Stepwise Credit Assignment for GRPO on Flow-Matching Models

Supplementary Material

In this appendix, we cover the following topics: A) our exploration on various design choices B) our improvement on SDE C) ablation studies D) our runtime analysis E) additional results F) comparisons with concurrent work.

A. Design Variations

We explored several alternative formulations of stepwise credit assignment to investigate potential improvements in variance reduction and temporal credit propagation. While these variations offer intuitive benefits in principle, our experiments show that the gain formulation from Sec. 5.2 performs best in practice.

A.1. Exponential Moving Average Baseline

Motivation. Raw gains $g_t^i = r_{t-1}^i - r_t^i$ can exhibit high variance across training iterations, potentially leading to noisy gradient estimates. A common variance reduction technique in RL is to center advantages using a baseline. We explore using an exponential moving average (EMA) of past gains as a temporal baseline.

Formulation. We maintain a per-step EMA baseline b_t updated after each training iteration:

$$b_t \leftarrow \alpha \cdot b_t + (1-\alpha) \cdot \mathbb{E}_i[g_t^i] \tag{10}$$

where $\alpha = 0.99$ is the decay rate. The centered gains are then:

$$\tilde{g}_t^i = g_t^i - b_t \tag{11}$$

These centered gains are normalized and converted to advantages as in the standard formulation.

Expected benefit. By subtracting a running average of typical gain magnitudes at each step, we hoped to reduce variance in advantage estimates while preserving the directional signal about which steps improve reward. This is analogous to value function baselines in policy gradient methods.

A.2. Generalized Advantage Estimation (GAE)

Motivation. Standard gains g_t^i only capture the immediate reward improvement from step t. However, a step's true contribution might include its downstream impact on future steps. GAE [39] provides a principled way to trade off bias and variance by exponentially weighting future gains.

Formulation. We apply GAE to the gain sequence with discount factor γ:

$$\text{GAE}_t^i = \sum_{k=0}^{T-t} \gamma^k g_{t+k}^i \tag{12}$$

In practice, we compute this efficiently via:

$$\text{GAE}_t^i = \frac{1}{\gamma^t} \sum_{k=t}^{T} \gamma^k g_k^i \tag{13}$$

We use $\gamma = 0.95$ following standard RL practice. The GAE values replace raw gains in the advantage calculation.

Expected benefit. GAE allows early steps to receive credit for setting up conditions that enable large future gains, even if their immediate gain g_t^i is small. For example, a compositional decision at $t \approx T$ might have small immediate reward improvement but enable larger gains at later refinement steps. We hypothesized this could improve credit assignment for temporally extended effects.

A.3. ODE-Based Progressive Distillation

Motivation. The SDE formulation requires introducing stochasticity σ_t for policy gradients, which can degrade sample quality. An alternative approach is to avoid the SDE entirely and instead optimize a progressive distillation objective between successive Tweedie estimates.

Formulation. Instead of computing log-probabilities under a Gaussian policy, we use a progressive distillation loss between the current step's Tweedie estimate $\hat{x}_0(t)$ and the twice-previous step's estimate $\hat{x}_0(t-2\Delta t)$:

$$\mathcal{L}_t^i = -\|\hat{x}_0^i(t-2\Delta t) - \mu_t^i\|^2 \tag{14}$$

where μ_t^i is the mean of the flow transition from x_t to $x_{t-2\Delta t}$:

$$\mu_t^i = \hat{x}_0^i(t) \cdot (1-(t-2\Delta t)) + \hat{x}_1^i(t) \cdot (t-2\Delta t) \tag{15}$$

We then scale this loss by gain-based advantages as in standard GRPO.

Expected benefit. This formulation completely avoids the stochasticity required for the SDE, potentially improving sample quality. The distillation objective encourages consistency between successive predictions of the final image, with stronger enforcement on steps with high gains. We hypothesized this could provide cleaner gradients while maintaining stepwise credit assignment.

A.4. Experimental Results

We evaluate all variations on the PickScore reward using the GenEval dataset, following the same training protocol as our main experiments.

Fig. 7 shows reward curves for all design variations compared to our standard gain formulation. No variation shows significant improvement over the standard formulation.

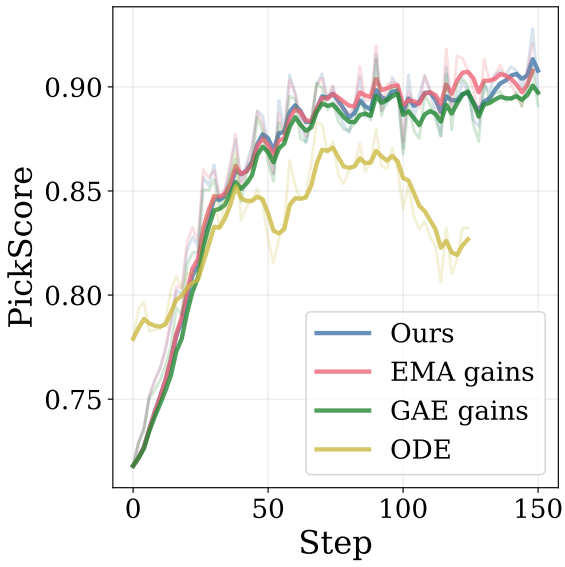

Figure 7. **Design variation comparison.** Reward vs. training iteration for different formulations of stepwise credit assignment on GenEval with PickScore reward. The standard gain formulation from the main paper matches all alternatives, demonstrating that preserving the natural temporal structure of diffusion gains is the most effective credit assignment.

The superior performance of using advantage calculated directly on the gains suggests that the natural temporal structure of diffusion generation should be preserved rather than normalized away. EMA centering and GAE both don't significantly improve with this structure, while progressive distillation is far less stable than the SDE formulation. So, in the main paper we keep the original, simpler implementation.

B. Improved SDE

Flow-GRPO's SDE generates noisy intermediate samples that degrade reward signal quality. While the SDE in Eq. (1) is theoretically sound for RL exploration, the noise injection mechanism produces visibly corrupted images at intermediate steps, see Fig. 8. Since reward models are typically trained on clean images, this noise substantially reduces the informativeness of reward signals, slowing RL convergence. We address this by replacing Flow-GRPO's SDE with a DDIM-inspired alternative that provides exploration while producing cleaner samples.

We construct an SDE that interpolates between deterministic ODE sampling and stochastic exploration while preserving the variance structure of the flow. Drawing inspiration from DDIM [43], we define the transition from step t to $t-\Delta t$ as $x_{t-\Delta t} =$

$$(1-(t-\Delta t))\,\hat{x}_0 + \sqrt{(t-\Delta t)^2-\sigma_t^2}\,\hat{x}_1 + \sigma_t \epsilon \tag{16}$$

where $\hat{x}_0 = x_t - t v_t$ and $\hat{x}_1 = x_t + (1-t)v_t$ are the predicted clean image and noise respectively given the flow prediction $v_t := v_\theta(x_t, t, c)$. Stochasticity is controlled with σ_t and $\epsilon \sim \mathcal{N}(0, I)$. When $\sigma_t = 0$, this recovers the deterministic ODE; when $\sigma_t > 0$, RL exploration is enabled through controlled noise injection. [1]

Rewriting Eq. (2) using $x_t = (1-t)\hat{x}_0 + t\hat{x}_1$, we obtain $x_{t-\Delta t} =$

$$(1-(t-\Delta t))\hat{x}_0 + \left((t-\Delta t) + \frac{\sigma_t^2 \Delta t}{2t}\right)\hat{x}_1 + \sigma_t\sqrt{\Delta t}\,\epsilon \tag{17}$$

Comparing with Eq. (16), the noise coefficients differ by $\sqrt{(t-\Delta t)^2-\sigma_t^2} \approx (t-\Delta t) - \frac{\sigma_t^2}{2(t-\Delta t)}$ via Taylor expansion. As $\Delta t \to 0$ and $\sigma_t \to 0$, the two formulations converge.

B.1. Variance-Preserving Property

The DDIM SDE exactly preserves the marginal variance at each step, while Flow-GRPO's SDE inflates it. For rectified flow, the training interpolation $x_t = (1-t)x_0 + t x_1$ implies that $\mathrm{Var}(x_t|x_0) = t^2$. Computing the variance of Eq. (16):

$$\mathrm{Var}(x_{t-\Delta t}|\hat{x}_0) \tag{18}$$
$$= ((t-\Delta t)^2 - \sigma_t^2)\,\mathrm{Var}(\hat{x}_1) + \sigma_t^2 \tag{19}$$

If $\mathrm{Var}(\hat{x}_1) = 1$, this exactly matches the expected variance. In contrast, from Eq. (17):

$$\mathrm{Var}_{\text{Flow}}(x_{t-\Delta t}|x_0) = \left((t-\Delta t) + \frac{\sigma_t^2\Delta t}{2t}\right)^2 + \sigma_t^2 \Delta t$$
$$> (t-\Delta t)^2 \quad \text{for any } \sigma_t > 0, \Delta t > 0 \tag{20}$$

This excess variance accumulates across steps, producing visibly noisier trajectories that confound reward evaluation.

B.2. Adaptive Noise Schedule

Setting $\sigma_t^2 = (t-\Delta t)^2$ in Eq. (16) yields:

$$x_{t-\Delta t} = (1-(t-\Delta t))\,\hat{x}_0 + (t-\Delta t)\epsilon \tag{21}$$

which completely discards the predicted noise $\hat{x}_1$. However, during training, the model learns that $x_t = (1-t)x_0 + t x_1$ where both x_0 and x_1 contain complementary information about the data distribution. At test time, both $\hat{x}_0$ and $\hat{x}_1$ carry

[1] This is exactly equal to the DDIM update if we let $\alpha_{t-\Delta t} = (1-(t-\Delta t))^2$ and $\beta_t = (t-\Delta t)^2$, where α_t and β_t are the signal and noise strengths in the paper respectively.

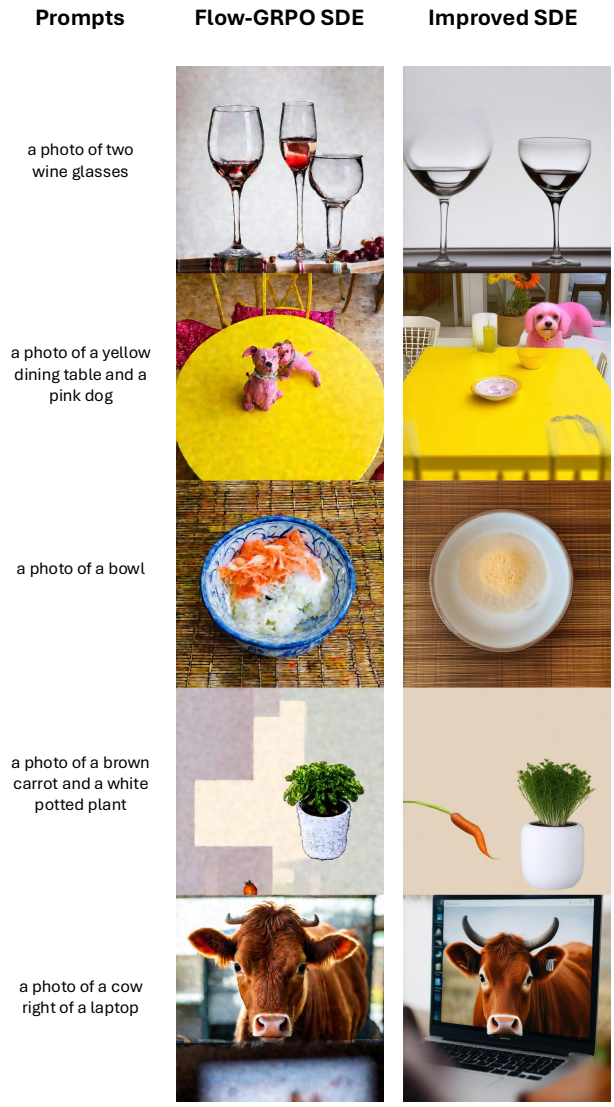

Figure 8. **Improved SDE produces cleaner images.** Qualitative comparison between Flow-GRPO SDE (middle) and our DDIM-inspired SDE (right) from Sec. 5.5. The improved formulation generates images that are much less noisy while maintaining stochasticity for policy gradients.

entangled information about the true clean image, i.e., $\hat{x}_1$ is not merely noise to be replaced, but rather encodes structure that the model has learned to extract from x_t. Fully replacing $\hat{x}_1$ with independent noise ϵ produces samples of the form $(1-t)\hat{x}_0 + t\epsilon$, which lie outside the model's training distribution and yield poor velocity predictions v_t.

Setting $\sigma_t = \eta(t - \Delta t)$ retains a $(1 - \eta^2)$ fraction of the predicted noise structure. This interpolates between deterministic sampling ($\eta = 0$) and full replacement ($\eta = 1$), providing RL exploration while maintaining samples within the model's learned distribution.

However, this uniform schedule treats all steps equally despite their differential impact on the final output. Noise injected at step t propagates through the remaining denoising steps and affects x_0 through the accumulated flow. To quantify this sensitivity, consider the Jacobian $J_t = \partial x_0 / \partial x_t$, which evolves according to the tangent flow equation:

$$\frac{dJ_t}{dt} = J_t \cdot \frac{\partial v_t(x_t, t, c)}{\partial x_t}, \quad J_0 = I \tag{22}$$

While exact computation is intractable, first-order analysis suggests $\|J_t\|_F^2 \approx 1 + ct$ for some constant $c > 0$ when $\|\partial v_t / \partial x_t\|_F = O(1)$ (typical for normalized neural networks). This indicates that perturbations at earlier steps (larger t) have amplified influence on the final image.

Setting $\sigma_t = \eta t\sqrt{1-t}$ provides this compensation: the $\sqrt{1-t}$ factor naturally downweights exploration at early steps (when $t \to 1$, $\sqrt{1-t} \to 0$) where sensitivity is highest, while allowing more exploration at later steps where perturbations have localized effects. This makes the sensitivity-weighted exploration $\sigma_t^2 \cdot (1 + ct)$ more uniform across the trajectory, ensuring that all steps contribute roughly equally to the RL exploration budget.

B.3. RL Objective

We apply the same GRPO objective as Liu et al. [26] but with the DDIM SDE as our policy. The marginal probability for Eq. (16) is:

$$\pi_\theta(x_{t-\Delta t} | x_t, c) = \mathcal{N}\left(x_{t-\Delta t}; \mu_t, \sigma_t^2 I\right) \tag{23}$$

where $\mu_t = (1-(t-\Delta t))\hat{x}_0 + \sqrt{(t-\Delta t)^2 - \sigma_t^2}\,\hat{x}_1$. We optimize the objective in Eq. (4) using this policy, inheriting the same clipping, KL regularization, and group-relative advantages from Flow-GRPO. The cleaner intermediate samples from our SDE enable more accurate reward evaluation, particularly for vision-based reward models sensitive to image quality.

C. Ablation Studies

We conduct ablation studies to validate key design choices in Stepwise-Flow-GRPO: the gain normalization strategy and the number of ODE substeps for computing intermediate clean image estimates. All experiments use PickScore reward on the GenEval dataset with the same training protocol as our main experiments.

C.1. Gain Normalization Strategy

A critical design choice is whether to normalize gains g_t^i globally across all steps and trajectories (joint normalization) or separately at each step (per-step normalization). This decision affects how the temporal magnitude of gains is preserved in the final advantages.

Joint normalization (our method): Compute mean and standard deviation across all steps and trajectories:

$$\tilde{A}_t^i = \frac{g_t^i - \mu_t}{\sigma_{\text{global}}}, \tag{24}$$

$$\mu_{\text{global}} = \frac{1}{NT}\sum_{j,k} g_k^j, \quad \sigma_{\text{global}} = \frac{1}{NT}\sum_{j,k}(g_k^j - \mu_{\text{global}})^2 \tag{25}$$

This preserves the relative magnitudes across steps, so early gains with naturally larger values receive proportionally larger advantages.

Stepwise normalization: Compute mean and standard deviation separately for each step:

$$\tilde{A}_t^i = \frac{g_t^i - \mu_t}{\sigma_t}, \tag{26}$$

$$\mu_t = \frac{1}{N}\sum_{j} g_t^j, \quad \sigma_t = \frac{1}{N}\sum_{j}(g_t^j - \mu_t)^2 \tag{27}$$

This equalizes the importance of all steps regardless of their natural gain magnitudes.

Fig. 9 shows that joint normalization significantly outperforms per-step normalization in convergence speed. Both methods eventually converge to similar final rewards, indicating that stepwise normalization doesn't improve final quality, only slows down learning.

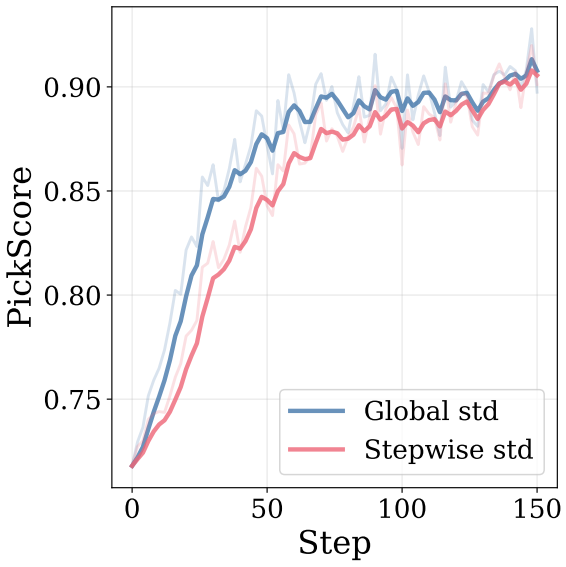

Figure 9. **Joint normalization preserves temporal structure and accelerates convergence.** Reward vs. training iteration comparing joint normalization (global mean/std across all steps and trajectories) against per-step normalization (separate mean/std for each step).

C.2. Number of Denoising Substeps

Computing intermediate reward estimates $r_t^i = R(\hat{x}_0(t), c)$ requires denoising from noisy state x_t to obtain a clean image estimate $\hat{x}_0(t)$. The number of ODE substeps T' controls the tradeoff between estimate quality and computational cost.

We compare $T' \in \{2, 5, 8\}$ on PickScore reward using the GenEval dataset, measuring both reward convergence and wall-clock time. Fig. 10 shows that all choices of T' achieve similar performance in both training iterations and wall-clock time, demonstrating that our method is robust to this hyperparameter. We select $T' = 5$ as our default, though the results suggest practitioners can adjust this based on their computational constraints without significantly impacting final performance.

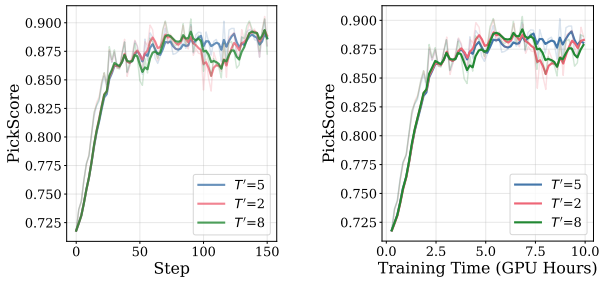

Figure 10. **Stepwise-Flow-GRPO is robust to the number of denoising substeps.** Reward vs. training iteration (top) and GPU Hours (bottom) for different numbers of substeps $T' \in \{2, 5, 8\}$ used to compute $\hat{x}_0(t)$. All settings achieve similar convergence speed and final performance, validating that our method is not sensitive to this hyperparameter choice.

D. Runtime Analysis

Stepwise-Flow-GRPO incurs additional computational cost from: (1) computing intermediate estimates $\hat{x}_0(t)$ via $T' = 5$ ODE substeps from each x_t, and (2) evaluating the reward model on these estimates at each step. Tab. 2 shows per-iteration timing on 8 NVIDIA A100 GPUs with batch size 16.

Generation overhead. Our method requires approximately 1.8-2.4× more generation time than Flow-GRPO due to additional ODE integration (50 extra steps for $T = 10$ steps). These intermediate denoising steps are embarrassingly parallel and batch efficiently.

Reward evaluation overhead. The overhead varies by reward model: PickScore (lightweight CNN) adds minimal cost (2.8s vs 1.4s), ImageReward shows 10× overhead (8.2s vs 0.7s), and UnifiedReward (7B VLM) dominates computation (114.6s vs 48.8s). For UnifiedReward, we use 8 separate A100 80GB GPUs with SGLang for efficient

Method (Reward)	Generation (s)	Reward eval (s)
Ours (PickScore)	24.4 ± 0.0	2.8 ± 0.0
Flow-GRPO (PickScore)	13.7 ± 0.9	1.4 ± 1.3
Ours (ImageReward)	29.8 ± 0.2	8.2 ± 0.2
Flow-GRPO (ImageReward)	7.9 ± 0.0	0.7 ± 0.0
Ours (UnifiedReward)	136.1 ± 5.8	114.6 ± 5.8
Flow-GRPO (UnifiedReward)	56.0 ± 30.9	48.8 ± 30.9

Table 2. **Per-iteration timing breakdown** in seconds per training iteration, averaged over multiple runs.

batched inference.

Implementation optimizations. We implement several optimizations to minimize overhead: (1) Batched intermediate denoising: All $T' = 5$ substeps for a given x_t are batched together. (2) Parallel reward evaluation: Intermediate estimates $\{\hat{x}_0(t)\}_{t=1}^{T}$ are evaluated in parallel across steps. (3) Asynchronous execution: Generation and reward evaluation are pipelined when possible.

Crucially, despite the 1.8-2.4× per-iteration slowdown, Stepwise-Flow-GRPO achieves *faster overall convergence in wall-clock time* across all settings (Fig. 5 in main paper). The superior sample efficiency—requiring 2-3× fewer training iterations to reach target rewards—more than compensates for the per-iteration overhead. In practice, our method converges 20-40% faster in total wall-clock time depending on the reward function, demonstrating that the improved learning signal justifies the additional computation.

E. Additional Results

We provide extended experimental results including: (1) OCR text rendering evaluation, (2) longer training runs with the GenEval reward, and (3) extended training with UnifiedReward. These experiments validate that our method's advantages extend across diverse reward functions and training durations.

E.1. OCR Text Rendering

We evaluate on a specialized OCR dataset using a combined reward: 80% OCR accuracy + 20% PickScore. This challenging compositional task requires rendering readable text while maintaining visual quality.

Fig. 11 shows that Stepwise-Flow-GRPO substantially outperforms Flow-GRPO. Flow-GRPO diverges after ∼500 steps, while our method continues improving and plateaus at a significantly higher reward. This demonstrates particularly strong benefits for hierarchical compositional tasks where early steps establish structure (letter shapes, spacing) that later steps refine.

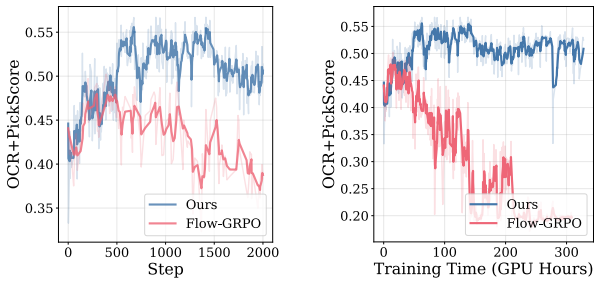

Figure 11. **OCR text rendering results.** Reward vs. training iteration (top) and wall-clock time (bottom) using combined OCR+PickScore reward (80% OCR, 20% PickScore). Flow-GRPO diverges after 500 steps while Stepwise-Flow-GRPO continues improving, demonstrating superior stability and final performance on compositional text rendering.

E.2. Extended GenEval Training

To test scalability, we conduct a 400 GPU hour training run on GenEval. Fig. 12 shows that after 400 hours, Stepwise-Flow-GRPO achieves 0.87 overall GenEval score—substantially outperforming Flow-GRPO (0.72 from paper) and even beats state-of-the-art models like GPT-4o (0.84).

The performance gap *widens* with extended training, particularly on challenging categories: our method achieves 0.89 on counting, 0.73 on spatial positioning, and 0.80 on attribute binding. These categories require precise compositional decisions that benefit most from accurate credit assignment. This suggests that appropriate credit assignment becomes more critical as models approach high performance levels.

E.3. UnifiedReward Training

We validate sustained advantages with UnifiedReward-7b-v1.5, a large vision-language model that evaluates caption alignment and visual quality.

UnifiedReward prompt: *"You are given a text caption and a generated image based on that caption. Your task is to evaluate this image based on two key criteria: (1) Alignment with the Caption: Assess how well this image aligns with the provided caption. Consider the accuracy of depicted objects, their relationships, and attributes. (2) Overall Image Quality: Examine the visual quality including clarity, detail preservation, color accuracy, and aesthetic appeal. Assign a score from 1 to 5 after 'Final Score:'."*

Fig. 13 shows that after 60 GPU hours, Stepwise-Flow-GRPO achieves 0.74 GenEval score (Tab. 3) with smooth, stable training curves. Our method maintains consistent efficiency advantages throughout extended training.

Training stability. Notably, Flow-GRPO with the standard SDE formulation consistently diverged when training

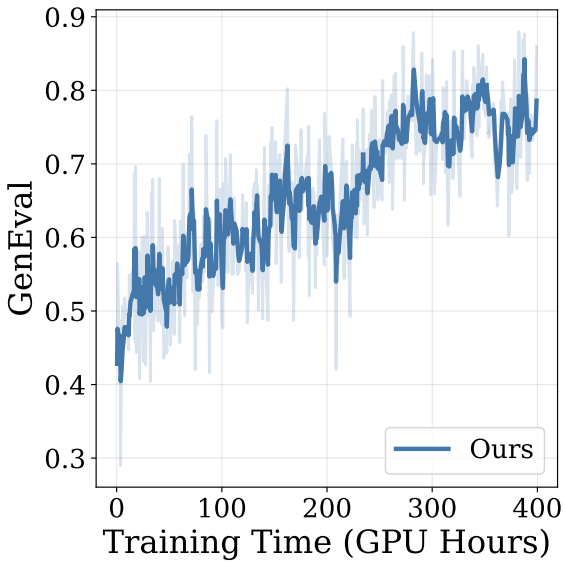

Figure 12. **Extended GenEval training.** GenEval overall score vs. wall-clock time for 400 GPU hour runs. Stepwise-Flow-GRPO achieves 0.87, substantially outperforming Flow-GRPO (0.72) and approaching state-of-the-art autoregressive models. The widening performance gap demonstrates that stepwise credit assignment provides increasing benefits at high performance levels.

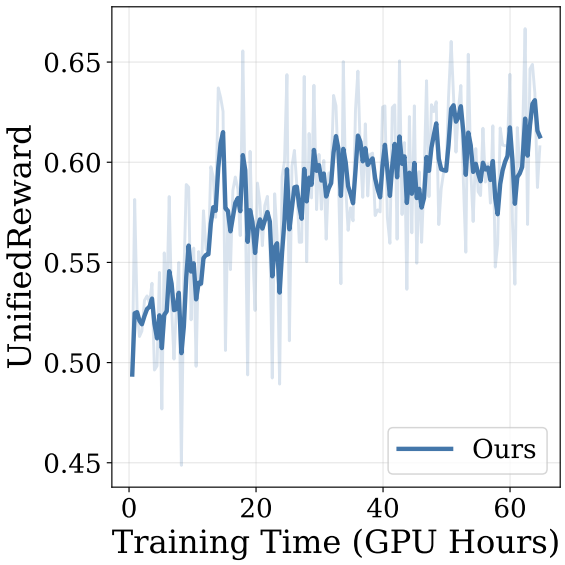

Figure 13. **UnifiedReward training.** Reward vs. wall-clock time for 60 GPU hour run on GenEval using UnifiedReward-7b-v1.5. Stepwise-Flow-GRPO trains stably while Flow-GRPO diverges with this reward function, demonstrating superior robustness with complex VLM-based rewards.

with UnifiedReward, preventing stable optimization. In contrast, our method trains stably throughout, demonstrating that stepwise credit assignment provides not only efficiency improvements but also fundamental stability benefits when using complex reward models. This increased robustness is particularly valuable for large VLM-based rewards where gradient noise can be substantial.

E.4. Qualitative Results

Figs. 15 to 17 provide extended qualitative comparisons between Flow-GRPO and Stepwise-Flow-GRPO. Each row shows a single prompt with the base model output (step 0) and results from both methods at training steps 60 and 120. Our method produces consistently better images, with the most pronounced differences at step 60 when the performance gap is largest.

Stepwise-Flow-GRPO shows clear improvements in **counting** (e.g., "three oranges", "four clocks", "a tennis racket and a bird"), **real-world dynamics** (e.g., "broccoli and a vase", "a white dining table and a red car"), and **overall image quality** (e.g., "a red train and a purple bear", "bed"). These improvements are consistent with our method's ability to assign credit to the denoising steps that establish composition and object placement, rather than uniformly rewarding the entire trajectory.

F. Comparison with Concurrent Work

TempFlow-GRPO [16] and Granular-GRPO [58] are concurrent works that also address the uniform credit assignment limitation in Flow-GRPO. All three methods recognize that early denoising steps have outsized impact on final image quality, but the approaches differ in several key respects.

What is optimized. TempFlow-GRPO and Granular-GRPO both use the standard GRPO advantage $\hat{A}^i = (R(x_0^i, c) - \text{mean})/\text{std}$, i.e., the normalized *final reward*, with per-step attribution via trajectory branching. In contrast, we optimize telescoping gains $g_t^i = r_{t-1}^i - r_t^i$, directly rewarding each step's *marginal improvement*. This captures causal contribution rather than terminal correlation.

How early steps are emphasized. TempFlow-GRPO applies a hand-designed noise-level weighting $\text{Norm}(\sigma_t\sqrt{\Delta t})$ to discount advantages at later steps. Granular-GRPO optimizes only the first 8 of 16 steps, leaving later steps unoptimized. Our gains are *data-dependent*: as shown in Fig. 2, gain magnitudes naturally decrease as $t \to 0$, automatically concentrating optimization on early steps without manual schedules or step selection.

Sampling efficiency. Both TempFlow-GRPO and Granular-GRPO use ODE→SDE→ODE branching, requiring $O(T^2K)$ forward passes for T steps and K branches. We use the SDE throughout with few-step Tweedie estimation, which is embarrassingly parallel and empirically

Model	Overall	Single Obj.	Two Objs.	Counting	Colors	Position	Attr. Binding
Pretrained Models							
SD3.5-M (cfg=1.0)	0.28	0.71	0.23	0.15	0.45	0.05	0.08
SD3.5-M (cfg=4.5)	0.63	0.98	0.78	0.50	0.81	0.24	0.52
Standard Training Duration							
Flow-GRPO (cfg=1.0, PickScore)	0.60	0.96	0.73	0.67	0.67	0.21	0.35
Ours (cfg=1.0, PickScore)	0.60	0.96	0.75	0.67	0.67	0.21	0.34
Flow-GRPO (cfg=4.5, PickScore)	0.68	0.98	0.82	0.64	0.82	0.24	0.59
Ours (cfg=4.5, PickScore)	**0.71**	**0.98**	**0.85**	**0.70**	**0.82**	**0.29**	**0.59**
Extended Training							
Flow-GRPO (cfg=4.5, GenEval, 400 GPU hrs)	0.72	–	–	–	–	–	–
Ours (cfg=4.5, UnifiedReward, 60 GPU hrs)	0.74	0.99	0.89	0.73	0.83	0.34	0.66
Ours (cfg=4.5, GenEval, 400 GPU hrs)	**0.87**	**0.99**	**0.93**	**0.89**	**0.87**	**0.73**	**0.80**
Reference: State-of-the-art Models							
Janus-Pro-7B	0.80	0.99	0.89	0.59	0.90	0.79	0.66
SANA-1.5 4.8B	0.81	0.99	0.93	0.86	0.84	0.59	0.65
GPT-4o	0.84	0.99	0.92	0.85	0.92	0.75	0.61

Table 3. **Complete GenEval results.** After 400 GPU hours, our method achieves 0.87 overall, substantially outperforming Flow-GRPO (0.72) and approaching GPT-4o (0.84). Flow-GRPO extended results from [26]; reference models from respective papers.

faster—running TempFlow-GRPO's released code at equivalent batch sizes, we observed our method to be approximately $1.5\times$ faster per epoch.

Figure 14. **Qualitative comparison across training objectives.** Generated images from GenEval prompts using base SD3.5-M (left), GenEval reward training (middle), and UnifiedReward training (right). While GenEval reward training improves prompt adherence and benchmark scores, UnifiedReward training produces higher overall visual quality and more photorealistic images.

Figure 15. **Extended qualitative comparison (page 1).** Each row shows a single prompt with the base model generation (step 0), followed by Flow-GRPO and Stepwise-Flow-GRPO (Ours) at training steps 60 and 120, both trained with PickScore reward. The gap is most visible at step 60, where our method demonstrates better counting, more plausible compositions, and higher image quality.

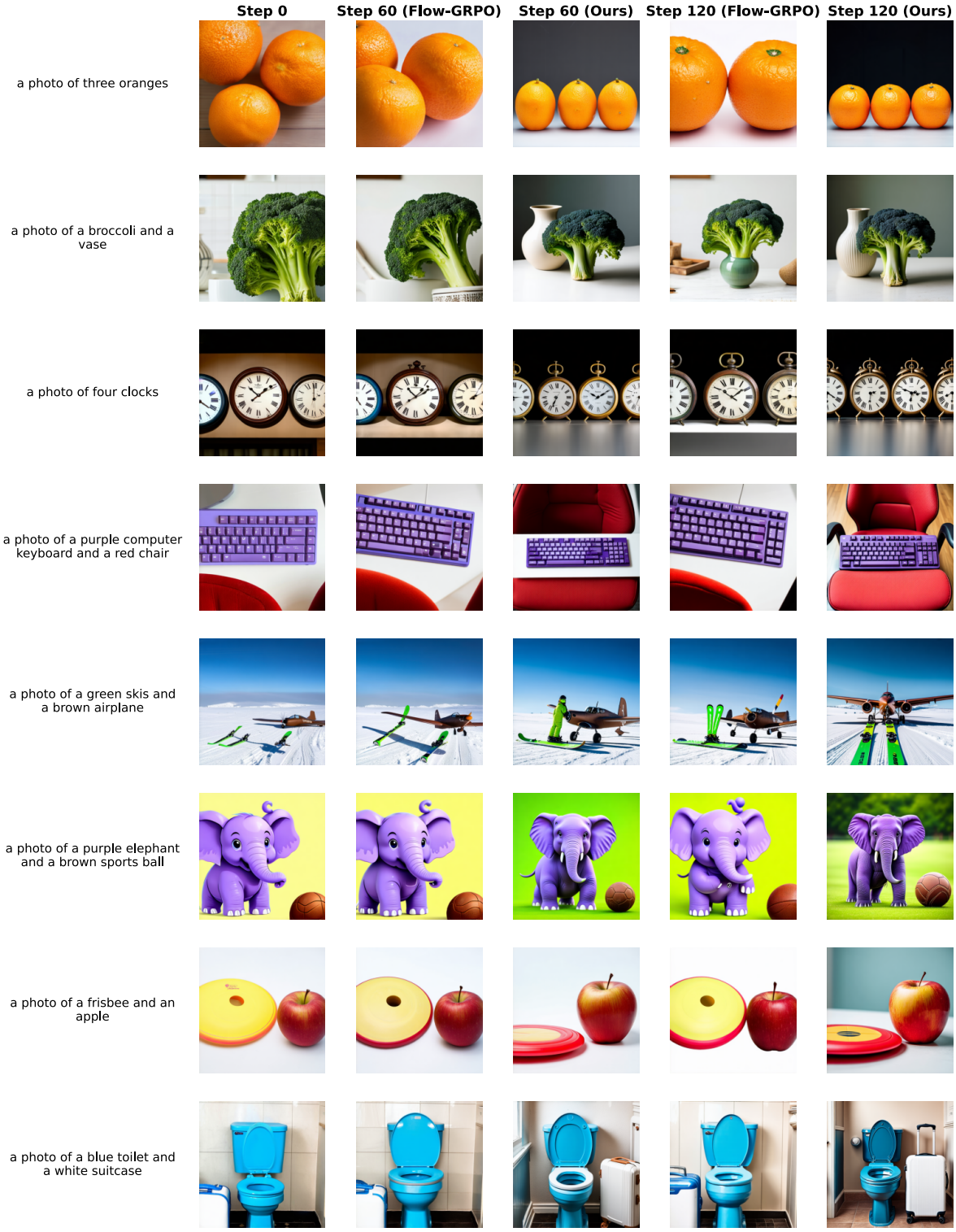

Figure 16. **Extended qualitative comparison (page 2).** Same layout as Fig. 15. Stepwise-Flow-GRPO consistently produces more accurate object counts, better spatial arrangements, and higher overall quality than Flow-GRPO across training.

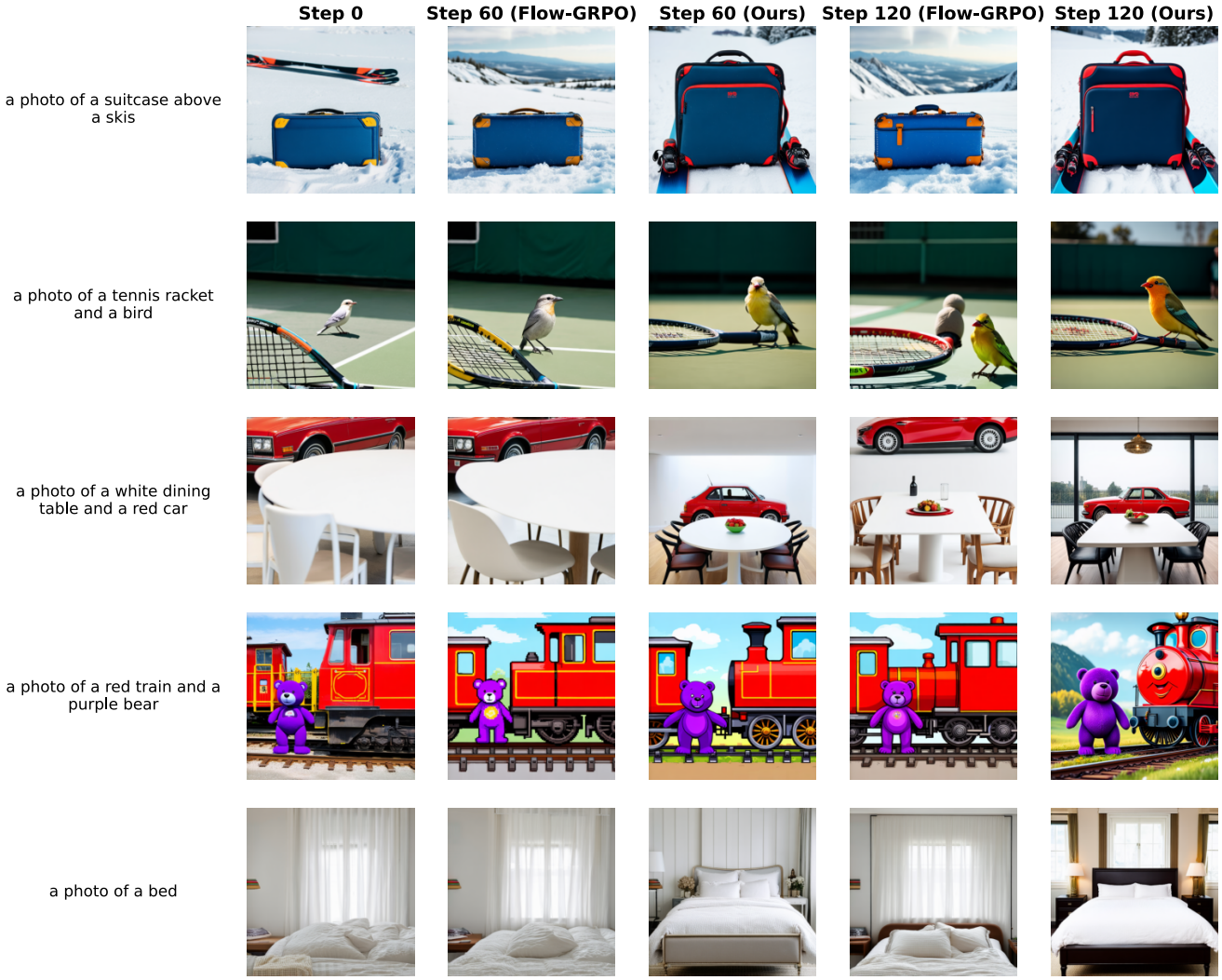

Figure 17. **Extended qualitative comparison (page 3).** Same layout as Fig. 15. Our method maintains qualitative advantages throughout training, with the largest visible differences at step 60.