TEMPFLOW-GRPO: WHEN TIMING MATTERS FOR GRPO IN FLOW MODELS

Xiaoxuan He[1,2]*, **Siming Fu**[1]*, **Yuke Zhao**[1]*, **Wanli Li**[1], **Jian Yang**[2],
Dacheng Yin[2]†, **Fengyun Rao**[2], **Bo Zhang**[1]‡

[1] ZheJiang University,
[2] WeChat Vision, Tencent Inc

ABSTRACT

Recent flow matching models for text-to-image generation have achieved remarkable quality, yet their integration with reinforcement learning for human preference alignment remains suboptimal, hindering fine-grained reward-based optimization. We observe that the key impediment to effective GRPO training of flow models is the temporal uniformity assumption in existing approaches: sparse terminal rewards with uniform credit assignment fail to capture the varying criticality of decisions across generation timesteps, resulting in inefficient exploration and suboptimal convergence. To remedy this shortcoming, we introduce **TempFlow-GRPO** (Temporal Flow GRPO), a principled GRPO framework that captures and exploits the temporal structure inherent in flow-based generation. TempFlow-GRPO introduces two key innovations: (i) a trajectory branching mechanism that provides process rewards by concentrating stochasticity at designated branching points, enabling precise credit assignment without requiring specialized intermediate reward models; and (ii) a noise-aware weighting scheme that modulates policy optimization according to the intrinsic exploration potential of each timestep, prioritizing learning during high-impact early stages while ensuring stable refinement in later phases. These innovations endow the model with temporally-aware optimization that respects the underlying generative dynamics, leading to state-of-the-art performance in human preference alignment and standard text-to-image benchmarks.

1 INTRODUCTION

While text-to-image diffusion models have achieved unprecedented visual quality and semantic control Esser et al. (2024); Xie et al. (2024a); Labs et al. (2025), aligning their outputs with human preference remains a formidable challenge. Reinforcement learning has emerged as a promising solution, giving rise to the field of Diffusion RL Wallace et al. (2024); Black et al. (2023); Fan et al. (2023). However, the performance of these methods remains suboptimal, hindered by two fundamental limitations that have been largely overlooked: ***ignoring the temporal dynamics of generation*** and ***lacking intermediate feedback signals***. These approaches apply uniform optimization across all timesteps and provide rewards only at completion, missing the varying importance of decisions throughout the generation process.

The majority of existing approaches Gu et al. (2024); Hong et al. (2024), including recent works like Flow-GRPO Liu et al. (2025) and DanceGRPO Xue et al. (2025), treat the multi-step generation process as a "black box" with temporally agnostic optimization. They apply uniform updates across all timesteps despite the fact that each timestep operates under different noise conditions and contributes differently to final image quality. Specifically, we plot the Fig. 1 (left) with applying SDE at only one timestep in the entire ODE trajectory, which ensures that any deviation in the final reward can be attributed to the stochastic exploration introduced at that specific step. As shown in

* Equal Contribution.
† Project Leader.
‡ Corresponding authors.

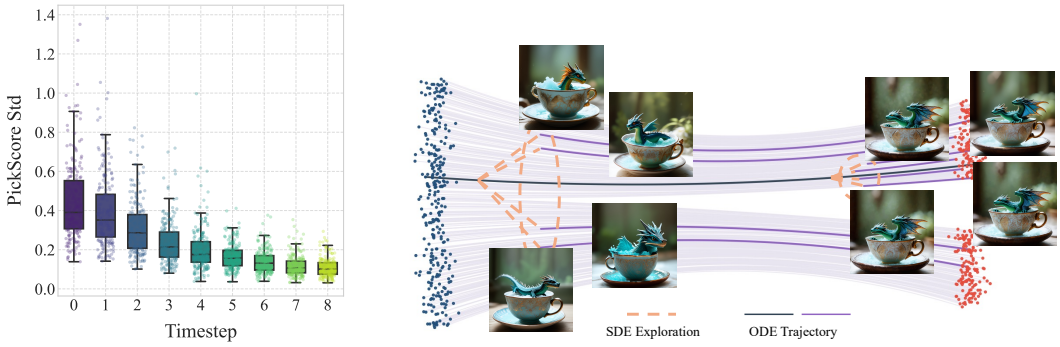

Figure 1: (Left) Reward Variance Analysis: We plot the standard deviation of PickScore at each denoising step for 200 prompts, per prompt group size is 24. The results, obtained via applying SDE at only one step, reveal that reward variance is highest in the initial steps, indicating that early-stage interventions are most impactful for exploration. (Right) Method Illustration: By branching a stochastic (SDE) exploration from a specific, known state on a deterministic (ODE) trajectory, we create a controlled experiment. The resulting difference in the final reward can be unambiguously attributed to the exploration action taken at that precise branching point.

Fig. 1 (left), the std of the reward varies dramatically between timesteps, reaching a peak during early structural decisions (steps 0-2) and approaching zero during final refinements (steps 6-8). Yet Flow-GRPO maintains uniform treatment throughout, squandering high-impact exploration opportunities while focusing on later steps. Alternative approaches like SPO Liang et al. (2025) attempt to address temporal dynamics through process reward models, but training such models on semantically ambiguous intermediate states is notoriously difficult. This raises a fundamental question: ***how can we effectively achieve precise credit assignment for intermediate actions while adapting optimization intensity to each timestep's exploration capacity?***

We address these limitations with **TempFlow-GRPO**, a temporally-aware RL framework built on two key insights. **First**, we define the visualization method in the left of Fig. 1 as trajectory branching, which enables precise credit assignment by strategically introducing stochasticity at individual timesteps while maintaining deterministic evolution elsewhere (Fig. 1, Right). This provides *provable* guarantees: (1) reward variance localizes to the branching point, (2) improvements are directly attributable to specific exploration outcomes, and (3) existing reward models require no modification. **Second**, noise-aware policy weighting modulates optimization intensity based on each timestep's intrinsic noise level. Early high-noise stages receive larger weight updates to encourage structural exploration, while late low-noise stages receive gentler updates to preserve learned features. Together, these mechanisms create a framework that is conceptually simple, computationally efficient, and seamlessly integrates into existing flow matching architectures—all while respecting the temporal dynamics that uniform approaches ignore.

Our main contributions are threefold:

- We pinpoint temporal uniformity—the equal treatment of all timesteps—as the primary limitation of flow-based GRPO. Our proposed **TempFlow-GRPO** overcomes this by introducing two key innovations: precise credit assignment to intermediate actions and noise-aware adaptation of optimization intensity.
- We introduce *trajectory branching* and *noise-aware weighting* to learn temporally-structured policies that respect generative dynamics.
- We demonstrate state-of-the-art performance on standard text-to-image benchmarks, achieving superior sample quality, human preference alignment, and compositional image generation compared to existing flow-based RL methods.

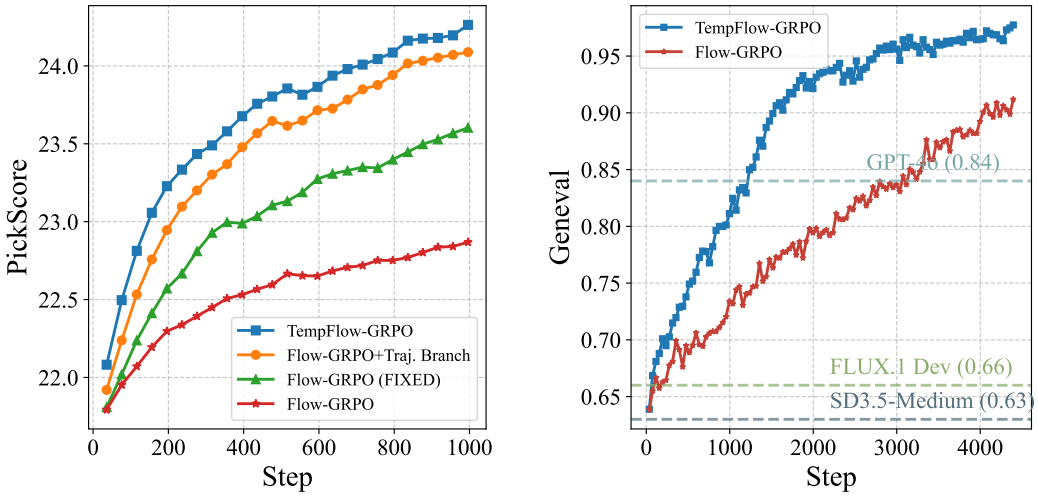

Figure 2: (Left) Performance comparison on the PickScore benchmark. To ensure a fair and robust evaluation, we first present Flow-GRPO (Fixed), **an improved baseline where we replace the original global std stabilization with a more appropriate group-wise std**. On top of this stronger baseline, our core innovation, trajectory branching (+ Traj. Branch), delivers a significant performance leap. Our full model, TempFlow-GRPO, integrates all components to achieve the highest performance. (Right) TempFlow-GRPO's dominance is further confirmed on the Geneval benchmark, where it substantially surpasses not only the Flow-GRPO baseline but also leading state-of-the-art models, including GPT-4o, SD3.5-M, and FLUX.

2 Related Work

Alignment for Diffusion Models. Alignment for diffusion models has become a rapidly emerging topic in generative modeling research. D3PO Yang et al. (2024) introduce the Direct Preference for Denoising Diffusion Policy Optimization method to directly fine-tune diffusion models. Diffusion-DPO Wallace et al. (2024) is adapted from the Direct Preference Optimization (DPO Rafailov et al. (2023)), a simpler alternative to RLHF which directly optimizes a policy that best satisfies human preferences under a classification objective. DyMO Xie & Gong (2025) propose a plug-and-play training-free alignment method for aligning the generated images and human preferences during inference. Recently, Flow-GRPO Liu et al. (2025), the first method to integrate online reinforcement learning (RL) into flow matching models. However, Flow-GRPO applies uniform optimization pressure across all timesteps and suffers from sparse terminal reward problems, failing to account for the time-varying exploration potential inherent in the stochastic diffusion process. Our TempFlow-GRPO addresses these limitations through trajectory branching for precise credit assignment and noise-aware policy weighting that aligns optimization pressure with the natural exploration capacity at each timestep.

Process Reward. Recent studies have demonstrated that shaping the reward process, rather than solely relying on sparse terminal rewards, can significantly accelerate learning and improve policy performance. Zhang et al. (2025) introduces a more comprehensive evaluation framework, which combines response-level and step-level metrics. ThinkPRM Khalifa et al. (2025) builds data-efficient PRMs as verbalized step-wise reward models that verify every step in the solution by generating a verification chain-of-thought (CoT). In diffusion models, SPO Liang et al. (2024) trains a separate step-aware preference model that can be applied to both noisy and clean images. Despite this, PRMs require step-level supervision, making them expensive to train. There are several methods using outcome reward to replace process reward. PRIME Cui et al. (2025) enables online PRM updates using only policy rollouts and outcome labels through implict process rewards. Math-Shepherd Wang et al. (2023) assigns a reward score to each step of math problem solutions. Due to the significant challenges in scoring noisy images, there is an urgent need for an algorithm that enables process reward on flow models. Our TempFlow-GRPO elegantly circumvents the need for specialized process reward models by directly attributing outcome-based reward signals to interme-

diate exploratory actions, enabling precise credit assignment without the computational overhead of training step-level evaluators for semantically ambiguous noisy states.

3 PRELIMINARY: FLOW-GRPO

Flow-GRPO enhances flow models using online RL. First, we revisit the core idea of GRPO. Then, we show how Flow-GRPO converts the deterministic ODE sampler into a SDE sampler with same marginal distribution, which introduces the stochasticity needed for applying GRPO.

GRPO. RL aims to learn a policy that maximizes the expected cumulative reward. GRPO optimizes the policy model by maximizing the following objective:

$$\mathcal{J}_{\text{Flow-GRPO}}(\theta) = \mathbb{E}_{\boldsymbol{c}\sim\mathcal{C},\{\boldsymbol{x}^i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{c})} f(r, \hat{A}, \theta, \epsilon, \beta) \tag{1}$$

where

$$\begin{aligned} f(r, \hat{A}, \theta, \epsilon, \beta) &= \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=0}^{T-1}(min(r_t^i(\theta)\hat{A}_t^i, \text{clip}(r_t^i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^i) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}})) \\ r_t^i(\theta) &= \frac{p_\theta(\boldsymbol{x}_{t-1}^i|\boldsymbol{x}_t^i, \boldsymbol{c})}{p_{\theta_{\text{old}}}(\boldsymbol{x}_{t-1}^i|\boldsymbol{x}_t^i, \boldsymbol{c})}, T \text{ is the timestep.} \end{aligned} \tag{2}$$

Given a prompt $\boldsymbol{c}$, the flow model p_θ samples a group of G individual images $\{\boldsymbol{x}_0^i\}_{i=1}^G$ and the corresponding reverse-time trajectories $\{(\boldsymbol{x}_T^i, \boldsymbol{x}_{T-1}^i, ..., \boldsymbol{x}_0^i\}_{i=1}^G$. Then, the advantage of the i-th image is calculated by normalizing the group-level rewards as follows:

$$\hat{A}_t^i = \frac{R(\boldsymbol{x}_0^i, \boldsymbol{c}) - \text{mean}(\{R(\boldsymbol{x}_0^i, \boldsymbol{c})\}_{i=1}^G)}{\text{std}(\{R(\boldsymbol{x}_0^i, \boldsymbol{c})\}_{i=1}^G)} \tag{3}$$

Convert ODE to SDE. GRPO relies on stochastic sampling to generate diverse trajectories for advantage estimation and exploration. However, flow matching models use a deterministic ODE for the forward process:

$$d\boldsymbol{x}_t = \boldsymbol{v}_t dt \tag{4}$$

Flow-GRPO converts the deterministic ODE into an equivalent SDE that matches the original model's marginal probability density function at all timesteps. The final update rule is as follows:

$$\boldsymbol{x}_{t+\Delta t} = \boldsymbol{x}_t + [\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) + \frac{\sigma_t^2}{2t}(\boldsymbol{x}_t + (1-t)\boldsymbol{v}_\theta(\boldsymbol{x}_t, t))]\Delta t + \sigma_t\sqrt{\Delta t}\boldsymbol{\epsilon} \tag{5}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ injects stochasticity and $\sigma_t = a\sqrt{\frac{t}{1-t}}$. And the KL divergence between π_θ and the reference policy π_{ref} is a closed form:

$$D_{KL}(\pi_\theta||\pi_{\text{ref}}) = \frac{||\bar{\boldsymbol{x}}_{t+\Delta t,\theta} - \bar{\boldsymbol{x}}_{t+\Delta t,\text{ref}}||}{2\sigma_t^2\Delta t} = \frac{\Delta t}{2}(\frac{\sigma_t(1-t)}{2t} + \frac{1}{\sigma_t})^2||\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{v}_{\text{ref}}(\boldsymbol{x}_t, t)||^2 \tag{6}$$

4 METHODS

4.1 TEMPORAL FLOW-GRPO

While Flow-GRPO represents a significant advancement in applying online RL to flow matching models, its direct adoption of the standard GRPO framework overlooks the unique, time-dependent

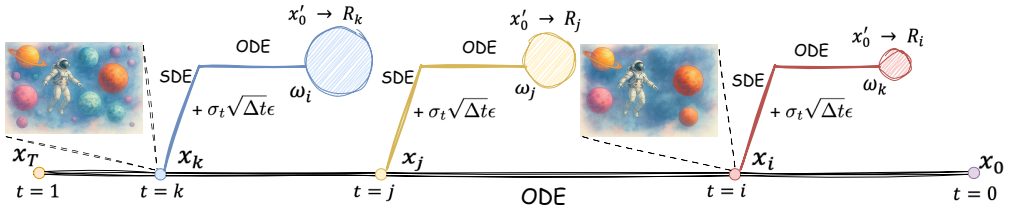

Figure 3: **Overview of TempFlow-GRPO Framework.** Our method performs trajectory branching by switching from ODE to SDE sampling at selected timesteps (t=k, j, i), injecting noise $\sigma_t\sqrt{\Delta t}\epsilon$ to create exploratory branches. Each branch generates a distinct outcome with reward R_i, enabling precise credit assignment. The framework applies noise-aware weighting where $\omega_i > \omega_j > \omega_k$, prioritizing optimization at high-noise early stages (larger circles) over low-noise refinement phases (smaller circles), aligning learning intensity with each timestep's intrinsic exploration capacity. We visualize the model's learning process as an astronaut exploring unknown planets: in early stages , the model explores vast possibility spaces with high uncertainty, while later stages involve focused navigation toward the final destination.

dynamics inherent in the generative process. We identify two fundamental limitations that motivate our approach:

Sparse Terminal Reward Problem: The current approach relies on sparse terminal rewards, assigning the same credit uniformly across all timesteps. This fails to differentiate the critical, high-impact decisions of early generation from the fine-tuning adjustments of later stages.

Uniform Optimization Weighting: The GRPO objective applies uniform optimization among all steps, ignoring that the underlying SDE sampler possesses non-uniform noise. The potential for meaningful exploration is greater in high-noise early steps than in the low-noise refinement phase Xie & Gong (2025).

To address these limitations, as illustrated in Fig. 3, we propose **TempFlow-GRPO**, a unified framework that introduces two synergistic innovations: process rewards via trajectory branching and noise-aware policy weighting.

4.1.1 Trajectory Branching for Process Rewards

Traditional process reward methods require specialized reward models to evaluate noisy intermediate states $\boldsymbol{x}_t$, which is exceptionally difficult due to the semantic ambiguity of partially-denoised representations. We propose an elegant alternative that leverages the deterministic-stochastic sampling methods of flow matching models.

Key Insight: Instead of training complex process reward models, we use existing high-quality outcome-based reward models and directly attribute their scores to intermediate exploratory actions through a novel trajectory branching mechanism.

Consider a generative process parameterized by θ. In flow matching models, the policy gradient can be written as:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T\sim\mathcal{N}(0,\boldsymbol{I}),\boldsymbol{\epsilon}\sim\mathcal{N}(0,\boldsymbol{I})}[\nabla_\theta \log p_\theta(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k)\hat{A}_k] \tag{7}$$

Definition 1 (Trajectory Branching): We define a trajectory branching operation where a trajectory evolves deterministically at a **designated branching timestep** k, where $\boldsymbol{x}_k$ is sampled from initial noise $\boldsymbol{x}_T$ using Eq. 4. At this branching point, stochasticity is introduced via noise variable $\boldsymbol{\epsilon}$ in Eq. 5, yielding $\boldsymbol{x}_{k-1} = \text{SDE}(\boldsymbol{x}_k, \boldsymbol{\epsilon})$. The remainder of the trajectory $\boldsymbol{x}_{k-2}, \boldsymbol{x}_{k-3}, \ldots, \boldsymbol{x}_0$ is generated deterministically as $\boldsymbol{x}_0 = \text{ODE}(\boldsymbol{x}_{k-1})$.

Theorem 1 (Credit Localization): Since all stochasticity and model controllability are concentrated at the branching point, the total reward variance and all parameter-dependent improvements are entirely attributable to the outcome of noise injection at k. This enables rigorous and efficient credit assignment localized to the branching point.

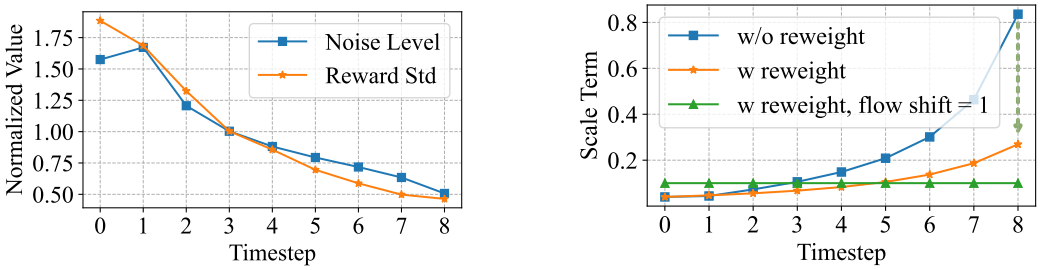

Figure 4: **(Left)** A comparative analysis between the reward standard deviation (Reward Std) and the noise level over the generative steps. The two curves show a strong correlation. **(Right)** Scale terms reveal a mismatch in standard GRPO: Scale terms are inversely proportional to noise level, causing low-noise refinement steps to dominate optimization despite having minimal impact on image content. Early steps that establish global structure receive weak gradients, while late steps that only adjust local details produce strong gradients. Our noise-aware reweighting compensates for this inverse relationship, ensuring that optimization intensity aligns with each timestep's actual capacity to influence the final image.

In practice, we replace the reward for the k-th step from $R(\boldsymbol{x}_0^i, \boldsymbol{c})$ to $R(\text{ODE}(\text{SDE}(\boldsymbol{x}_k^i, \boldsymbol{\epsilon}^i)), \boldsymbol{c})$, where $\boldsymbol{x}_0^i$ is sampled with SDE and $\text{ODE}(\text{SDE}(\boldsymbol{x}_k^i, \boldsymbol{\epsilon}^i)$ is sampled with ODE-SDE-ODE. Trajectory branching allows for the precise attribution of the terminal reward to step k, effectively creating a temporally-aware reward signal.

4.1.2 Noise-Aware Policy Weighting

While trajectory branching provides precise reward signals for individual timesteps, the generative process consists of a sequence of T potential branching points with fundamentally different characteristics. The SDE sampler exhibits time-varying stochasticity: the noise injection magnitude $\sigma_t\sqrt{\Delta t}$ is large during initial generation stages and diminishes to near zero during final refinement stages. This non-uniform noise distribution implies that exploration capacity varies dramatically across timesteps. An exploratory action at an early stage has vastly different impact and risk compared to perturbations on near-perfect images. However, standard GRPO applies uniform optimization pressure, implicitly assuming equal learning importance at all stages.

Reweighting by Reward Std. An intuitive approach to reweighting the policy loss is to utilize the standard deviation of the rewards. As illustrated in the left of Fig. 4, the reward standard deviation exhibits a similar decaying trend as the denoising process unfolds, suggesting it could serve as a meaningful indicator of learning importance. However, this approach introduces significant complexity. It would necessitate maintaining a dynamic weighting hyperparameter for each timestep, likely tracked via an Exponential Moving Average (EMA Morales-Brotons et al. (2024)). This dynamic estimation process can be unstable and adds an undesirable layer of complexity to the training regime. Consequently, we seek a more direct and elegant weighting scheme.

Reweighting by Noise Level. We wonder if directly leverage the level of the exploration space itself as a proxy for a reweighting factor. We visualized the noise level alongside the reward standard deviation (Fig. 4, left) and observed a striking correspondence between the two. This strong correlation suggests that the noise level, serves as an excellent and intrinsic proxy for the exploration capacity and associated risk at each timestep.

We therefore propose to reweight the policy loss directly using the noise level. For each timestep t, we introduce a weighting factor proportional to the noise level. Specifically, we modify the original policy gradient loss function to the following weighted form:

$$\mathcal{J}_{\text{policy}}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=0}^{T-1}\text{Norm}(\sigma_t\sqrt{\Delta t})(min(r_t^i(\theta)\hat{A}_t^i, \text{clip}(r_t^i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^i) \tag{8}$$

The intuition behind this weighting strategy is to align the optimization pressure with the inherent properties of the generative process. In the early stages of generation, noise is large, amplifying the

learning signal during these high-noise, high-impact phases and encouraging the model to perform effective macroscopic exploration. As generation proceeds, noise diminishes, shifts the optimization focus towards fine-grained adjustments and stability, preventing aggressive exploration from corrupting a high-fidelity state with noise or artifacts.

4.2 Policy Gradient-Based Theoretical Justification

To provide a deeper understanding of our approach, we now examine it from the policy gradient perspective. **Notice that in the $\sum_{k=0}^{T-1}$, k is the timestep, and in the equation, k is the value of the timestep.** Simplifying Eq. 5, we obtain $\boldsymbol{x}_{k-1} \sim \mathcal{N}(\mu_\theta(\boldsymbol{x}_k, k), \sigma_k^2 \Delta k \boldsymbol{I})$, where:

$$\mu_\theta(\boldsymbol{x}_k, k) = \boldsymbol{x}_k + \left[\boldsymbol{v}_\theta(\boldsymbol{x}_k, k) + \frac{\sigma_k^2}{2k}\left(\boldsymbol{x}_k + (1-k)\boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\right)\right]\Delta k \tag{9}$$

Starting from the policy gradient formulation in Eq. 7, we have:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}[\nabla_\theta \log p_\theta(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k)\hat{A}_k] \tag{10}$$

Substituting $\boldsymbol{x}_{k-1}$ in the log-probability:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\nabla_\theta \log \exp\left(-\frac{\|\boldsymbol{x}_{k-1} - \mu_\theta(\boldsymbol{x}_k, k)\|^2}{2\sigma_k^2 \Delta k}\right)\hat{A}_k\right] \tag{11}$$

$$= \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\nabla_\theta \left(-\frac{\|\boldsymbol{x}_{k-1} - \mu_\theta(\boldsymbol{x}_k, k)\|^2}{2\sigma_k^2 \Delta k}\right)\hat{A}_k\right] \tag{12}$$

Taking the gradient with respect to θ:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\frac{\boldsymbol{x}_{k-1} - \mu_\theta(\boldsymbol{x}_k, k)}{\sigma_k^2 \Delta k} \cdot \nabla_\theta \mu_\theta(\boldsymbol{x}_k, k)\hat{A}_k\right] \tag{13}$$

Since $\boldsymbol{x}_{k-1} = \mu_\theta(\boldsymbol{x}_k, k) + \sigma_k \sqrt{\Delta k} \cdot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\frac{\boldsymbol{\epsilon}}{\sigma_k \sqrt{\Delta k}} \cdot \nabla_\theta \mu_\theta(\boldsymbol{x}_k, k)\hat{A}_k\right] \tag{14}$$

Expanding $\nabla_\theta \mu_\theta(\boldsymbol{x}_k, k)$:

$$\nabla_\theta \mu_\theta(\boldsymbol{x}_k, k) = \nabla_\theta \left[\boldsymbol{x}_k + \left(\boldsymbol{v}_\theta(\boldsymbol{x}_k, k) + \frac{\sigma_k^2}{2k}\left(\boldsymbol{x}_k + (1-k)\boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\right)\right)\Delta k\right] \tag{15}$$

$$= \nabla_\theta \left[\left(\boldsymbol{v}_\theta(\boldsymbol{x}_k, k) + \frac{\sigma_k^2(1-k)}{2k}\boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\right)\Delta k\right] \tag{16}$$

$$= \left(1 + \frac{\sigma_k^2(1-k)}{2k}\right)\Delta k \cdot \nabla_\theta \boldsymbol{v}_\theta(\boldsymbol{x}_k, k) \tag{17}$$

Substituting back:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\frac{\boldsymbol{\epsilon}}{\sigma_k \sqrt{\Delta k}} \cdot \left(1 + \frac{\sigma_k^2(1-k)}{2k}\right)\Delta k \cdot \nabla_\theta \boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\hat{A}_k\right] \tag{18}$$

$$= \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T \sim \mathcal{N}(0,\boldsymbol{I}), \boldsymbol{\epsilon} \sim \mathcal{N}(0,\boldsymbol{I})}\left[\left(\frac{\sqrt{\Delta k}}{\sigma_k} + \frac{\sigma_k \sqrt{\Delta k}(1-k)}{2k}\right) \cdot \boldsymbol{\epsilon} \cdot \nabla_\theta \boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\hat{A}_k\right] \tag{19}$$

With $\sigma_k = a\sqrt{\frac{k}{1-k}}$, we get:

$$\frac{\sqrt{\Delta k}}{\sigma_k} = \frac{\sqrt{\Delta k}}{a\sqrt{\frac{k}{1-k}}} = \frac{1}{a}\sqrt{\frac{\Delta k(1-k)}{k}} \tag{20}$$

$$\frac{\sigma_k\sqrt{\Delta k}(1-k)}{2k} = \frac{a\sqrt{\frac{k}{1-k}}\sqrt{\Delta k}(1-k)}{2k} = \frac{a}{2}\sqrt{\frac{\Delta k(1-k)}{k}} \tag{21}$$

Therefore:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T\sim\mathcal{N}(0,\boldsymbol{I}),\boldsymbol{\epsilon}\sim\mathcal{N}(0,\boldsymbol{I})} \left[\left(\frac{1}{a} + \frac{a}{2}\right) \underbrace{\sqrt{\frac{\Delta k(1-k)}{k}}}_{\text{Scale Term}} \cdot\boldsymbol{\epsilon}\cdot \nabla_\theta \boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\hat{A}_k \right] \tag{22}$$

This reveals that the natural gradient coefficient is proportional to $\sqrt{\frac{1-k}{k}}\sqrt{\Delta k}$, which captures the intrinsic exploration potential at timestep k. After reweighting, we have the following derivation:

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{k=0}^{T-1} \mathbb{E}_{\boldsymbol{x}_T\sim\mathcal{N}(0,\boldsymbol{I}),\boldsymbol{\epsilon}\sim\mathcal{N}(0,\boldsymbol{I})} \left[\left(\frac{1}{a} + \frac{a}{2}\right) \underbrace{\Delta k}_{\text{Scale Term}} \cdot\boldsymbol{\epsilon}\cdot \nabla_\theta \boldsymbol{v}_\theta(\boldsymbol{x}_k, k)\hat{A}_k \right] \tag{23}$$

Consider $\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\boldsymbol{I})}[\boldsymbol{\epsilon}\hat{A}_k]$, suppose the final reward is a function of the small noise vector $\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}$ applied at a certain step. When $\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}$ is small (and drawn from a zero-mean Gaussian), we can approximate the reward using a first-order Taylor expansion:

$$R_k(\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}) \approx R_k(0) + \sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}^T \nabla_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}} R_k|_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}=0} \tag{24}$$

Since $\hat{A}_k$ is normalized version of R_k, the mean and std are as follows (let $\nabla_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}} R_k|_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}=0} = g_k$):

$$\begin{aligned}
\text{mean} &= \mathbb{E}_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}}[R_k(0) + \sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}^T g_k] = R_k(0) \\
\text{std} &= \sqrt{\mathbb{E}_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}}[(R_k(\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}) - \text{mean})^2]} \\
&= \sqrt{\mathbb{E}_{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}}[(\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}^T g_k)^2]} \\
&= \sigma_k\sqrt{\Delta k}||g_k||
\end{aligned} \tag{25}$$

Therefore:

$$\begin{aligned}
\hat{A}_k &= \frac{R_k - \text{mean}}{\text{std}} = \frac{\sigma_k\sqrt{\Delta k}\boldsymbol{\epsilon}^T g_k}{\sigma_k\sqrt{\Delta k}||g_k||} \\
\mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}\hat{A}_k] &= \mathbb{E}_{\boldsymbol{\epsilon}}[\frac{\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T g_k}{||g_k||}] = \frac{g_k}{||g_k||}
\end{aligned} \tag{26}$$

Eq. 26 indicates that the norm of $\mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}\hat{A}_k]$ is invariant among the timesteps. So in Eq. 22 and Eq. 23, the scale terms that modulate the contribution of each timestep's model gradient $\nabla_\theta \boldsymbol{v}_\theta(\boldsymbol{x}_k, k)$ to the overall reward gradient simplify to distinct functions, which we denote as $\sqrt{\frac{\Delta k(1-k)}{k}}$ and Δk. We visualize these scale terms under different flow shift, as depicted in the right of Fig. 4. As our analysis indicates, the setting of "w/o reweighting" exhibits a highly imbalanced contribution from each timestep's model gradient to the final reward gradient. This non-uniformity systematically causes the contribution from the early denoising step where the model performs broad, structural exploration to be significantly smaller than the contribution from the late step, which focuses on fine-grained refinement. By employing our proposed noise-aware policy reweighting, this issue is substantially mitigated, as the scale term simplifies to being directly proportional to the step size, Δk. Furthermore, when the flow shift is set as 1, our method achieves a perfect equilibrium: it ensures that the gradient contribution from every single timestep is precisely equal, thereby completely balancing this effect across the entire generation trajectory.

Table 1: **GenEval Result.** Best scores are in blue, second-best in green. Results for models are from Flow-GRPO. Obj.: Object; Attr.: Attribution.

Model	Step	Overall	Single Obj.	Two Obj.	Counting	Colors	Position	Attr. Binding
Diffusion Models								
LDM Rombach et al. (2022)	-	0.37	0.92	0.29	0.23	0.70	0.02	0.05
SD1.5 Rombach et al. (2022)	-	0.43	0.97	0.38	0.35	0.76	0.04	0.06
SD2.1 Rombach et al. (2022)	-	0.50	0.98	0.51	0.44	0.85	0.07	0.17
SD-XL Podell et al. (2023)	-	0.55	0.98	0.74	0.39	0.85	0.15	0.23
DALLE-2 Ramesh et al. (2022)	-	0.52	0.94	0.66	0.49	0.77	0.10	0.19
DALLE-3 Betker et al. (2023)	-	0.67	0.96	0.87	0.47	0.83	0.43	0.45
Autoregressive Models								
Show-o Xie et al. (2024b)	-	0.53	0.95	0.52	0.49	0.82	0.11	0.28
Emu3-Gen Wang et al. (2024)	-	0.54	0.98	0.71	0.34	0.81	0.17	0.21
JanusFlow Ma et al. (2025)	-	0.63	0.97	0.59	0.45	0.83	0.53	0.42
Janus-Pro-7B Chen et al. (2025)	-	0.80	0.99	0.89	0.59	0.90	0.79	0.66
GPT-4o Hurst et al. (2024)	-	0.84	0.99	0.92	0.85	0.92	0.75	0.61
Flow Matching Models								
FLUX.1 Dev Black et al. (2025)	-	0.66	0.98	0.81	0.74	0.79	0.22	0.45
SD3.5-L Esser et al. (2024)	-	0.71	0.98	0.89	0.73	0.83	0.34	0.47
SANA-1.5 4.8B Xie et al. (2025)	-	0.81	0.99	0.93	0.86	0.84	0.59	0.65
SD3.5-M Esser et al. (2024)	-	0.63	0.98	0.78	0.50	0.81	0.24	0.52
GRPO based Methods								
SD3.5-M+Flow-GRPO Liu et al. (2025)	5600	0.95	1.00	0.99	0.95	0.92	0.99	0.86
SD3.5-M+Flow-GRPO Liu et al. (2025)	4400	0.90	0.99	0.97	0.90	0.88	0.85	0.80
SD3.5-M+TempFlow-GRPO	4400	0.97	1.00	0.99	0.96	0.97	0.98	0.90

5 EXPERIMENT

Following Flow-GRPO, we validate our approach on Compositional Image Generation in Geneval Ghosh et al. (2023) and Human Preference Alignment in PickScore Kirstain et al. (2023). To ensure a fair comparison, we normalized the weights applied to the policy loss to have a mean of 1 at all timesteps. We use num groups as 48 and group size as 24. The beta of kl loss, is set to 0.001 for PickScore and 0.004 in Geneval. Unless otherwise specified, the image size is set to 512, consistent with the configuration used for Flow-GRPO. **It is worth noting that we further improve upon the original Flow-GRPO method by proposing a variant, Flow-GRPO (FIXED), which demonstrates superior performance. In our PickScore experiments, we adopt this improved version to ensure a fair and strong baseline for comparison.**

5.1 MAIN RESULTS

Compositional Image Generation. We evaluate the compositional image generation capability of TempFlow-GRPO on the Geneval benchmark with its corresponding reward model. The experimental results are summarized in Tab. 1. As shown, our approach significantly improves the performance of the base model, increasing the overall score from 0.63 to 0.97. Furthermore, among GRPO-based methods, our method substantially outperforms Flow-GRPO: it achieves a performance of 0.97 within only 4,400 steps, whereas Flow-GRPO reaches only 0.90 under the same conditions. Additionally, as illustrated in Fig. 2, our method requires only about 2,000 steps to achieve a score of 0.95, while Flow-GRPO needs approximately 5,600 steps to reach the same level. Overall, these results demonstrate that TempFlow-GRPO not only accelerates convergence but also achieves superior final performance compared to existing approaches.

Human Preference Alignment. To further validate the generalizability of our approach, we conducted experiments on the PickScore benchmark, using PickScore as the reward model. As shown in the left of Fig. 2, our method, TempFlow-GRPO, achieves the highest performance, surpassing the original Flow-GRPO by approximately 1.3% and outperforming the improved baseline Flow-GRPO (Fixed) by about 0.6 %. Notably, our method requires only 100–200 training steps to match the performance of Flow-GRPO, and just 300–400 steps to reach the level of Flow-GRPO (Fixed). These results on PickScore further demonstrate the general applicability of our method as a unified flow-based RL algorithm across different reward models.

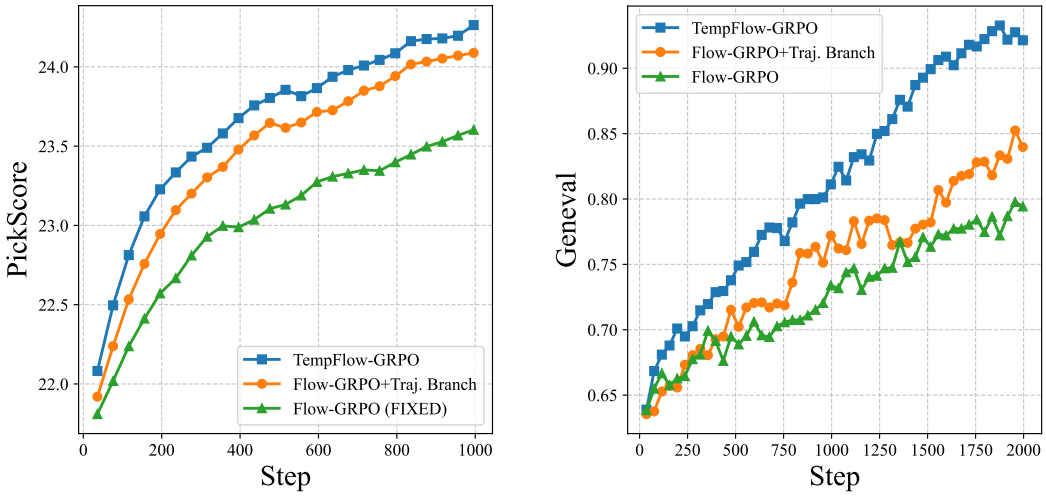

Figure 5: Ablation Studies on trajectory branch and noise-aware policy reweighting. **Left**: PickScore Benchmark. **Right**: Geneval Benchmark.

5.2 Analysis

Ablation of TempFlow-GRPO. We conducted ablation studies to explore the effectiveness of our proposed components: the trajectory branch and noise-aware policy weighting. These ablations were performed on both the Geneval and PickScore benchmarks. As shown in the Fig. 5, on the PickScore benchmark, introducing the trajectory branch further improves the performance of Flow-GRPO (Fixed), and applying noise-aware reweighting on top of this yields the highest pickscore results. On the Geneval benchmark, the benefit of the noise-aware strategy is even more significant: compared to Flow-GRPO, noise-aware policy reweighting boosts performance from 0.85 to 0.94, a 9% improvement while the trajectory branch also brings about a substantial gain of approximately 5%. These ablation results clearly demonstrate the effectiveness of our proposed methods.

Result of 1024 resolution. We further explored the effectiveness of our approach across different image resolutions, using PickScore as the reward model. As illustrated in the Fig. 6, after 450 training steps, TempFlow-GRPO achieves a 0.6% improvement in PickScore. Notably, our method requires only about 180 steps to reach the performance that Flow-GRPO (Fixed) attains after 450 steps. This further demonstrates the efficiency and effectiveness of our proposed method across varying resolutions.

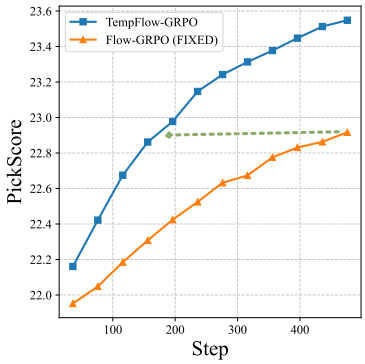

Figure 6: Comparsion on PickScore benchmark in 1024 resolution.

Qualitative Result. We also conducted qualitative analyses on the SD3.5-M, Flow-GRPO (Fixed), and TempFlow-GRPO. As shown in the visualization results in Fig 7, compared to Flow-GRPO (Fixed), TempFlow-GRPO produces images with noticeably finer details and fewer visual artifacts or mistakes. In particular, our approach demonstrates superior capability in preserving complex structures and realistic textures. These qualitative improvements further highlighting the advantages of our method in generating high-quality, visually appealing images.

6 Limitation and Future Work

Although our method achieves significant improvements in both performance and image quality, the current experiments are based on a single reward model. **In future work, we plan to focus**

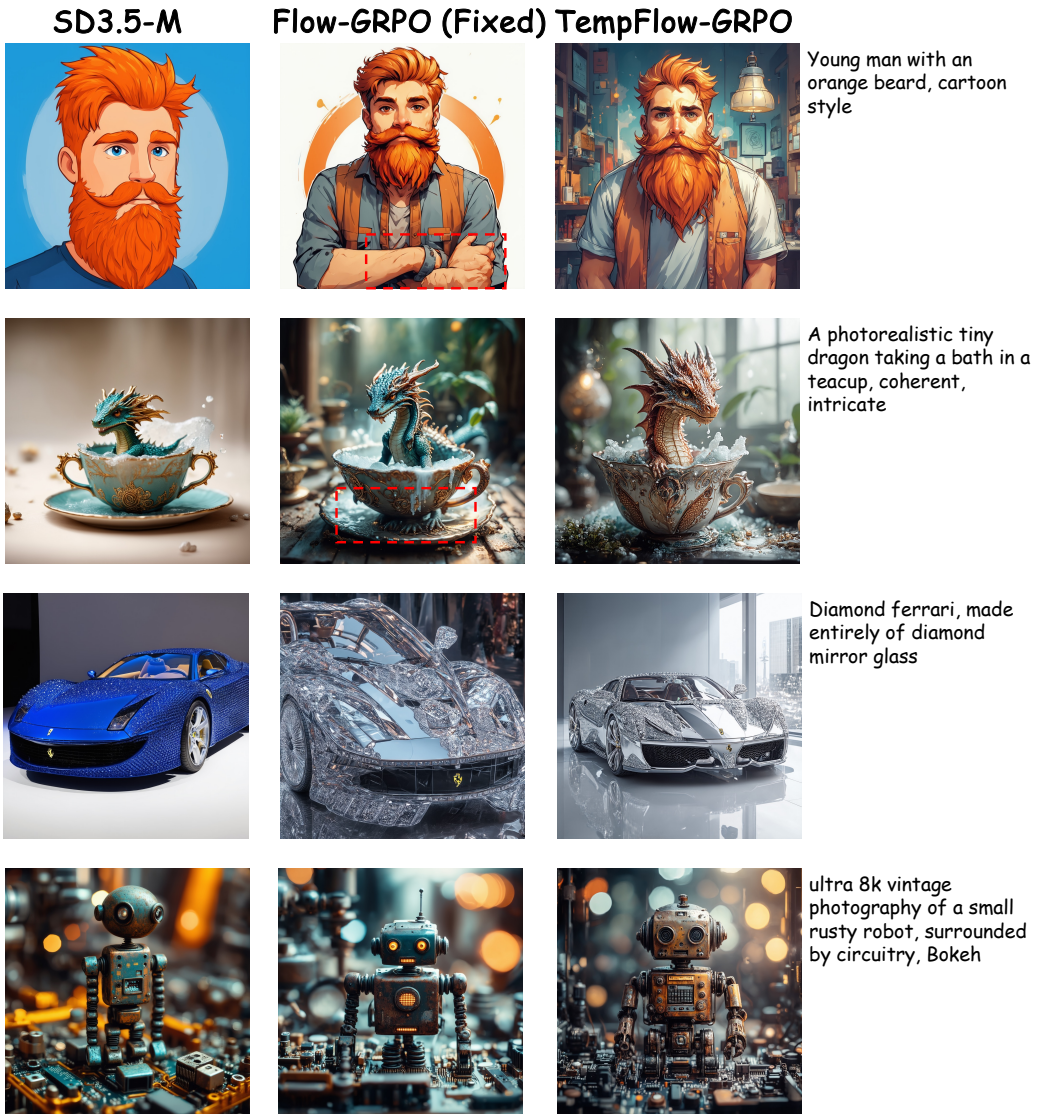

Figure 7: Qualitative comparison between SD3.5-M, Flow-GRPO and TempFlow-GRPO with PickScore as reward. We use the red box to indicate the error region.

on incorporating multi rewards from more powerful models, aiming to enhance performance across multiple dimensions. Additionally, we intend to design a comprehensive pipeline that can systematically improve various aspects of generative quality.

7 Conclusion

We presented TempFlow-GRPO, a temporally-aware reinforcement learning framework that addresses fundamental limitations in existing flow-based GRPO methods. Our key insight is that the uniform treatment of all timesteps creates a misalignment between optimization effort and actual impact on generation quality. Through trajectory branching, we enable precise credit assignment to intermediate actions without requiring specialized process reward models. Through noise-aware weighting, we ensure that optimization intensity matches each timestep's exploration potential, preventing both under-exploration of critical early decisions and over-optimization of minor refinements. Our extensive experiments demonstrate that TempFlow-GRPO achieves state-of-the-art per-

formance on human preference alignment and compositional generation benchmarks, surpassing both traditional GRPO methods and recent flow-based approaches.

REFERENCES

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Forest Labs Black et al. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

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

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

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.

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. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Yi Gu, Zhendong Wang, Yueqin Yin, Yujia Xie, and Mingyuan Zhou. Diffusion-rpo: Aligning diffusion models through relative preference optimization. *arXiv preprint arXiv:2406.06382*, 2024.

Jiwoo Hong, Sayak Paul, Noah Lee, Kashif Rasul, James Thorne, and Jongheon Jeong. Margin-aware preference optimization for aligning diffusion models without reference. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 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.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Zhanhao Liang, Yuhui Yuan, Shuyang Gu, Bohan Chen, Tiankai Hang, Ji Li, and Liang Zheng. Step-aware preference optimization: Aligning preference with denoising performance at each step. *arXiv preprint arXiv:2406.04314*, 2(5):7, 2024.

Zhanhao Liang, Yuhui Yuan, Shuyang Gu, Bohan Chen, Tiankai Hang, Mingxi Cheng, Ji Li, and Liang Zheng. Aesthetic post-training diffusion models from generic preferences with step-by-step preference optimization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 13199–13208, 2025.

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*, 2025.

Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Xingkai Yu, et al. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 7739–7751, 2025.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *arXiv preprint arXiv:2411.18704*, 2024.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

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.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

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.

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.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024a.

Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Chengyue Wu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024b.

Xin Xie and Dong Gong. Dymo: Training-free diffusion model alignment with dynamic multi-objective scheduling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 13220–13230, 2025.

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.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8941–8951, 2024.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.