. On the guidance side, classifier guidance (Dhariwal & Nichol, 2021) trains a time-dependent classifier on noisy variables, classifier-free guidance (CFG) (Ho & Salimans, 2022) injects guidance without an explicit classifier and has become a *de facto* standard, and training-free guidance (Ye et al., 2024) enables controllable generation with off-the-shelf reward models at inference time. On the RL side, DDPO (Black et al., 2023) formulates diffusion denoising as an MDP and applies policy gradient methods, while DPO (Rafailov et al., 2023) and its diffusion variant (Wallace et al., 2024) bypass explicit reward modeling by learning from pairwise preferences. More recently, GRPO (Shao et al., 2024) introduces group relative advantages for stable optimization, and DanceGRPO (Xue et al., 2025) and FlowGRPO (Liu et al., 2025a) scale diffusion RL to large video and flow-matching models, with applications in production systems (Gao et al., 2025). While guidance methods require differentiable rewards but no training, RL methods require training but work with arbitrary black-box rewards. A deeper unification of the two paradigms remains to be explored.

Integrating SFT and RL. Generative models are typically post-trained through a multi-stage pipeline, where supervised fine-tuning (SFT) uses a high-quality dataset to improve the model and RL subsequently optimizes for a reward that captures human preferences. DDRL reveals the possibility of integrating the two stages for diffusion models: since its regularization is equivalent to a diffusion training loss on data, SFT and RL can be unified into a single objective. Similar attempts have recently been made for LLMs (Chen et al., 2025; Lv et al., 2025), although these remain empirical and preliminary. Our study provides theoretical justification for this integration, showing that it benefits both model robustness and data efficiency. Notice that, while forward KL is classically associated with mode-covering and potential loss of sharpness, this concern arises primarily when the model has limited expressivity. Diffusion models are expressive approximators natively trained with the forward KL (the denoising objective), and the exploration–regularization trade-off controlled by β universally exists regardless of KL direction.

B Proof of Theorem 1

Part 1: Optimal solution. We derive the optimal policy for the theoretical DDRL objective defined in Eq. (8).

Expanding the KL divergence term:

$$D_{\text{KL}}(\tilde{p}_{\text{ref}}||p_\theta) = \int \tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)\log\frac{\tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)}{p_\theta(\mathbf{x}_{0:T}|c)}d\mathbf{x}_{0:T} \tag{11}$$

$$= \mathbb{E}_{\tilde{p}_{\text{ref}}}[\log\tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)] - \int \tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)\log p_\theta(\mathbf{x}_{0:T}|c)d\mathbf{x}_{0:T}. \tag{12}$$

Maximizing $\mathcal{J}_{\text{DDRL}}(p_\theta)$ is equivalent to maximizing the simplified functional $\hat{\mathcal{J}}_{\text{DDRL}}(p_\theta)$

$$\hat{\mathcal{J}}_{\text{DDRL}}(p) = \int p_\theta(\mathbf{x}_{0:T}|c)\lambda\left(\frac{r(\mathbf{x}_0,c)-Z}{\beta}\right)d\mathbf{x}_{0:T} + \int \tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)\log p_\theta(\mathbf{x}_{0:T}|c)d\mathbf{x}_{0:T}. \tag{13}$$

The first term is linear in p_θ, and the second term involving $\log p_\theta$ is strictly concave. Thus, the functional $\hat{\mathcal{J}}_{\text{DDRL}}(p)$ is strictly concave with respect to the trajectory distribution $p_\theta(\mathbf{x}_{0:T}|c)$. This ensures that the stationary point derived below is the unique global maximizer.

We formulate the Lagrangian with the normalization constraint:

$$\mathcal{L}(p_\theta,\mu) = \hat{\mathcal{J}}_{\text{DDRL}}(p_\theta) - \mu\left(\int p_\theta(\mathbf{x}_{0:T}|c)d\mathbf{x}_{0:T} - 1\right). \tag{14}$$

Taking the functional derivative w.r.t. $p_\theta(\mathbf{x}_{0:T}|c)$ and setting it to zero:

$$\lambda\left(\frac{r(\mathbf{x}_0,c)-Z}{\beta}\right) + \frac{\tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)}{p_\theta(\mathbf{x}_{0:T}|c)} - \mu = 0. \tag{15}$$

Solving for $p_\theta(\mathbf{x}_{0:T}|c)$:

$$p_\theta(\mathbf{x}_{0:T}|c) = \frac{\tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)}{\mu - \lambda\left(\frac{r(\mathbf{x}_0,c)-Z}{\beta}\right)}. \tag{16}$$

Substituting $\lambda(u) = -\exp(-u)$ and setting $\mu = 0$ (which satisfies the constraint):

$$p^*_\theta(\mathbf{x}_{0:T}|c) = \tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)\exp\left(\frac{r(\mathbf{x}_0,c) - Z}{\beta}\right). \tag{17}$$

To find the optimal sample distribution $p^*_\theta(\mathbf{x}_0|c)$, we marginalize out the intermediate steps $\mathbf{x}_{1:T}$. Recall that $\tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)$ is defined by the forward process starting from the data distribution $\tilde{p}_{\text{data}}$. Integrating over the transition probabilities $\mathbf{x}_{1:T}$ yields the desired result:

$$p^*_\theta(\mathbf{x}_0|c) = \tilde{p}_{\text{data}}(\mathbf{x}_0|c)\exp\left(\frac{r(\mathbf{x}_0,c) - Z}{\beta}\right) \propto \tilde{p}_{\text{data}}(\mathbf{x}_0|c)\exp\left(\frac{r(\mathbf{x}_0,c)}{\beta}\right). \tag{18}$$

Part 2: Equivalence of loss. We now prove that minimizing the regularization term $D_{\text{KL}}(\tilde{p}_{\text{ref}}||p_\theta)$ in Eq. (8) is equivalent to minimizing the diffusion loss $\mathcal{L}(\theta;\tilde{p}_{\text{data}})$ in Eq. (9).

The forward KL divergence is:

$$D_{\text{KL}}(\tilde{p}_{\text{ref}}(\cdot|c)||p_\theta(\cdot|c)) = \mathbb{E}_{\tilde{p}_{\text{ref}}}\left[\log\frac{\tilde{p}_{\text{ref}}(\mathbf{x}_{0:T}|c)}{p_\theta(\mathbf{x}_{0:T}|c)}\right] \tag{19}$$

Start from

$$D_{\text{KL}}\big(\tilde{p}_{\text{ref}}\|p_\theta\big) = \mathbb{E}_{\tilde{p}_{\text{ref}}}\Big[\log\tilde{p}_{\text{data}}(\mathbf{x}_0) + \sum_{t=1}^{T}\log q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \tag{20}$$

$$- \log p(\mathbf{x}_T) - \sum_{t=1}^{T}\log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\Big]. \tag{21}$$

By the telescoping identity, we have

$$\sum_{t=1}^{T}\log q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \log q(\mathbf{x}_T \mid \mathbf{x}_0) + \sum_{t=2}^{T}\log q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0), \tag{22}$$

Hence we have

$$D_{\text{KL}}\big(\tilde{p}_{\text{ref}}\|p_\theta\big) = C_0 + \mathbb{E}\Big[\underbrace{\log q(\mathbf{x}_T \mid \mathbf{x}_0) - \log p(\mathbf{x}_T)}_{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))} - \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) \tag{23}$$

$$+ \sum_{t=2}^{T}\big(\log q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) - \log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\big)\Big], \tag{24}$$

with $C_0 = \mathbb{E}[\log\tilde{p}_{\text{data}}(\mathbf{x}_0)]$ independent of θ. Under the Gaussian assumptions and fixed $\tilde{\beta}_t$, each $t \geq 2$ term is a KL between Gaussians with the same covariance, hence a quadratic in the difference of means; writing both means via ε gives a weighted ε-MSE contribution with weight w_t. The $t = 1$ term equals $-\mathbb{E}\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)$, which is another quadratic that yields the weight w_1. Collecting all terms produces Eq. (9) with a constant C that does not depend on θ. The optimizer equivalence follows since subtracting C does not change $\arg\max_\theta$.

C Experiment Details

In this section, we provide a detailed description of our experimental setup.

Table 5: Hyperparameters used for video post-training.

Model size	2B	14B
Batch size (Number of unique condition c)	16	16
Rollout size N	8	8
Number of GPUs per data instance	2	8
Total GPUs used	256	1024
Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.99$)	
Learning rate	1e-5	3e-6

C.1 VIDEO POST-TRAINING

Model. Except for the SFT-RL integration experiments that we will cover later, our video experiments are initialized from the NVIDIA Cosmos foundation model that is first pre-trained on a large-scale pre-training dataset, fine-tuned separately on six different categories of high-quality datasets, and eventually merged into a single checkpoint ready for RL training. Notice that this model is different from the Cosmos2.5 models released in Ali et al. (2025), which is the checkpoint after they post-train the merged checkpoint with DDRL, i.e., the same process we conduct in Tab. 2. Both 2B and 14B models are dense transformer-based diffusion models, performed under the latent tokenized space that is encoded and decoded using the Wan-2.1 variational autoencoder (Wan et al., 2025). Both models are trained and evaluated at high resolution, where the output videos have resolutions within the ranges (1280, 720) and (720, 1280). All videos have 93 frames, which corresponds to 24 latent frames with a FPS of 16. For the text encoder, we leverage Cosmos-Reason1 (Azzolini et al., 2025) and encode the input prompt c into embedding vectors. This process follows exactly the pre-training stage, with more details provided in Ali et al. (2025).

Data. All algorithms sample prompts (and videos for DDRL) from the high-quality dataset that is used to SFT the model. This dataset includes videos of different categories, ranging from human dynamics, physics, robotics, autonomous driving, and smart spaces. The training dataset is *i.i.d.* to the evaluation PAI-Bench dataset. We always post-train the model for within a thousand iterations, consuming no more than 8,000 unique data points, which is only a small fraction of the whole dataset. Throughout the paper, we use the same 1:1 mixture of Text to Video (T2V) and Image to Video (I2V) data for rollout. This dataset is also used in the SFT-RL integration study to fine-tune the pre-trained model.

Training recipe. We train with rollout size $N = 8$ and batch size 16. We distribute our model using FSDP2. To control the per-GPU memory usage, similar to the Cosmos pre-training strategy, we employ context parallelism (CP), where one data copy will be distributed to two GPUs for the 2B training and eight GPUs for the 14B training. All parameters used for RL are presented in Tab. 5. During each iteration, we first synchronize data across GPU workers within the same rollout group, and then perform rollout in parallel. The output latents are sent to the reward server for reward calculation, which will be clarified in Sec. C.2. Each worker immediately starts computing the diffusion loss when rewards are computed asynchronously, thereby avoiding reward calculation from blocking the training process. Rewards are fetched until we compute the final loss for gradient calculation. We host all reward models in a remote server with 512 GPUs to calculate rewards asynchronously.

Evaluation details. As we mentioned above, we adopt PAI-Bench (Zhou et al., 2025) as our test dataset, with 1044 high-quality test samples (and initial conditioning frames) ranging across five categories (human, industry, robotics, driving, others). For each checkpoint after post-training with each method, we generate videos using 35 denoising steps and a guidance strength of 7 (for 2B) and 3 (for 14B), separately for the text-to-video and image-to-video settings. Once we have all videos generated, we evaluate their reward score using the reward server same as in the training stage.

For human voting comparison between DDRL and each of the baseline methods (and w/o RL setting), we randomly sample a condition c from the test dataset, and then present the videos generated under condition c side-by-side (random order). The voter is asked to choose which video is preferred (or tie), and voters do not know their voting results until they finish voting. In total, 15 people

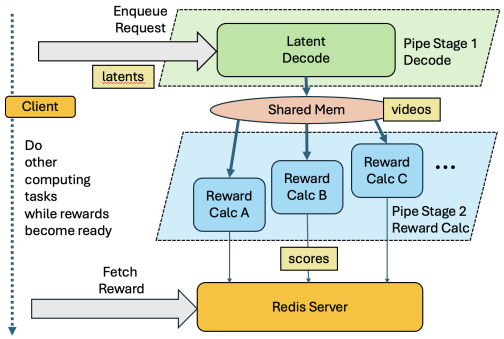

Figure 5: An illustration of the reward server infrastructure.

participated in the voting, where half of the voters are authors of this paper and the other half did not know this paper at all.

SFT-RL integration details. Our study of integrating SFT and RL in Sec. 4.3 shares the identical setting of our main experiments except for the initialization. Specifically, the first setting is initialized with the pre-trained checkpoint (no fine-tuning or model merging). For the second setting, we simplify the Cosmos2.5 training and only conduct a single fine-tuning on the post-training dataset (so results are different from those in Tab. 2). We fine-tuned the pre-trained model for 20,000 iterations using 512 GPUs. We optimize with AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$) with a learning rate of 3e-5.

C.2 Video reward server

To ensure scalability, flexibility, and computational efficiency, we offload reward computation to a dedicated external service, decoupling it from the core generative pipeline. This service supports diverse video quality and alignment metrics—including VBench, VideoAlign, and Cosmos-Reason1-7B-Reward—enabling modular, extensible reward modeling. To minimize network overhead, we transmit only the latent representation of generated videos, decoding them into pixel-space frames on demand within the service.

The reward service is architected as a high-throughput, asynchronous, overlapping pipeline comprising two logically separated stages: Decoding and Reward Calculation. The Decoding stage reconstructs video frames from incoming latents, while the Reward Calculation stage concurrently executes multiple reward models in parallel, each computing distinct metrics on the decoded frames. These stages run as independent processes, enabling heterogeneous software environments (e.g., varying PyTorch/CUDA versions or Python dependencies) to coexist without interference. All processes—including one decoder and multiple reward evaluators—run in parallel to maximize throughput. Such a design also ensures high scalability and flexibility. Inter-stage data transfer leverages CUDA Inter-Process Communication (IPC) for zero-copy sharing of decoded frames between workers, eliminating costly host-device memory copies. When resources are disaggregated, NCCL with high bandwidth is employed for efficient inter-stage communication.

The pipeline processes multiple videos concurrently in a producer-consumer fashion, overlapping decoding and inference to fully utilize GPU and CPU resources. Each reward request is assigned a unique UUID upon enqueue and returns immediately, enabling asynchronous execution. Computed rewards are persisted in a Redis-backed key-value store, indexed by UUID, and retrieved on demand. During the interval between enqueue and result fetch, the calling system may proceed with other tasks to maximize utilization. The system natively supports batched inference: multiple videos can be submitted in a single request, each evaluated independently across all configured reward models. This architecture enables efficient resource utilization, seamless scalability across multiple GPUs and nodes, and robust fault isolation—critical for large-scale video generation evaluation at scale.



Figure 6: Examples of reference images used for diffusion loss regularization in the absence of real data.

Table 6: Breakdown of the reward scores. We compare the base model against DanceGRPO, FlowGRPO, and DDRL across Text-to-Video (T2V) and Image-to-Video (I2V) settings.

			VideoAlign (3 scores)					VBench (5 scores)						
Model	**Setting**	**Method**	**text alignment**	**motion quality**	**visual quality**	**Mean**	**STD**	**subject consistency**	**background consistency**	**motion smoothness**	**aesthetic quality**	**imaging quality**	**Mean**	**STD**
Cosmos2.5-2B	T2V	w/o RL	1.69	-0.45	-0.01	0.408	0.520	0.94	0.95	0.99	0.54	0.73	0.830	0.027
		DanceGRPO	1.50	0.16	0.49	0.715	0.683	0.97	0.97	0.99	0.57	0.75	0.849	0.022
		FlowGRPO	1.67	-0.43	-0.02	0.408	0.556	0.95	0.95	0.99	0.54	0.72	0.830	0.028
		DDRL	1.76	-0.19	0.24	0.604	0.594	0.96	0.96	0.99	0.56	0.73	0.842	0.024
	I2V	w/o RL	1.57	-0.82	-0.52	0.079	0.434	0.92	0.94	0.99	0.52	0.69	4.050	0.160
		DanceGRPO	1.46	-0.45	-0.24	0.254	0.534	0.94	0.94	0.99	0.53	0.71	0.822	0.028
		FlowGRPO	1.57	-0.81	-0.55	0.069	0.436	0.92	0.94	0.99	0.51	0.67	0.807	0.033
		DDRL	1.61	-0.66	-0.42	0.177	0.465	0.94	0.95	0.99	0.52	0.70	0.819	0.030
Cosmos2.5-14B	T2V	w/o RL	1.58	-0.46	-0.04	0.359	0.506	0.94	0.95	0.99	0.52	0.70	0.820	0.034
		DanceGRPO	1.47	-0.19	0.20	0.494	0.587	0.95	0.96	0.99	0.54	0.73	0.834	0.028
		FlowGRPO	1.72	-0.35	0.06	0.476	0.554	0.96	0.96	0.99	0.54	0.70	0.829	0.031
		DDRL	1.72	-0.22	0.16	0.555	0.546	0.95	0.96	0.99	0.55	0.72	0.835	0.030
	I2V	w/o RL	1.46	-0.79	-0.50	0.058	0.429	0.93	0.95	0.99	0.51	0.68	0.813	0.032
		DanceGRPO	1.38	-0.54	-0.36	0.160	0.506	0.93	0.95	0.99	0.52	0.70	0.818	0.029
		FlowGRPO	1.55	-0.69	-0.47	0.128	0.467	0.94	0.95	0.99	0.51	0.65	0.808	0.035
		DDRL	1.51	-0.67	-0.44	0.134	0.453	0.94	0.95	0.99	0.51	0.69	0.819	0.031

C.3 Image post-training

Model, reward and data. We apply DDRL to text-to-image (T2I) tasks based on 512×512 resolution SD3.5-Medium (Esser et al., 2024), with settings aligned with FlowGRPO (Liu et al., 2025a). We focus on improving the OCR reward for visual text rendering, which is the most vulnerable to reward hacking among FlowGRPO's settings. We adopt the same text accuracy calculation strategy as FlowGRPO based on edit distance. We use synthetic data sampled from the base model with 40 ODE steps and its default CFG scale of 4.5 for diffusion loss regularization. The synthetic data are visualized in Figure 6. Despite they are not accurate in text rendering, they provide abundant information of background details to regularize the RL process.

Training and evaluation configurations. Our setup largely follows FlowGRPO, adopting the same number of groups per epoch (48), group size (24), LoRA configuration ($\alpha = 64, r = 32$), and its training and evaluation prompts. We decrease the learning rate from $3e-4$ to $1e-4$ as we find it more stable for DDRL. The relative weight of the diffusion loss is set to 0.001. For synthetic image data, forward noising and diffusion loss computation are performed exactly on the corresponding sampling timesteps. We employ a 10-step SDE sampler for FlowGRPO training and a 40-step ODE sampler for evaluation. The DanceGRPO baseline is implemented by modifying FlowGRPO to remove the reverse KL regularization and use the same initial noise for data collection of the same prompt. The models are trained from around 300 iterations, and we choose DDRL and baseline checkpoints with similar final OCR accuracy for evaluation.

D Results Breakdown

We present a breakdown of Tab. 2 in Tab. 6. This table is also a comprehensive result of Fig. 3. Notice that for each algorithm, the table contain results of four post-trained models (two model size and two rewards). We mix T2V and I2V data during training, and the resulting checkpoint is used both for T2V and I2V evaluation.