TMPO: Trajectory Matching Policy Optimization for Diverse and Efficient Diffusion Alignment

Jiaming Li[1,2,*] **Chenyu Zhu**[1,*] **Nanxi Yi**[1] **Youjun Bao**[2] **Li Sun**[2] **Quanying Lv**[2]
Xiang Fang[3] **Daizong Liu**[4] **Jianjun Li**[1] **Kun He**[1] **Bowen Zhou**[5] **Zhiyuan Ma**[1,†,✉]

[1]Huazhong University of Science and Technology [2]Kuaishou Technology
[3]Nanyang Technological University [4]Wuhan University [5]Tsinghua University
[*]Equal contribution. [†]Corresponding author.

Abstract

Reinforcement learning (RL) has shown extraordinary potential in aligning diffusion models to downstream tasks, yet most of them still suffer from significant reward hacking, which degrades generative diversity and quality by inducing visual mode collapse and amplifying unreliable rewards. We identify the root cause as the *mode-seeking* nature of these methods, which maximize expected reward without effectively constraining probability distribution over acceptable trajectories, causing concentration on a few high-reward paths. In contrast, we propose **Trajectory Matching Policy Optimization (TMPO)**, which replaces scalar reward maximization with trajectory-level reward distribution matching. Specifically, TMPO introduces a **Softmax Trajectory Balance (Softmax-TB)** objective to match the policy probabilities of K trajectories to a reward-induced Boltzmann distribution. We prove that this objective inherits the *mode-covering* property of forward KL divergence, preserving coverage over all acceptable trajectories while optimizing reward. To further reduce multi-trajectory training time on large-scale flow-matching models, TMPO incorporates **Dynamic Stochastic Tree Sampling**, where trajectories share denoising prefixes and branch at dynamically scheduled steps, reducing redundant computation while improving training effectiveness. Extensive results across diverse alignment tasks such as human preference, compositional generation and text rendering show that TMPO improves generative diversity over state-of-the-art methods by 9.1%, and achieves competitive performance in all downstream and efficiency metrics, attaining the optimal trade-off between reward and diversity.

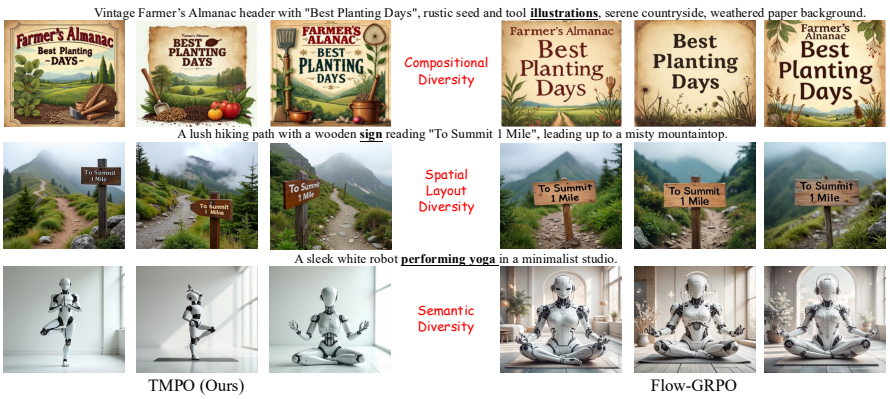


Figure 1: Generative diversity comparison between TMPO (Ours) and Flow-GRPO.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

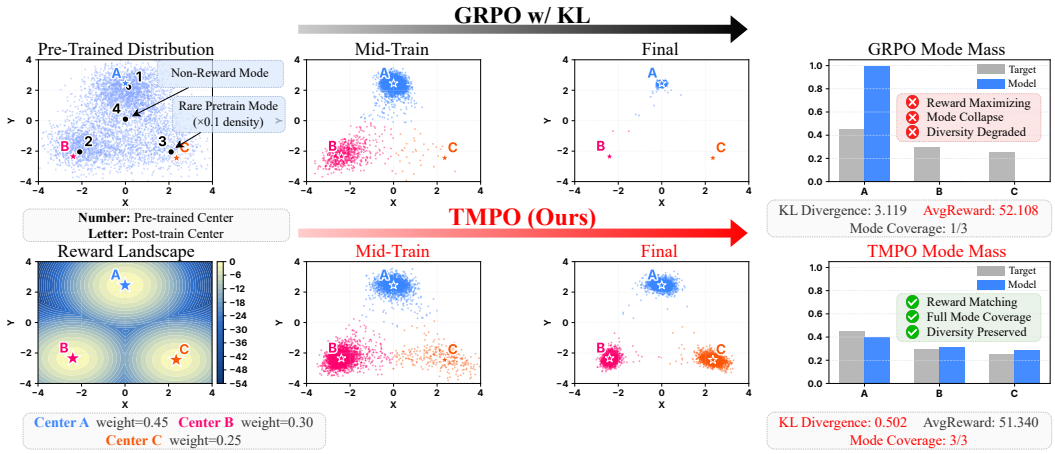


Figure 2: A toy experiment on a three-layer MLP diffusion model pre-trained on a Gaussian mixture distribution containing non-reward modes and rare density modes. The post-training objective is to match a multimodal reward distribution. GRPO-based reward maximization collapses to a single high-reward mode even with KL regularization, whereas TMPO covers all reward clusters by matching trajectory-level reward distributions while maintaining a high average reward.

Recently, RL has achieved remarkable success in aligning Large Language Models (LLMs), generative diffusion models and flow-matching models. However, existing methods still suffer from significant *reward hacking* issue, which implies the model may ignore the distribution diversity and instead take a shortcut by over-reinforcing a small proportion of the recognized high-reward modes. Similar to LLM, this dilemma can be called *visual mode collapse*. While mitigating reward hacking has been extensively studied in the field of LLMs [49, 50], the issue remains largely unexplored for diffusion or flow models, which raises two critical issues: ***1) Degradation of Diversity:*** for language models, a problem may solely own a single definite answer, whereas for generative models (*i.e.,* text-to-image models), the same prompt may correspond to many plausible outputs that vary in *composition*, *spatial layout*, and *scene semantics*. Reward hacking, however, drives the model to generate images with only a small set of compositions and styles, degrading generative diversity. (Fig. 1). ***2) Amplification of Unreliable Rewards:*** for text-to-image models, both model-based rewards (*e.g.,* ImageReward [10], HPS [11], PickScore [31]) and rule-based rewards (*e.g.,* OCR accuracy and GenEval [44]) are intrinsically imperfect proxies for human preferences [12–15, 37]. Reward hacking on these proxy rewards amplifies spuriously rewarded image attributes, thereby deteriorating generation quality. This raises a core question in generative diffusion alignment: *How can we preserve the anisotropic distribution of outputs learned by the model from pre-training data while encouraging the model to satisfy rewards with specific preferences?*

The reward hacking behavior in diffusion model alignment can be attributed to the training objectives of current diffusion-based RL methods. Existing methods use reward maximization as the optimization objective [4–7, 26, 30], which we prove is intrinsically *mode-seeking*. In particular, mode-seeking models tend to explore high-reward trajectory, rather than covering all reasonable counterparts, as illustrated by the GRPO example in Fig. 2. **An ideal optimization objective should be *mode-covering*, which encourages the model to cover as many reasonable solutions as possible.** This analysis further suggests that alleviating reward hacking requires a fundamental shift in the optimization objective. Existing methods mainly attempt to address the issue of reward hacking through two lines of work: ***1) Constraining Over-optimization:*** KL regularization [5, 6, 15, 38, 39] is used to constrain the deviation of the policy from the reference model, but it does not alter the reward-maximization objective, resulting in limited improvement in generative diversity. ***2) Improving Training Objective:*** Generative Flow Network (GFlowNet)-based methods [23, 24, 29] seek to generate outputs with probabilities proportional to their rewards, but existing methods usually establish local constraints on individual denoising steps, which requires estimating an intractable state-flow function for policy updates, thereby introducing additional training error. Overall, existing mitigation methods still fall short of providing a direct mode-covering objective over the full denoising trajectory.

To address this gap, we propose **Trajectory Matching Policy Optimization (TMPO)**. Inspired by the reward distribution matching nature of GFlowNets [21], TMPO fundamentally shifts the training objective from reward maximization to trajectory-level reward distribution matching. We prove that TMPO is *mode-covering* and can naturally encourage the coverage of reasonable solutions, thereby alleviating reward hacking from the ground up, as shown in Fig. 2. Specifically, we introduce a **Softmax Trajectory Balance (Softmax-TB)** objective, which performs reward distribution matching on the complete denoising trajectory. For K trajectories generated from the same starting point, Softmax-TB matches the generation probabilities of these trajectories to a target Boltzmann distribution computed from rewards ($P_\theta(\tau) \propto \exp(\beta R(\tau))$). Meanwhile, by normalizing the trajectory probabilities $P_\theta(\tau)$ and exponential rewards $\exp(\beta R(\tau))$ among the K trajectories within each group, we eliminate the intractable partition function, yielding a simpler and directly optimizable objective. To reduce the training time of fully sampling multiple trajectories under Softmax-TB [29], TMPO further introduces **Dynamic Stochastic Tree Sampling** [26, 27, 30], which allows trajectories to share denoising steps through a tree structure, reducing redundant computation and lowering training time. By dynamically adjusting the branching points, the tree sampler also promotes effective exploration at different denoising stages and improves training performance.

We evaluate TMPO on FLUX.1-dev across three alignment tasks: compositional image generation, visual text rendering, and human preference alignment. Results show that TMPO obtains the highest diversity metrics across all tasks, improving generative diversity by 9.1% on average over existing state-of-the-art methods, while also achieving the best GenEval accuracy and PickScore. Moreover, TMPO reduces training time by up to 27% compared with state-of-the-art methods and achieves the most favorable trade-offs in both reward-diversity and reward-efficiency comparisons, as shown in Fig. 6. These benefits also extend to the SD3.5-Medium backbone.

Our contributions are as follows: **(1)** We identify reward maximization as the root cause of reward hacking in diffusion-based RL and reformulate the optimization objective as reward distribution matching. **(2)** We introduce **Softmax Trajectory Balance**, a partition-function-free trajectory-level distribution matching objective, and prove that its mode-covering property naturally preserves generative diversity. **(3)** We introduce **Dynamic Stochastic Tree Sampling**, which substantially reduces the training time of our trajectory balance objective and enables scalable training on large-scale flow-matching models.

2 Related Work

Mitigating reward hacking in diffusion model alignment. Diffusion-based RL alignment is highly susceptible to reward hacking under model-based or rule-based rewards [13, 14, 16, 18]. Existing mitigations, including KL regularization [5, 6], pairwise preference rewards [20], and diversity-aware regularization [15], still optimize scalar rewards and thus inherit mode-collapse risks [17]. TMPO addresses this limitation by reformulating scalar rewards as a preference distribution to be matched, encouraging coverage within each sampled trajectory group.

Distribution matching and GFlowNet-based alignment. GFlowNets train policies with terminal distributions proportional to reward [21]. DGFS [23] uses subtrajectory constraints, while DAG [29] and ∇-GFlowNet [24] use local balance, with the latter also using reward gradients. These approaches fit intermediate state flows. TMPO instead matches complete trajectories without intermediate-flow estimation; the short three-transition FLUX setting motivates this design (Appendix E.1). FlowRL [25] applies Boltzmann reward matching to LLM reasoning with a learned prompt-conditioned normalizer. TMPO shares the distribution-matching motivation but cancels the normalizer within each group and uses a different surrogate update, as detailed below and tested in Section 5.3.

Efficient trajectory sampling in diffusion model alignment. RL post-training for large-scale flow models is bottlenecked by redundant rollouts, and complete-trajectory objectives amplify this cost by requiring policy probabilities over multiple trajectories [29]. MixGRPO reduces stochastic optimization to selected SDE windows [26], while Dynamic-TreeRPO and TreeGRPO amortize early computation by sharing denoising prefixes [27, 30]. TMPO adopts prefix sharing and further introduces dynamic branching schedules to maintain effective exploration across training stages.

3 Preliminaries

3.1 Trajectory Balance and Reward Matching

Generative Flow Networks (GFlowNets) train stochastic policies to sample terminal objects in proportion to non-negative rewards [21]. For a trajectory $\tau = (s_0 \to \cdots \to s_n)$ ending at $x = s_n$ with reward $R(x)$, Trajectory Balance (TB) imposes: [22]

$$Z_\theta \prod_{t=1}^{n} P_F(s_t \mid s_{t-1}; \theta) = R(x) \prod_{t=1}^{n} P_B(s_{t-1} \mid s_t; \theta), \tag{1}$$

where P_F, P_B, and Z_θ are the forward policy, backward policy, and total flow. This reward-proportional view preserves multiple high-reward modes. Diffusion GFlowNet methods and FlowRL motivate reward distribution matching instead of scalar reward maximization [23–25].

3.2 ODE-to-SDE Conversion for Stochastic Diffusion Policies

Modern text-to-image generators are commonly built on diffusion or flow-matching models [1–3, 9, 32, 34, 35]. In rectified flow, a clean sample x_0 and noise $x_1 \sim \mathcal{N}(0, \mathbf{I})$ are connected by:

$$x_t = (1-t)x_0 + tx_1, \qquad dx_t = v_\theta(x_t, t, c)\, dt, \tag{2}$$

Deterministic ODE solvers [28, 33] lack the transition probabilities required by policy-gradient RL. Following prior work [6–8], we convert the probability-flow ODE to an equivalent SDE that admits tractable likelihoods for credit assignment while preserving marginals:

$$dx_t = \left[v_\theta(x_t, t, c) + \frac{1}{2}\sigma^2(t) \nabla_x \log p_\theta(x_t \mid c, t) \right] dt + \sigma(t) dW_t. \tag{3}$$

Here $\sigma(t)$ controls the noise scale, with $\sigma(t) \equiv 0$ recovering the deterministic ODE.

4 Trajectory Matching Policy Optimization

In this section, we present **Trajectory Matching Policy Optimization (TMPO)** for online reward-induced trajectory distribution matching in flow-matching text-to-image models. We adapt GFlowNet trajectory balance to within-prompt trajectory groups, yielding a partition-free **Softmax-TB** objective with provable mode-covering properties (Section 4.1). We then describe how reward-bearing samples are efficiently collected through prefix-sharing SDE tree rollouts with **Dynamic Stochastic Tree Sampling**. The overall pipeline is illustrated in Figure 3.

4.1 From GFlowNets to Softmax Trajectory Balance

GFlowNets learn a policy π_θ such that $P_\theta(\tau) \propto R(\tau)$ by minimizing a trajectory balance (TB) residual:

$$\mathcal{L}_{\text{TB}}(\tau) = \big(\log Z + \log P_\theta(\tau) - \log R(\tau)\big)^2, \tag{4}$$

where Z is the partition function typically estimated via an auxiliary network. In TMPO's tree topology, all K trajectories for the same prompt share a common Z, so normalizing within the group cancels the partition function exactly: $P_\theta(\tau_i)/\sum_j P_\theta(\tau_j) = R(\tau_i)/\sum_j R(\tau_j)$ (Appendix A).

To introduce adjustable sharpness, we replace raw rewards with a Boltzmann target $p^*(\tau) \propto \exp(\beta R(\tau))$, where β controls mode concentration. This is the maximum-entropy distribution subject to the expected reward constraint (Appendix A). Substituting into the within-group matching and defining shorthand $q_i \triangleq \mathrm{softmax}_i(\beta R)$ and $p_i \triangleq P_\theta(\tau_i)/\sum_j P_\theta(\tau_j)$, we obtain the **Softmax-TB advantage**:

$$A_i = \log q_i - \log p_i = \log \frac{\exp(\beta R_i)}{\sum_{j=1}^{K} \exp(\beta R_j)} - \log \frac{P_\theta(\tau_i)}{\sum_{j=1}^{K} P_\theta(\tau_j)} \tag{5}$$

The following result shows that the Softmax-TB advantage is intimately connected to forward KL minimization.

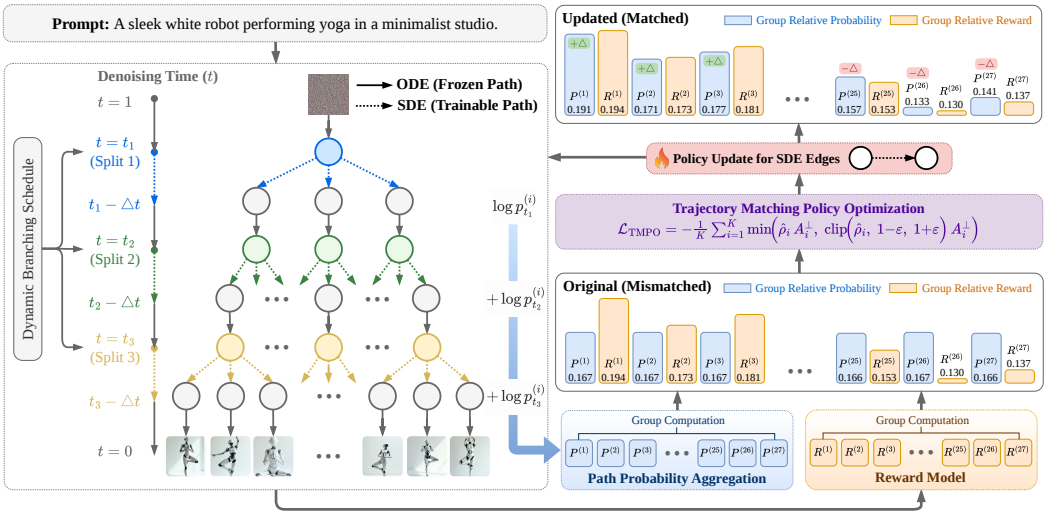


Figure 3: Overview of the TMPO framework. For each prompt, Dynamic Stochastic Tree Sampling shares a deterministic ODE prefix and injects SDE noise at three curriculum-scheduled branch points, yielding 3^3=27 trajectories with reduced compute. Terminal images are scored by a reward model, and Softmax Trajectory Balance converts the rewards into a Boltzmann target, computing the per-trajectory advantage as the log-ratio between the target and policy distributions.

Mode-Covering Equilibrium. Let $q_i = \exp(\beta R_i)/\sum_{j=1}^{K}\exp(\beta R_j)$ denote the within-group Boltzmann target and $p_i = P_\theta(\tau_i)/\sum_{j=1}^{K} P_\theta(\tau_j)$ the within-group policy distribution. Then the Boltzmann-weighted Softmax-TB advantage equals the within-group forward KL divergence:

$$\sum_{i=1}^{K} q_i\, A_i = D_{\mathrm{KL}}^{(K)}(q_\beta \,\|\, p_\theta) \;\geq\; 0,$$

with equality if and only if $p_i = q_i$ for all i. Because $D_{\mathrm{KL}}^{(K)}(q\|p) \to +\infty$ whenever $p_i \to 0$ for any i with $q_i > 0$, Softmax-TB inherently penalizes under-coverage of any mode that carries positive Boltzmann weight. Within-group normalization cancels Z_β and Z_θ, making this conditional forward KL exactly computable over the K observed trajectories, without requiring independent samples from the global intractable q_β (extended proof and quantitative bounds in Appendix A.2).

Proof. By definition, $A_i = \log q_i - \log p_i$. Substituting into the Boltzmann-weighted sum: $\sum_i q_i A_i = \sum_i q_i(\log q_i - \log p_i) = D_{\mathrm{KL}}^{(K)}(q\|p)$. Non-negativity and the equality condition follow from Gibbs' inequality. Since both q and p are normalized within the group, the global partition function Z_β and the policy's total probability Z_θ cancel and need not be estimated. □

Reverse KL vs. Forward KL. Standard policy gradient methods sample from π_θ and therefore exhibit reverse-KL-like, mode-seeking behavior: under the entropy-regularized Boltzmann formulation, reward maximization (e.g., Flow-GRPO, TreeGRPO) can be interpreted as minimizing $D_{\mathrm{KL}}(p_\theta\|q_\beta)$ (Appendix A.3). Its advantage $A_i \propto R_i - \bar{R}$ is agnostic to the policy probability p_i, so dropping a mode incurs no direct gradient penalty. In contrast, TMPO's log-ratio advantage $A_i = \log(q_i/p_i)$ is distribution-aware: it diverges whenever any mode with positive Boltzmann weight is under-covered ($p_i \to 0 \Rightarrow A_i \to +\infty$), inheriting the mode-covering behavior of forward KL despite sampling from the policy. **This distinction—the distribution-aware advantage of Equation (7)—provides the theoretical basis for diversity preservation without auxiliary KL regularization.**

Complete Softmax-TB Objective Since A_i involves the shared softmax denominator that couples all K trajectories, we detach it as a stop-gradient signal and route the gradient exclusively through

the IS ratio $\hat{\rho}_i$. The complete TMPO loss is:

$$\boxed{\mathcal{L}_{\text{TMPO}} = -\frac{1}{K}\sum_{i=1}^{K} \min\Big(\hat{\rho}_i\, A_i^{\perp},\ \text{clip}\big(\hat{\rho}_i,\ 1{-}\varepsilon,\ 1{+}\varepsilon\big)\, A_i^{\perp}\Big)} \tag{6}$$

$$A_i = \log \frac{\exp(\beta R_i)}{\sum_{j=1}^{K}\exp(\beta R_j)} - \log \frac{P_\theta(\tau_i)}{\sum_{j=1}^{K} P_\theta(\tau_j)}, \qquad \hat{\rho}_i = \prod_{t=1}^{T} \frac{\pi_\theta(x_{s_t^-}^{(i)} \mid x_{s_t}^{(i)})}{\pi_{\theta_{\text{old}}}(x_{s_t^-}^{(i)} \mid x_{s_t}^{(i)})} \tag{7}$$

Here $A_i^{\perp}$ denotes the stop-gradient advantage and $\hat{\rho}_i$ is clipped to $[1{-}\varepsilon, 1{+}\varepsilon]$ for trust-region protection [36]. The per-step log-ratios are centered via RatioNorm [19] to remove the deterministic negative shift of Gaussian transition kernels and restore symmetric clipping (Appendix B).

Relation to FlowRL. Both methods start from Boltzmann reward matching, but implement it differently. FlowRL [25] jointly learns a prompt-conditioned normalizer Z_ϕ and the policy through a squared trajectory-balance residual, weighted by a detached clipped importance ratio. TMPO instead normalizes reward and trajectory-probability shares within the same group, eliminating Z_ϕ. It detaches the log-ratio advantage A_i and routes policy gradients through the importance ratio in Equation (6). Both methods use importance weighting and clipping; the distinction lies in normalization, the surrogate loss, and the gradient path. Section 5.3 tests a FlowRL-style diffusion adaptation.

4.2 Dynamic Stochastic Tree Sampling

Unlike the independent parallel rollouts in standard GRPO-based methods [6, 7], TMPO places three consecutive stochastic branch points along the denoising trajectory, producing a $3^3{=}27$ terminal tree per prompt. At each branch point, $B{=}3$ child trajectories are spawned from independent noise realizations. Early branch points at high noise levels introduce coarse-grained diversity across trajectory groups, while later branch points at lower noise levels supply finer-grained structured variations. Shared prefix computations before each bifurcation substantially reduce the cost of RL training on large-scale models such as FLUX.

Dynamic Branching Schedule Branch positions follow a curriculum that shifts from early high-noise regions toward later structured regions as training progresses ($p = \text{clip}(u/U, 0, 1)$). To prevent overfitting to a fixed tree geometry, each position is stochastically perturbed:

$$\xi_i \sim \text{Beta}(\kappa\bar{\mu}_i(p),\ \kappa(1{-}\bar{\mu}_i(p))), \quad \tilde{s}_i = \lfloor s_{\min} + (s_{\max} - s_{\min})\xi_i + 0.5 \rfloor \tag{8}$$

with final indices obtained by sorting. At each branch point s_i, child nodes are generated by injecting SDE noise with magnitude $\gamma(\sigma) = \eta\sqrt{\sigma/(1{-}\sigma)}\sqrt{-\Delta t}$, where $\eta{=}0.7$ following the CPS recommendation [48]:

$$x_{s_i^-}^{(r)} = \mu_\theta(x_{s_i}, s_i) + \gamma(\sigma_{s_i})\varepsilon_r, \quad \varepsilon_r \sim \mathcal{N}(0, \mathbf{I}) \tag{9}$$

Between branch points, the denoising trajectory follows deterministic ODE steps that do not require gradient computation. The trajectory log-probability accumulates only the T stochastic transitions at branch points: $\log P_\theta(\tau) = \sum_{i=1}^{T} \log \pi_\theta(x_{s_i^-} \mid x_{s_i})$, so gradient back-propagation is confined to T steps rather than the full denoising horizon, substantially reducing per-iteration training cost. Full derivations of the SDE noise injection, Beta branching schedule, and log-probability computation are provided in Appendix C.

5 Experiments

This section empirically evaluates TMPO on three tasks: (1) Compositional Image Generation, (2) Visual Text Rendering, and (3) Human Preference Alignment. Beyond alignment quality, we also assess generative diversity and per-iteration training efficiency.

Table 1: Performance on Compositional Image Generation, Visual Text Rendering, and Human Preference benchmarks on FLUX.1-dev, evaluated by task performance on task-specific test prompts and by preference scores and diversity on DrawBench prompts. Best values are **bold** and second-best are underlined. Light gray rows denote the pretrained baseline without RL training. Dark gray rows denote our approach. Orange cells mark the proxy reward used for training in each section. ImgRwd: ImageReward; Cos. Div.: DINOv2-space cosine diversity.

Method	Time (s) ↓	Task Metric ↑		Human Preference ↑			Diversity ↑	
		GenEval	OCR Acc.	HPS-v2.1	ImgRwd	PickScore	LGMD	Cos. Div.
Compositional Image Generation								
FLUX.1-dev	—	0.647	—	0.301	1.099	22.301	-0.031	0.211
Flow-GRPO	160.7	0.943	—	0.285	1.078	22.097	-0.092	0.201
MixGRPO	150.8	0.932	—	**0.310**	1.086	21.850	-0.271	0.183
TreeGRPO	125.9	0.939	—	0.278	1.079	21.540	-0.285	0.181
GARDO	165.3	0.923	—	0.289	1.091	22.245	0.012	0.238
TMPO (Ours)	**91.9**	**0.949**	—	0.306	**1.159**	**22.901**	**0.131**	**0.247**
Visual Text Rendering								
FLUX.1-dev	—	—	0.591	0.292	**1.121**	21.968	-0.040	0.215
Flow-GRPO	121.3	—	**0.940**	0.291	1.100	21.365	-0.092	0.208
MixGRPO	116.2	—	0.919	0.282	1.098	20.878	-0.302	0.179
TreeGRPO	93.4	—	0.921	0.290	1.110	21.099	-0.292	0.180
GARDO	123.8	—	0.931	0.291	1.085	22.087	0.064	0.228
TMPO (Ours)	**76.3**	—	0.935	**0.310**	1.111	**22.309**	**0.110**	**0.241**
Human Preference Alignment								
FLUX.1-dev	—	—	—	0.310	1.119	22.604	-0.056	0.214
Flow-GRPO	108.9	—	—	**0.377**	1.600	24.210	-0.107	0.209
MixGRPO	103.5	—	—	0.367	**1.621**	23.934	-0.232	0.183
TreeGRPO	79.1	—	—	0.369	1.580	23.657	-0.284	0.176
GARDO	112.7	—	—	0.343	1.571	23.959	-0.060	0.212
TMPO (Ours)	**68.3**	—	—	0.373	1.610	**24.277**	**0.204**	**0.252**

5.1 Experimental setup

Models, prompts, and metrics. We use FLUX.1-dev as the backbone, fine-tuned with LoRA [40] (rank r=64, α=128) targeting all attention and feed-forward projections in each transformer block. All images are generated at 512×512 resolution. Training rollouts use 6 denoising steps for efficiency; all evaluations use 28 steps. Preference-alignment training prompts are drawn from HPDv2 [11]; preference and diversity scores in Table 1 use DrawBench prompts. For compositional image generation (GenEval) and visual text rendering (OCR), we construct task-specific training and test prompt sets. All reported evaluations generate 10 images per prompt. Task-specific metrics include GenEval [44] accuracy for composition, OCR accuracy ($1-$NED) for text rendering, and HPS-v2.1, ImageReward, PickScore for preference alignment. Diversity is measured by two complementary metrics. **Log Geometric Mean Distance (LGMD)** computes the log geometric mean of pairwise distances in VAE [45] latent space: LGMD $= \frac{2}{N(N-1)} \sum_{i<j} \log\big(\|\phi(x_i) - \phi(x_j)\|_2/\sqrt{D}\big)$, where $\phi(\cdot)$ is the flattened VAE latent and D the feature dimension; positive values indicate healthy diversity, while negative values signal mode collapse. **Cosine Diversity (Cos. Div.)** follows GARDO [15] and measures the mean pairwise cosine distance in DINOv2 [41] ViT-L/14 feature space: Cos. Div. $= \frac{2}{N(N-1)} \sum_{i<j}(1 - \cos(\psi(x_i), \psi(x_j)))$. LGMD captures low-level structural duplicates, while Cos. Div. captures semantic layout and texture differences.

Training protocols and baselines. We consider three single-reward protocols, namely GenEval only (compositional accuracy), OCR only (text rendering accuracy), and PickScore [31] only (human preference), as well as joint preference training with HPS-v2.1, ImageReward, and PickScore at equal weight, where each reward is independently z-score normalized within the K-trajectory group before summation. TMPO uses a three-level prefix-sharing tree with $K=3^3=27$ terminal trajectories per prompt. TMPO is compared against Flow-GRPO [6], MixGRPO [26], TreeGRPO [30], and GARDO [15] under matched backbone and reward settings. Reference-policy KL coefficients are 0.03 for Flow-GRPO, MixGRPO, and TreeGRPO, and 0.04 for GARDO; TMPO uses no auxiliary reference-policy KL penalty. Full hyperparameters, model details, and compute resources are provided in Appendix D.

5.2 Main results

Table 1 summarizes the three single-reward protocols, Table 2 evaluates joint preference training, and Section 5.4 evaluates SD3.5-Medium. Appendix E reports FlowRL-style training checkpoints, PP-OCR results, GenEval category scores, and matched-execution runtime.

Compositional image generation. Under GenEval only training, TMPO achieves the best compositional accuracy (0.949) while preserving superior diversity on both LGMD and Cosine Diversity. Despite the sparse binary reward signal, Softmax-TB effectively distributes probability mass toward compositionally correct trajectories without collapsing onto a narrow set of solutions. The category breakdown shows that TMPO is best or tied best in five of six categories, with counting only 0.002 below Flow-GRPO (Appendix E.3).

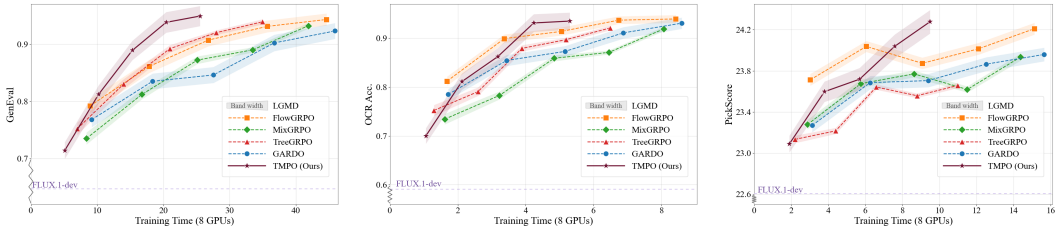
Visual text rendering. TMPO achieves the best iteration time, the highest HPS-v2.1 and PickScore, and the best diversity on both LGMD and Cosine Diversity. Although Flow-GRPO reaches the highest EasyOCR accuracy, it substantially reduces diversity, indicating stronger overfitting to the task reward. A separate PP-OCR retraining experiment favors TMPO on OCR accuracy and diversity (Appendix E.2, Table 8).

Human preference alignment. Under PickScore only training, TMPO obtains the best PickScore and diversity, with competitive HPS-v2.1 and ImageReward. Figure 4 shows training curves across all three protocols; qualitative examples are shown in Figure 5.

Joint preference training. With the three normalized preference rewards weighted equally (Table 2), TMPO achieves the best HPS-v2.1 and diversity, second-best ImageReward and PickScore, and the lowest iteration time (72.1 s).

Table 2: Joint preference training on FLUX.1-dev. RL methods use HPS-v2.1, ImageReward, and PickScore at equal weight. Best values are **bold** and second-best are underlined.

Method	Time (s) ↓	HPS-v2.1 ↑	ImgRwd ↑	PickScore ↑	LGMD ↑	Cos. Div. ↑
FLUX.1-dev	—	0.310	1.119	22.604	-0.056	0.221
Flow-GRPO	119.4	0.342	**1.622**	22.769	-0.102	0.199
MixGRPO	106.5	0.351	1.590	23.717	-0.308	0.175
TreeGRPO	87.6	0.355	1.603	**23.951**	-0.282	0.178
GARDO	125.9	0.341	1.574	22.927	-0.184	0.209
TMPO (Ours)	**72.1**	**0.360**	1.613	23.842	**0.132**	**0.243**



(a) GenEval (composition) (b) OCR Acc. (text rendering) (c) PickScore (preference)

Figure 4: Training curves across three single-reward protocols. Each plot shows the task reward versus wall-clock training time. TMPO converges faster and reaches higher final reward than all baselines, benefiting from both prefix-sharing tree sampling and Softmax-TB distribution matching.

Efficiency and diversity. TMPO improves the Pareto frontier between alignment, diversity, and training cost. Because trajectory log-probabilities accumulate only over $T=3$ stochastic branch points, gradient back-propagation is confined to these transitions rather than the full denoising horizon; combined with prefix sharing that amortizes forward computation, TMPO reduces iteration time by $\sim$20% relative to TreeGRPO across three single-reward settings (Table 1; joint training in Table 2). More importantly, the LGMD gap is consistent across all settings: GRPO-style baselines often raise reward scores while driving LGMD negative, whereas TMPO keeps LGMD positive in every setting.

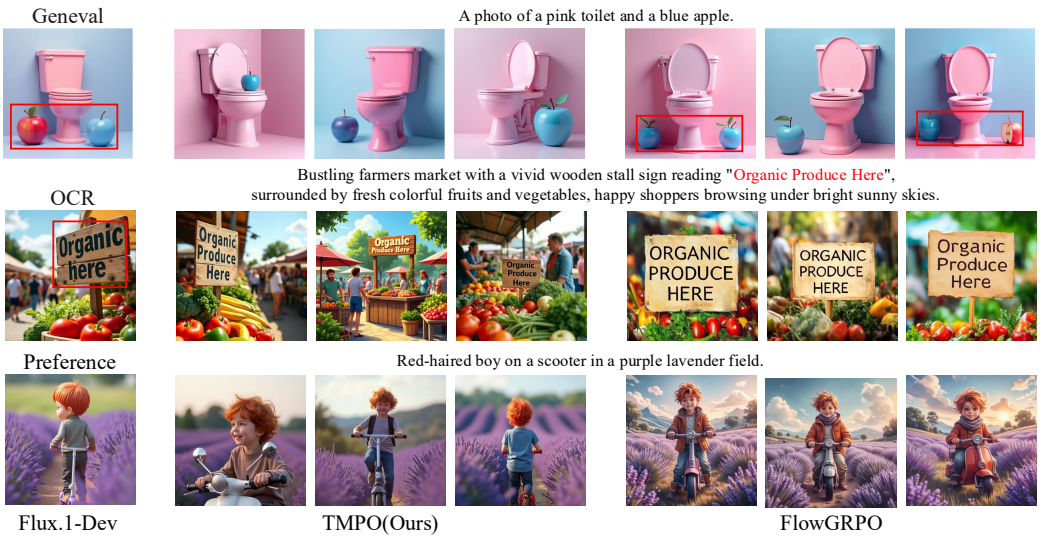


Figure 5: Qualitative results of different alignment methods. TMPO produces diverse, high-fidelity images that faithfully follow the text prompt.

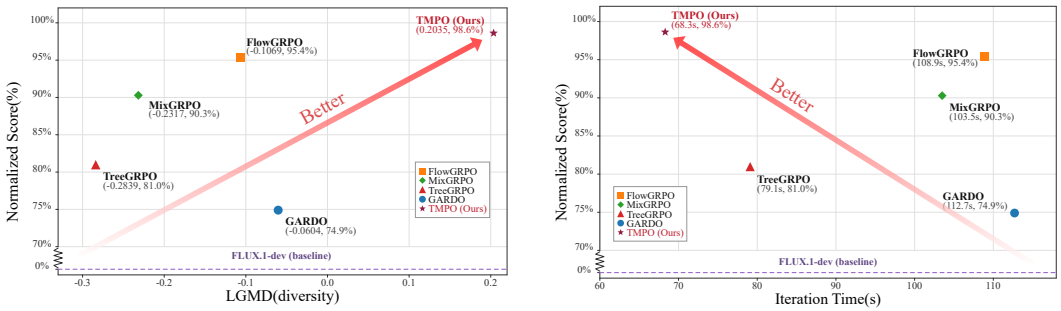


Figure 6: Pareto analysis of TMPO against GRPO-based alignment methods. **Left:** normalized reward score versus generation diversity measured by LGMD. **Right:** normalized reward score versus iteration time. Normalized scores are computed via min-max normalization across all compared methods for each metric. TMPO achieves the most favorable trade-off in both reward-diversity and reward-efficiency comparisons.

5.3 Comparison with FlowRL-style Optimization

To examine the operational distinction in Section 4, we adapt FlowRL's learned-normalizer trajectory-balance approach to FLUX.1-dev on OCR, pretraining the normalizer offline before joint policy updates. Over 250 steps, the FlowRL-style adaptation falls from 0.591 to 0.429 OCR, whereas TMPO reaches 0.935; FlowRL-style's final PickScore is 20.681, below base FLUX's 21.968. Appendix E.1 gives the checkpoints and discusses normalizer–policy coupling. These results concern the diffusion adaptation evaluated here.

5.4 Generalization to SD3.5-Medium

To verify that TMPO generalizes beyond the FLUX.1-dev backbone, we evaluate all methods under the PickScore only protocol on SD3.5-Medium [47] with LoRA fine-tuning. All methods use the same LoRA rank and learning rate; all other hyperparameters follow their respective FLUX configurations.

The results mirror the FLUX.1-dev findings: TMPO achieves the best scores on all metrics with a consistent $\sim$37% iteration-time reduction over Flow-GRPO. Flow-GRPO, MixGRPO, and TreeGRPO again exhibit negative LGMD; GARDO preserves positive LGMD through its explicit diversity mechanism but still trails TMPO on both diversity and reward metrics, confirming that Softmax-TB provides a stronger diversity guarantee without auxiliary reweighting.

Table 3: PickScore only training on SD3.5-Medium. All methods use LoRA fine-tuning. Best values are **bold** and second-best are underlined.

Method	Time (s) ↓	HPS-v2.1 ↑	ImgRwd ↑	PickScore ↑	LGMD ↑	Cos. Div. ↑
SD3.5-Medium	—	0.272	0.893	21.592	-0.061	0.213
Flow-GRPO	101.8	0.355	1.329	23.289	-0.123	0.195
MixGRPO	96.2	0.348	1.351	23.019	-0.238	0.175
TreeGRPO	74.6	0.351	1.298	22.817	-0.289	0.168
GARDO	105.1	0.329	1.335	23.102	0.072	0.227
TMPO (Ours)	**63.9**	**0.361**	**1.358**	**23.551**	**0.118**	**0.239**

5.5 Ablation study

We ablate major components of TMPO under the PickScore only protocol (Table 4), covering three categories: *objective* (reward temperature β), *balance granularity* (trajectory vs. step-wise), and *sampling strategy* (tree structure, dynamic branching).

Table 4: Ablation study on FLUX.1-dev under the PickScore only protocol. Time denotes iteration time in seconds.

Variant	Time ↓	HPS-v2.1 ↑	ImageReward ↑	PickScore ↑	LGMD ↑	Cos. Div. ↑
TMPO (Full)	**68.3**	**0.373**	**1.610**	**24.277**	**0.204**	**0.252**
w/o Reward Temp. β (β=1)	68.5	0.352	1.501	23.891	0.199	0.250
Step-wise Balance (Detailed)	87.2	0.339	1.139	23.107	0.012	0.223
w/o Tree Structure (K=27)	123.7	0.369	1.598	24.109	0.173	0.245
w/o Dynamic Branching	78.7	0.361	1.553	23.878	0.121	0.239

The results confirm the contribution of each component. **w/o Reward Temp.** β removes the β-warmup schedule and fixes β=1 throughout training. Without progressive sharpening, the target distribution remains flat, slowing reward improvement (PickScore $24.28 \rightarrow 23.89$); however, the mild optimization pressure preserves diversity well (LGMD 0.199, Cos. Div. 0.250). **Step-wise detailed balance** enforces per-step flow matching, causing severe loss oscillation and convergence difficulty. The substantial drop in ImageReward ($1.61 \rightarrow 1.14$) and near-zero LGMD (0.012) show that local supervision fragments credit assignment across the trajectory. **w/o Tree Structure** uses K=27 independent rollouts and achieves comparable alignment, confirming the generality of Softmax-TB; however, the original 123.7 s timing includes chunked recomputation under per-GPU group placement. Using the baseline execution scheme instead takes 105.9 s, close to Flow-GRPO's 108.9 s; the full tree takes 68.3 s (Appendix E.4). **w/o Dynamic Branching** fixes branch positions throughout training, resulting in lower diversity (ΔLGMD $= -0.083$, ΔCos. Div. $= -0.013$) and higher iteration time (78.7 s vs. 68.3 s).

6 Conclusion

We presented TMPO, which reformulates diffusion RL alignment as trajectory-level distribution matching via Softmax Trajectory Balance, a partition-free objective whose advantage characterizes the forward KL divergence, inheriting mode-covering diversity guarantees without auxiliary regularization. Combined with prefix-sharing tree rollouts and dynamic branching, TMPO achieves the best reward–diversity trade-off across all evaluated settings while reducing iteration time by up to 27%.

Limitations & Future Work. This work focuses on T2I alignment; two natural extensions are: (1) the Boltzmann target quality is upper-bounded by the underlying reward model; pairing TMPO with stronger reward models may yield further improvements, and (2) extending trajectory-level distribution matching to video generation, 3D synthesis, and robotic action generation.

Acknowledgments

This work was sponsored by the National Natural Science Foundation of China (No. 62406161), the CCF-Kuaishou Large Model Explorer Fund (No. CCF-KuaiShou 2025003), and the CCF-Tencent Rhino-Bird Open Research Fund (No. CCF-Tencent RAGR20260103).

References

[1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[2] 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 *ICML*, 2024.

[3] Black Forest Labs. FLUX. https://github.com/black-forest-labs/flux, 2024.

[4] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *ICLR*, 2024.

[5] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *NeurIPS*, 2023.

[6] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *NeurIPS*, 2025.

[7] 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.

[8] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[9] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[10] 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. In *NeurIPS*, 2023.

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

[12] Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. TextDiffuser: Diffusion models as text painters. In *NeurIPS*, 2023.

[13] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *ICLR*, 2022.

[14] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *NeurIPS*, 2022.

[15] Haoran He, Yuxiao Ye, Jie Liu, Jiajun Liang, Zhiyong Wang, Ziyang Yuan, Xintao Wang, Hangyu Mao, Pengfei Wan, and Ling Pan. GARDO: Reinforcing diffusion models without reward hacking. *arXiv preprint arXiv:2512.24138*, 2025.

[16] Yunqi Hong, Kuei-Chun Kao, Hengguang Zhou, and Cho-Jui Hsieh. Understanding reward hacking in text-to-image reinforcement learning. *arXiv preprint arXiv:2601.03468*, 2026.

[17] 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.

[18] Jie Wu, Yu Gao, Zilyu Ye, Ming Li, Liang Li, Hanzhong Guo, Jie Liu, Zeyue Xue, Xiaoxia Hou, Wei Liu, et al. RewardDance: Reward scaling in visual generation. *arXiv preprint arXiv:2509.08826*, 2025.

[19] Jing Wang, Jiajun Liang, Jie Liu, Henglin Liu, Gongye Liu, Jun Zheng, Wanyuan Pang, Ao Ma, Zhenyu Xie, Xintao Wang, et al. GRPO-Guard: Mitigating implicit over-optimization in flow matching via regulated clipping. *arXiv preprint arXiv:2510.22319*, 2025.

[20] Yibin Wang, Zhimin Li, Yuhang Zang, Yujie Zhou, Jiazi Bu, Chunyu Wang, Qinglin Lu, Cheng Jin, and Jiaqi Wang. Pref-GRPO: Pairwise preference reward-based GRPO for stable text-to-image reinforcement learning. *arXiv preprint arXiv:2508.20751*, 2025.

[21] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *NeurIPS*, 2021.

[22] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. In *NeurIPS*, 2022.

[23] Dinghuai Zhang, Ricky T. Q. Chen, Chenghao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *ICLR*, 2024.

[24] Zhen Liu, Tim Z. Xiao, Weiyang Liu, Yoshua Bengio, and Dinghuai Zhang. Efficient diversity-preserving diffusion alignment via gradient-informed GFlowNets. In *ICLR*, 2025.

[25] Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, et al. FlowRL: Matching reward distributions for LLM reasoning. In *ICLR*, 2026.

[26] Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Yiming Cheng, Miles Yang, Zhao Zhong, and Liefeng Bo. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE-SDE. *arXiv preprint arXiv:2507.21802*, 2025.

[27] Xiaolong Fu, Lichen Ma, Zipeng Guo, Gaojing Zhou, Chongxiao Wang, ShiPing Dong, Shizhe Zhou, Ximan Liu, Jingling Fu, Tan Lit Sin, et al. Dynamic-TreeRPO: Breaking the independent trajectory bottleneck with structured sampling. *arXiv preprint arXiv:2509.23352*, 2025.

[28] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[29] Dinghuai Zhang, Yizhe Zhang, Jiatao Gu, Ruixiang Zhang, Joshua M. Susskind, Navdeep Jaitly, and Shuangfei Zhai. Improving GFlowNets for text-to-image diffusion alignment. *Transactions on Machine Learning Research*, 2025.

[30] Zheng Ding and Weirui Ye. TreeGRPO: Tree-advantage GRPO for online RL post-training of diffusion models. In *ICLR*, 2026.

[31] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-Pic: An open dataset of user preferences for text-to-image generation. In *NeurIPS*, 2023.

[32] Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[34] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

[35] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *ICML*, 2023.

[38] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[39] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *NeurIPS*, 2020.

[40] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[41] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[42] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

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

[44] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS Datasets and Benchmarks Track*, 2023.

[45] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

[46] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[47] Stability AI. Stable Diffusion 3.5 Medium. https://huggingface.co/stabilityai/stable-diffusion-3.5-medium, 2024.

[48] Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

[49] Ted Moskovitz, Aaditya K. Singh, DJ Strouse, Tuomas Sandholm, Ruslan Salakhutdinov, Anca D. Dragan, and Stephen McAleer. Confronting reward model overoptimization with constrained RLHF. In *ICLR*, 2024.

[50] Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023.

Appendix of TMPO: Trajectory Matching Policy Optimization for Diverse and Efficient Diffusion Alignment

A Theoretical Derivations of Softmax-TB **16**

A.1 Boltzmann Target and Partition-Free Softmax Matching 16

A.1.1 Max-Entropy Derivation of the Boltzmann Distribution 16

A.1.2 Elimination of the Partition Function in Tree Topologies 16

A.2 Forward KL Characterization of Softmax-TB (Extended Proof) 17

A.2.1 Formal Statement and Proof 17

A.2.2 Gradient Direction Under Detached Advantage 17

A.2.3 Mode-Covering vs. Mode-Seeking: Formal Characterization 18

A.2.4 Monotonic Descent Under Exact Updates 18

A.2.5 Connection to Within-Group Distribution Matching 19

A.3 Advantage Structure Comparison with Standard Reward Maximization 19

A.3.1 Reward Maximization as Reverse KL Minimization 19

A.3.2 Advantage Structure Comparison 20

A.3.3 Asymptotic Behavior Under Mode Dropping 20

B Importance Sampling, RatioNorm, and Gradient Analysis **20**

B.1 Importance Sampling Ratio Decomposition 20

B.2 Trajectory-Level Gradient Decomposition 21

B.3 Gradient Behavior in Unclipped and Clipped Regions 22

B.4 The Necessity of RatioNorm for Symmetric Clipping 22

C Dynamic Stochastic Tree Sampling **22**

C.1 SDE Noise Injection at Branch Points 22

C.2 Curriculum-Guided Beta Branching Schedule 23

C.2.1 Deterministic Curriculum Trajectory 23

C.2.2 Stochastic Perturbation via Beta Distribution 23

C.2.3 Discrete Mapping and Constraint Enforcement 24

C.3 Trajectory Log-Probability 24

D Further Details on the Experimental Setup **24**

D.1 Quality Metrics 24

D.2 Model and Reward Specification 25

D.3 Training Pipeline 25
D.4 Hyperparameter Specification 26
D.5 Baseline Hyperparameters 26
D.6 Compute Resources Specification 26

E Extended Experimental Results **26**
E.1 FlowRL-style Training Dynamics 26
E.2 OCR Verification with a Different Recognizer 27
E.3 GenEval Category Breakdown 27
E.4 No-Tree Runtime under Matched Execution 28
E.5 Qualitative Comparison with Baselines 28
E.6 Evolution of Evaluation Images Across Training Steps 28

Our Appendix consists of 5 sections. Readers can click on each section number to navigate to the corresponding section:

- Section A and Section B provide mathematical derivations of Softmax-TB and importance sampling analysis.
- Section C details the dynamic stochastic tree sampling algorithm.
- Section D presents details about our experimental setup.
- Section E provides additional experimental results, including FlowRL-style training dynamics, OCR recognizer verification, GenEval category scores, matched-execution runtime, qualitative comparisons with baselines, and evolution of evaluation images across training steps.

A Theoretical Derivations of Softmax-TB

A.1 Boltzmann Target and Partition-Free Softmax Matching

This subsection provides detailed proofs of the two foundational results used in the main text: (i) the Boltzmann distribution as the unique max-entropy target, and (ii) the elimination of the partition function via within-group normalization.

A.1.1 Max-Entropy Derivation of the Boltzmann Distribution

In the TMPO framework, we transform the raw reward R into an exponential form $R \mapsto \exp(\beta R)$. The theoretical justification for this mapping is that it represents the unique solution that maximizes the distributional entropy (exploration diversity) subject to a fixed expected reward constraint.

Proof. Consider the search for an optimal probability distribution $p^*(\tau)$ in trajectory space. Our objective is to maximize the Shannon entropy $H(p)$ while satisfying the mean reward constraint $\mathbb{E}_p[R(\tau)] = \bar{R}$:

$$\begin{aligned} \max_p \quad & H(p) = -\sum_\tau p(\tau) \log p(\tau) \\ \text{s.t.} \quad & \sum_\tau p(\tau) R(\tau) = \bar{R}, \quad \sum_\tau p(\tau) = 1, \end{aligned} \tag{10}$$

where $\bar{R}$ denotes a predefined expected reward level. Introducing the Lagrangian multipliers β (inverse temperature) and λ, we construct the functional:

$$\mathcal{L}(p, \beta, \lambda) = -\sum_\tau p(\tau) \log p(\tau) + \beta \left(\sum_\tau p(\tau) R(\tau) - \bar{R} \right) + \lambda \left(\sum_\tau p(\tau) - 1 \right). \tag{11}$$

Setting the functional derivative with respect to $p(\tau)$ to zero:

$$\frac{\delta \mathcal{L}}{\delta p(\tau)} = -\log p(\tau) - 1 + \beta R(\tau) + \lambda = 0. \tag{12}$$

Solving for $p^*(\tau)$ yields $p^*(\tau) = \exp(\lambda - 1) \exp(\beta R(\tau))$. By enforcing the normalization constraint $\sum_\tau p(\tau) = 1$, we define the partition function $Z_\beta = \sum_{\tau'} \exp(\beta R(\tau'))$, resulting in:

$$p^*(\tau) = \frac{1}{Z_\beta} \exp(\beta R(\tau)). \tag{13}$$

This derivation proves that the Boltzmann distribution is the unique maximum-entropy distribution consistent with a given expected reward level. □

A.1.2 Elimination of the Partition Function in Tree Topologies

In standard GFlowNets [21], the Trajectory Balance (TB) objective [22] is defined as $Z \prod_t P_F = R \prod_t P_B$. In the tree-structured sampling of TMPO, each state possesses a unique parent, which implies that the backward path probability is strictly $P_B(\tau) = 1$. Consequently, the flow matching condition simplifies to $P_\theta(\tau) = R(\tau)/Z_\theta$.

Consider a group of K trajectories sampled from the same branching point x_{split}. Since these trajectories share a common prefix path, the global partition function Z_θ is constant across the group. The relative probability of trajectory τ_i within the group is given by:

$$\frac{P_\theta(\tau_i)}{\sum_{j=1}^K P_\theta(\tau_j)} = \frac{R(\tau_i)/Z_\theta}{\sum_{j=1}^K R(\tau_j)/Z_\theta} = \frac{R(\tau_i)}{\sum_{j=1}^K R(\tau_j)}. \tag{14}$$

Substituting the exponential reward mapping $R \to \exp(\beta R)$ yields the softmax matching format:

$$\frac{P_\theta(\tau_i)}{\sum_{j=1}^K P_\theta(\tau_j)} = \frac{\exp(\beta R_i)}{\sum_{j=1}^K \exp(\beta R_j)}. \tag{15}$$

A.2 Forward KL Characterization of Softmax-TB (Extended Proof)

We now formally characterize the mode-covering property of the Softmax-TB advantage and its connection to forward KL divergence, providing complete proofs and quantitative guarantees that supplement the mode-covering equilibrium result in Section 4.1.

A.2.1 Formal Statement and Proof

Define the within-group Boltzmann target $q_i \triangleq \exp(\beta R_i)/\sum_{j=1}^{K}\exp(\beta R_j)$ and the within-group policy distribution $p_i \triangleq P_\theta(\tau_i)/\sum_{j=1}^{K} P_\theta(\tau_j)$. Both q and p are valid probability distributions over the K trajectories, i.e., $q_i \geq 0$, $p_i > 0$, $\sum_i q_i = \sum_i p_i = 1$.

The Softmax-TB advantage is $A_i = \log q_i - \log p_i$. We now prove:

$$\sum_{i=1}^{K} q_i\, A_i = \sum_{i=1}^{K} q_i(\log q_i - \log p_i) = D_{\mathrm{KL}}(q\|p) \geq 0. \tag{16}$$

Step 1 (Substitution). By definition of the KL divergence:

$$D_{\mathrm{KL}}(q\|p) = \sum_{i=1}^{K} q_i \log \frac{q_i}{p_i} = \sum_{i=1}^{K} q_i(\log q_i - \log p_i) = \sum_{i=1}^{K} q_i A_i. \tag{17}$$

Step 2 (Non-negativity via Gibbs' inequality). By Jensen's inequality applied to the concave function $\log(\cdot)$:

$$\sum_{i=1}^{K} q_i \log \frac{p_i}{q_i} \leq \log \sum_{i=1}^{K} q_i \cdot \frac{p_i}{q_i} = \log \sum_{i=1}^{K} p_i = \log 1 = 0. \tag{18}$$

Since $D_{\mathrm{KL}}(q\|p) = -\sum_i q_i \log(p_i/q_i)$, it follows that $D_{\mathrm{KL}}(q\|p) \geq 0$.

Step 3 (Equality condition). Equality holds in Jensen's inequality if and only if p_i/q_i is constant for all i with $q_i > 0$. Combined with the normalization constraints $\sum_i p_i = \sum_i q_i = 1$, this constant must be 1, i.e., $p_i = q_i$ for all i.

Step 4 (Partition function independence). The within-group normalization is critical: $q_i = \exp(\beta R_i)/\sum_j \exp(\beta R_j)$ and $p_i = P_\theta(\tau_i)/\sum_j P_\theta(\tau_j)$. The global partition function $Z_\beta = \sum_{\text{all }\tau}\exp(\beta R(\tau))$ and the global policy normalization Z_θ both cancel in the ratio. The forward KL is computed *exactly* over the K observed trajectories without requiring access to the full trajectory space.

A.2.2 Gradient Direction Under Detached Advantage

When the advantage is detached ($A_i^{\perp}$), the policy gradient of the Softmax-TB loss takes the form:

$$\nabla_\theta \mathcal{L} = -\frac{1}{K}\sum_{i=1}^{K} A_i^{\perp} \cdot \nabla_\theta \log P_\theta(\tau_i). \tag{19}$$

We show that this gradient direction is sign-consistent with the gradient of the forward KL $D_{\mathrm{KL}}(q\|p_\theta)$.

Exact forward KL gradient. Let $p_i(\theta) = P_\theta(\tau_i)/\sum_j P_\theta(\tau_j)$. By the chain rule:

$$\nabla_\theta D_{\mathrm{KL}}(q\|p_\theta) = -\sum_{i=1}^{K} q_i\, \nabla_\theta \log p_i(\theta). \tag{20}$$

Expanding $\nabla_\theta \log p_i = \nabla_\theta \log P_\theta(\tau_i) - \sum_j p_j \nabla_\theta \log P_\theta(\tau_j)$, we obtain:

$$\nabla_\theta D_{\mathrm{KL}}(q\|p_\theta) = -\sum_{i=1}^{K} (q_i - p_i)\, \nabla_\theta \log P_\theta(\tau_i), \tag{21}$$

where the baseline $\sum_j p_j \nabla_\theta \log P_\theta(\tau_j)$ has been subtracted. Comparing the two gradients, TMPO uses $A_i/K = \log(q_i/p_i)/K$ as the per-trajectory weight, while the exact forward KL uses $(q_i - p_i)$. Although the magnitudes differ, the signs are strictly identical: $\text{sign}(A_i) = \text{sign}(\log(q_i/p_i)) = \text{sign}(q_i - p_i)$. This means:

- $A_i > 0$ ($q_i > p_i$): trajectory τ_i is under-covered $\Rightarrow$ both gradients increase $P_\theta(\tau_i)$.
- $A_i < 0$ ($q_i < p_i$): trajectory τ_i is over-represented $\Rightarrow$ both gradients decrease $P_\theta(\tau_i)$.

The detached gradient omits the baseline term, introducing magnitude bias but preserving the sign of each trajectory's gradient contribution. This is analogous to REINFORCE without baseline: the per-trajectory gradient direction remains correct, while the baseline would reduce variance at the cost of cross-trajectory coupling. In finite groups, the omitted baseline $\sum_j p_j \nabla_\theta \log P_\theta(\tau_j)$ does not vanish exactly, so the estimator is biased in magnitude but sign-consistent with the exact forward KL gradient.

Role of the IS ratio. In the full TMPO loss, the gradient flows through the bias-corrected IS ratio $\hat{\rho}_i = P_\theta(\tau_i)/P_{\text{old}}(\tau_i)$ rather than directly through $\log P_\theta(\tau_i)$. The IS ratio serves as the gradient carrier for off-policy correction; it is *not* a substitute for the Boltzmann weight q_i that appears in the exact forward KL gradient. These two quantities are functionally independent: $\hat{\rho}_i$ corrects for the distribution shift between successive policy updates, while $A_i^\perp$ provides the directional signal for distribution matching.

A.2.3 Mode-Covering vs. Mode-Seeking: Formal Characterization

The forward and reverse KL divergences impose fundamentally different penalties on the mismatch between q and p. We formalize this distinction below.

Forward KL $D_{\text{KL}}(q\|p) = \sum_i q_i \log(q_i/p_i)$: For any trajectory i with $q_i > 0$, the penalty $q_i \log(q_i/p_i) \to +\infty$ as $p_i \to 0$. This infinite cost forces p to assign non-zero probability mass everywhere that q is positive, producing *mode-covering* behavior. Formally, the minimizer $p^* = \arg\min_p D_{\text{KL}}(q\|p)$ satisfies $\text{supp}(p^*) \supseteq \text{supp}(q)$.

Reverse KL $D_{\text{KL}}(p\|q) = \sum_i p_i \log(p_i/q_i)$: When $p_i = 0$, the contribution is exactly 0 regardless of q_i. The policy can collapse to a single mode without penalty, producing *mode-seeking* behavior. The minimizer concentrates on high-q regions: $p^* = \arg\min_p D_{\text{KL}}(p\|q)$ typically yields $\text{supp}(p^*) \subseteq \text{supp}(q)$.

Quantitative bound via Pinsker's inequality. The mode-covering guarantee can be strengthened by Pinsker's inequality, which relates the KL divergence to the total variation distance:

$$\text{TV}(q,p) \triangleq \frac{1}{2}\sum_{i=1}^{K} |q_i - p_i| \leq \sqrt{\frac{1}{2} D_{\text{KL}}(q\|p)}. \tag{22}$$

Therefore, when Softmax-TB drives $D_{\text{KL}}(q\|p) \to 0$, the policy distribution p converges to q in total variation, guaranteeing that no mode is under-covered by more than $\sqrt{D_{\text{KL}}(q\|p)/2}$ in absolute probability.

Implication for Softmax-TB. Since the Boltzmann target q assigns strictly positive probability to all K trajectories ($\exp(\beta R_i) > 0$ for any finite R_i), minimizing $D_{\text{KL}}(q\|p)$ requires $p_i > 0$ for all i. Moreover, by Pinsker's inequality, bounding the forward KL below δ ensures $|p_i - q_i| \leq \sqrt{\delta/2}$ for every trajectory. This provides a *quantitative* diversity guarantee that standard GRPO-style reward maximization (reverse KL) cannot offer.

A.2.4 Monotonic Descent Under Exact Updates

The Softmax-TB loss and the within-group forward KL share the same unique fixed point $p_i = q_i$ for all i. Under idealized exact updates, we establish that the forward KL is monotonically non-increasing.

Claim. Let $\theta^{(n)}$ denote the parameters after iteration n. If the update rule is $\theta^{(n+1)} = \arg\min_\theta D_{\text{KL}}(q^{(n)}\|p_\theta)$ where $q^{(n)}$ is the Boltzmann target computed from the group sampled at iteration n, then:

$$D_{\text{KL}}(q^{(n)}\|p_{\theta^{(n+1)}}) \leq D_{\text{KL}}(q^{(n)}\|p_{\theta^{(n)}}). \tag{23}$$

Proof. By definition, $\theta^{(n+1)}$ minimizes $D_{\mathrm{KL}}(q^{(n)} \| p_\theta)$ over θ. Since $\theta^{(n)}$ is a feasible point, we have $D_{\mathrm{KL}}(q^{(n)} \| p_{\theta^{(n+1)}}) \leq D_{\mathrm{KL}}(q^{(n)} \| p_{\theta^{(n)}})$. We note that this single-step guarantee is tautological under the exact-minimization assumption; it serves to establish the fixed-point structure, not to claim practical convergence rates. In practice, TMPO performs gradient steps rather than exact minimization, and the trust-region clipping ensures that each step remains close to $\theta^{(n)}$, bounding the gap between the approximate and exact updates. □

A.2.5 Connection to Within-Group Distribution Matching

A natural concern is that Softmax-TB samples from p_{old} rather than q_β, seemingly preventing the computation of the log-ratio advantage $A_i = \log(q_i/p_i)$. We clarify that this is a question of *computational feasibility*, not of the loss identity.

In each training iteration, TMPO generates a fixed group of K trajectories $\{\tau_1, \ldots, \tau_K\}$ from the current tree sampler. Within this group, both q and p are discrete distributions over the *same* K atoms. The advantage $A_i = \log(q_i/p_i)$ is computed exactly over these K fully observed atoms; no sampling from q is required. This is analogous to supervised cross-entropy loss: given a fixed dataset of K labeled examples, the cross-entropy $H(q, p) = -\sum_i q_i \log p_i$ can be evaluated without sampling from the label distribution q, because all K labels are observed.

Formally, define the within-group forward KL at iteration n as:

$$\mathcal{F}_n(\theta) \triangleq D_{\mathrm{KL}}(q^{(n)} \| p_\theta^{(n)}) = \sum_{i=1}^{K} q_i^{(n)} \log \frac{q_i^{(n)}}{p_i(\theta)}, \tag{24}$$

where $q^{(n)}$ and $p^{(n)}$ are both normalized over the K trajectories sampled at iteration n. Note that any group-based method (including GRPO) could in principle compute this quantity; the distinction is that TMPO's advantage $A_i = \log(q_i/p_i)$ *uses* this log-ratio as its optimization signal, whereas reward-maximization methods discard the policy probability p_i entirely. Within-group normalization converts the intractable global distribution matching problem into a tractable local one, ensuring that the distribution-aware advantage can be computed exactly without access to the global Boltzmann distribution q_β. Each group covers a finite subset of the trajectory space; stochastic tree exploration (SDE noise injection and the dynamic Beta branching schedule; Section C) diversifies successive groups so that the cumulative effect of exact within-group matching progressively extends across the trajectory manifold.

A.3 Advantage Structure Comparison with Standard Reward Maximization

Standard policy gradient methods, including TMPO, sample trajectories from the current policy π_θ, placing them in a reverse KL sampling regime. The distinction lies not in the sampling distribution but in the advantage structure. We formalize the connection between reward maximization and reverse KL, then contrast the resulting advantage with the distribution-aware advantage of Softmax-TB.

A.3.1 Reward Maximization as Reverse KL Minimization

Consider the standard RL objective of maximizing expected reward:

$$\max_\theta \ \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]. \tag{25}$$

Let $q(\tau) = \exp(R(\tau))/Z$ denote the Boltzmann target, so that $R(\tau) = \log q(\tau) + \log Z$. Substituting:

$$\mathbb{E}_{\pi_\theta}[R(\tau)] = \mathbb{E}_{\pi_\theta}[\log q(\tau)] + \log Z = -H(\pi_\theta, q) + \log Z, \tag{26}$$

where $H(\pi_\theta, q) = -\sum_\tau \pi_\theta(\tau) \log q(\tau)$ is the cross-entropy. Using $D_{\mathrm{KL}}(\pi_\theta \| q) = H(\pi_\theta, q) - H(\pi_\theta)$:

$$\boxed{\mathbb{E}_{\pi_\theta}[R(\tau)] = -D_{\mathrm{KL}}(\pi_\theta \| q) - H(\pi_\theta) + \log Z.} \tag{27}$$

Since $\log Z$ is a constant, maximizing expected reward is equivalent to minimizing $D_{\mathrm{KL}}(\pi_\theta \| q) + H(\pi_\theta)$. This reveals a compounding mode-collapse pressure: reward maximization not only minimizes the reverse KL (which is mode-seeking) but also minimizes the policy entropy, further concentrating probability mass. Ignoring the entropy contribution: $\max_\theta \mathbb{E}[R] \equiv \min_\theta D_{\mathrm{KL}}(\pi_\theta \| q)$.

A.3.2 Advantage Structure Comparison

The fundamental distinction between TMPO and standard reward maximization lies in the advantage structure:

Reward-maximization advantage (z-scored reward, e.g. Flow-GRPO [6]):

$$A_i^{\mathrm{GRPO}} = \frac{R_i - \mu_R}{\sigma_R}, \tag{28}$$

where μ_R and σ_R are the within-group mean and standard deviation of the rewards. This is a *linear function of* R_i that is entirely agnostic to the policy probability $p_i = P_\theta(\tau_i)/\sum_j P_\theta(\tau_j)$.

Softmax-TB advantage (log-ratio):

$$A_i^{\mathrm{TMPO}} = \log q_i - \log p_i = \log \frac{q_i}{p_i}. \tag{29}$$

This is *distribution-aware*: it simultaneously encodes both the Boltzmann target weight q_i and the current policy probability p_i.

A.3.3 Asymptotic Behavior Under Mode Dropping

Consider the scenario where the policy drops a mode, i.e., $p_i \to 0$ for some trajectory i with $q_i > 0$:

Reward maximization: The advantage $A_i^{\mathrm{GRPO}} = (R_i - \mu_R)/\sigma_R$ depends only on R_i and the group statistics. As $p_i \to 0$, the advantage does not change; the gradient signal for trajectory i is unaffected by the policy's diminishing probability on it. Formally, the reverse KL contribution $p_i \log(p_i/q_i) \to 0$ as $p_i \to 0$, confirming that mode-dropping incurs zero penalty.

TMPO: The advantage $A_i^{\mathrm{TMPO}} = \log(q_i/p_i) \to +\infty$ as $p_i \to 0$. This divergent correction signal forces the policy to restore probability mass on the dropped mode. This behavior directly mirrors the forward KL: $q_i \log(q_i/p_i) \to +\infty$ as $p_i \to 0$. We summarize the structural comparison between the two advantage functions in Table 5.

Table 5: Structural comparison of advantage functions and their KL divergence properties.

	Reward Maximization	**TMPO (Softmax-TB)**
Advantage	$A_i \propto R_i - \bar{R}$ (linear in reward)	$A_i = \log(q_i/p_i)$ (log-ratio)
Aware of p_i?	No	Yes
$p_i \to 0$ behavior	A_i unchanged	$A_i \to +\infty$
KL regime	Reverse $D_{\mathrm{KL}}(p\|q)$	Inherits forward $D_{\mathrm{KL}}(q\|p)$ asymptote
Equilibrium	$\pi_\theta \to \arg\max R$	$p_i = q_i \ \forall i$
Mode-covering guarantee	Requires external regularization	Built into advantage structure

Since the reverse KL assigns zero penalty to dropped modes ($p_i \log(p_i/q_i) \to 0$), reward-maximization methods must rely on external mechanisms—KL penalties against a reference policy, or heuristic advantage reweighting—to counteract mode collapse. These mechanisms are either agnostic to the reward landscape (KL penalty) or unable to detect modes that have already been dropped (reweighting). The forward KL asymptote built into TMPO's advantage provides this guarantee structurally: any mode with $p_i < q_i$ receives a corrective signal proportional to $\log(q_i/p_i)$, without auxiliary losses or hyperparameters.

B Importance Sampling, RatioNorm, and Gradient Analysis

This section provides full derivations of the importance sampling decomposition and bias correction used in the Softmax-TB objective (Equation (6)), and analyzes the resulting trust-region properties.

B.1 Importance Sampling Ratio Decomposition

To enable multiple policy updates per batch, TMPO requires importance sampling (IS) correction. In the tree-structured sampling, the path log-probability decomposes into T per-step transition

probabilities at the stochastic branch points: $\log P_\theta(\tau_i) = \sum_{t=1}^{T} \log \pi_\theta(x_{t-1}^{(i)} \mid x_t^{(i)})$. Consequently, the trajectory-level log IS ratio admits a per-step decomposition:

$$\log \rho_i = \log P_\theta(\tau_i) - \log P_{\text{old}}(\tau_i) = \sum_{t=1}^{T} \underbrace{[\log \pi_\theta(x_{t-1} \mid x_t) - \log \pi_{\text{old}}(x_{t-1} \mid x_t)]}_{\triangleq \log w_{i,t}} \tag{30}$$

Under the Gaussian transition kernel $\pi_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1};\, \mu_\theta(x_t),\, \sigma_t^2 \Delta t\, \mathbf{I})$, each per-step log-ratio decomposes as:

$$\log w_{i,t} = \frac{\Delta\mu_\theta \cdot \epsilon}{\sigma_t \sqrt{\Delta t}} - \frac{\|\Delta\mu_\theta\|^2}{2\sigma_t^2 \Delta t} \tag{31}$$

where $\Delta\mu_\theta \triangleq \mu_\theta(x_t) - \mu_{\text{old}}(x_t)$ is the mean shift and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is the sampled noise. The second term introduces a deterministic negative shift: $\mathbb{E}[\log w_{i,t}] = -\|\Delta\mu_\theta\|^2/(2\sigma_t^2\Delta t) < 0$. Note that the IS ratio itself is unbiased in ratio space ($\mathbb{E}[w_{i,t}] = 1$ by construction), and the negative log-expectation is a natural consequence of Jensen's inequality ($\mathbb{E}[\log w] < \log \mathbb{E}[w] = 0$). However, this deterministic negative shift in log-space causes the clipping interval $[1{-}\varepsilon, 1{+}\varepsilon]$ to be asymmetrically effective, suppressing gradient signals from high-reward trajectories.

Log-space centering (RatioNorm). Following GRPO-Guard [19], TMPO removes the deterministic negative shift to center the log-ratio distribution around zero:

$$\log \hat{w}_{i,t} = \log w_{i,t} + \frac{\|\Delta\mu_\theta\|^2}{2\sigma_t^2 \Delta t} = \frac{\Delta\mu_\theta \cdot \epsilon}{\sigma_t \sqrt{\Delta t}} \tag{32}$$

The resulting centered log-ratio has zero mean ($\mathbb{E}[\log \hat{w}_{i,t}] = 0$ since $\mathbb{E}[\epsilon] = 0$), restoring symmetric clipping behavior. Note that this centering implies $\mathbb{E}[\hat{w}_{i,t}] > 1$ by Jensen's inequality; the operation trades unbiasedness in ratio space for symmetry in log-space, which is the relevant domain for the clipping operator. Unlike the full GRPO-Guard formulation which additionally rescales by $\sigma_t\sqrt{\Delta t}$ for variance normalization, TMPO retains the unscaled form to preserve gradient magnitude in the few-step ($T \leq 5$) SDE regime. During backpropagation, the bias correction term is treated as a detached constant while $\log w_{i,t}$ retains its gradient.

Trajectory-level aggregation. The trajectory-level log-ratio is constructed as $\log \hat{w}_i = \sum_{t=1}^{T} \log \hat{w}_{i,t}$. In TMPO's tree sampler, $T \leq 5$ stochastic steps are used, so the sum remains $O(1)$ in practice and does not saturate the clipping interval $[1{-}\varepsilon, 1{+}\varepsilon]$. This direct summation preserves the full per-step gradient signal, which is important in the few-step SDE regime where time-averaging ($1/T$) would attenuate the gradient contribution of each stochastic transition.

B.2 Trajectory-Level Gradient Decomposition

The complete Softmax-TB loss takes the form:

$$\mathcal{L}_{\text{TB}}(\theta) = -\frac{1}{K} \sum_{i=1}^{K} \min\big(\hat{w}_i(\theta)\, A_i^{\perp},\ \text{clip}(\hat{w}_i(\theta),\, 1{-}\varepsilon,\, 1{+}\varepsilon)\, A_i^{\perp}\big) \tag{33}$$

where $\hat{w}_i(\theta) = \exp\Big(\sum_{t=1}^{T} \log \hat{w}_{i,t}(\theta)\Big)$ is the bias-corrected importance ratio that depends on θ, and $A_i^{\perp}$ is the detached Softmax-TB advantage treated as a constant with respect to θ. The $\min$ selects whichever term yields the lower (more pessimistic) objective. Since no gradient flows through $A_i^{\perp}$, the gradient of the active branch admits a single-term decomposition:

$$\nabla_\theta \mathcal{L}_{\text{TB}}(\theta) = -\frac{1}{K} \sum_{i=1}^{K} \nabla_\theta \hat{w}_i^{\text{active}}(\theta) \cdot A_i^{\perp} \tag{34}$$

where $\hat{w}_i^{\text{active}}$ denotes whichever of $\hat{w}_i$ or $\text{clip}(\hat{w}_i)$ is selected by the $\min$ operator. The detached advantage $A_i^{\perp}$ provides the directional signal, while the importance ratio is the sole carrier of gradient information. Crucially, since the bias correction terms are constants with respect to θ, we have $\nabla_\theta \log \hat{w}_i = \sum_t \nabla_\theta \log \pi_\theta(x_{t-1} \mid x_t)$, which depends only on the log-probability of trajectory τ_i. This ensures that each trajectory's gradient contribution is independent of all other trajectories within the group.

B.3 Gradient Behavior in Unclipped and Clipped Regions

The trust-region property arises from how the clipping operator modulates the gradient based on the magnitude of the importance ratio.

Case 1: Within trust region ($\hat{w}_i \in [1-\varepsilon, 1+\varepsilon]$).
Both branches of the $\min$ coincide and the gradient becomes:

$$\nabla_\theta \mathcal{L}_{\text{TB}}(\theta)\Big|_{\text{unclipped}} = -\frac{1}{K}\sum_{i=1}^{K} \hat{w}_i(\theta) \cdot \nabla_\theta \log \hat{w}_i(\theta) \cdot A_i^\perp \tag{35}$$

In this region, the off-policy update proceeds normally. A positive advantage $A_i^\perp > 0$ drives the gradient to increase the path probability of trajectory τ_i, while $A_i^\perp < 0$ decreases it.

Case 2: Beneficial over-update ($A_i^\perp > 0$, $\hat{w}_i > 1+\varepsilon$ or $A_i^\perp < 0$, $\hat{w}_i < 1-\varepsilon$).
The policy has moved in the direction indicated by the advantage beyond the trust region. In this case the clipped term yields a *lower* objective value, so $\min$ selects it. Since the clipped term has zero gradient with respect to θ, the trajectory contributes no update, preventing further over-optimization.

Case 3: Harmful deviation ($A_i^\perp > 0$, $\hat{w}_i < 1-\varepsilon$ or $A_i^\perp < 0$, $\hat{w}_i > 1+\varepsilon$).
The policy has drifted in the *opposite* direction to the advantage. The unclipped term now yields the lower objective, so $\min$ selects it and the gradient flows through the unclipped ratio $\hat{w}_i$, providing a corrective signal that pushes the policy back. This asymmetric behavior—blocking beneficial over-updates while allowing corrective ones—is the key difference from a naive $\text{clip}(\hat{w}_i) \cdot A_i^\perp$ formulation and is essential for robust trust-region control.

B.4 The Necessity of RatioNorm for Symmetric Clipping

Applying the clipping bounds $[1-\varepsilon, 1+\varepsilon]$ directly to the raw importance ratio $\rho_i = P_\theta(\tau_i)/P_{\text{old}}(\tau_i)$ is ineffective due to the systematic bias inherent in the per-step log-ratios. As derived in the IS Ratio Decomposition (Section B), the raw log-ratio contains a negative bias:

$$\mathbb{E}[\log w_{i,t}] = -\frac{\|\Delta\mu_\theta\|^2}{2\sigma_t^2 \Delta t} < 0 \tag{36}$$

This causes $\mathbb{E}[\log \rho_i] < 0$, so the distribution of ρ_i concentrates below 1: the majority of trajectories fall below the lower clipping bound $1-\varepsilon$, while the upper bound $1+\varepsilon$ is rarely reached. The resulting asymmetry undermines the trust-region mechanism: the clipping cannot prevent excessively large updates in the direction of decreasing trajectory probability.

Bias correction restores symmetry by removing the negative expectation from each per-step log-ratio:

$$\log \hat{w}_{i,t} = \log w_{i,t} + \frac{\|\Delta\mu_\theta\|^2}{2\sigma_t^2 \Delta t} = \frac{\Delta\mu_\theta \cdot \epsilon}{\sigma_t \sqrt{\Delta t}} \tag{37}$$

The bias correction ensures $\mathbb{E}[\log \hat{w}_{i,t}] = 0$, centering the trajectory-level ratio $\hat{w}_i = \exp(\sum_t \log \hat{w}_{i,t})$ around 1 and enabling the symmetric clipping interval $[1-\varepsilon, 1+\varepsilon]$ to function as intended.

C Dynamic Stochastic Tree Sampling

This section provides detailed derivations for the stochastic tree sampler described in Section 4.2, including the SDE noise injection, the curriculum-guided Beta branching schedule, and the trajectory log-probability computation.

C.1 SDE Noise Injection at Branch Points

The denoising process in flow-matching models follows the ODE $\mathrm{d}x = v_\theta(x, t)\,\mathrm{d}t$, where v_θ is the learned velocity field and t decreases from 1 (pure noise) to 0 (clean image). To introduce stochasticity at branch points, TMPO converts this ODE into an equivalent SDE by injecting Gaussian noise. At branch point s_i with noise schedule value σ_{s_i}, the SDE transition is:

$$x_{s_i^-} = \mu_\theta(x_{s_i}, s_i) + \gamma_i\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{38}$$

where μ_θ is the deterministic drift (incorporating both the velocity field and SDE drift correction), and γ_i is the noise magnitude:

$$\gamma_i = \eta_i \sqrt{\frac{\sigma_{s_i}}{1-\sigma_{s_i}}} \cdot \sqrt{-\Delta t_{s_i}}. \tag{39}$$

Here $\eta_i \in (0, 1]$ is a per-layer noise coefficient, $\sigma_{s_i}/(1-\sigma_{s_i})$ is the inverse signal-to-noise ratio at timestep s_i, and $\Delta t_{s_i} = \sigma_{s_i^-} - \sigma_{s_i} < 0$ is the (negative) time step. The $\sqrt{\text{SNR}^{-1}}$ factor ensures that noise magnitude scales naturally with the diffusion schedule: more noise is injected at early high-noise steps, while late low-noise steps receive proportionally less perturbation.

The SDE drift μ_θ includes a correction term to preserve the marginal distribution:

$$\mu_\theta = x_{s_i}\left(1 + \frac{\gamma_i^2}{2\sigma_{s_i}}\Delta t\right) + v_\theta(x_{s_i}, s_i)\left(1 + \frac{\gamma_i^2(1-\sigma_{s_i})}{2\sigma_{s_i}}\right)\Delta t, \tag{40}$$

where the additional γ_i^2 terms arise from the Itô-to-Stratonovich conversion and ensure that the SDE marginals match those of the original ODE.

At each branch point, B independent noise realizations $\{\varepsilon_b\}_{b=1}^B$ are drawn, producing B child branches from the same parent state. With T branch points and branching factor B, the tree produces $K = B^T$ terminal trajectories per prompt (e.g., $B = 3, T = 3 \Rightarrow K = 27$).

C.2 Curriculum-Guided Beta Branching Schedule

Branch positions determine where along the denoising trajectory the tree bifurcates. Early branching (high σ) creates globally diverse structures, while late branching (low σ) produces fine-grained variations that share semantic structure. TMPO uses a curriculum scheduler that gradually shifts branch positions from early to late as training progresses.

C.2.1 Deterministic Curriculum Trajectory

For each branch point $i \in \{1, 2, 3\}$, the curriculum defines early positions e_i and late positions l_i (hyperparameters). Given normalized training progress $p = \min(u/U, 1)$ where u is the current step and U the total steps, the deterministic curriculum mean is:

$$\mu_i(p) = e_i + (l_i - e_i) \cdot p. \tag{41}$$

This linear interpolation shifts branch positions from broad early exploration to fine-grained late refinement.

C.2.2 Stochastic Perturbation via Beta Distribution

To prevent the policy from overfitting to a fixed tree geometry, each branch position is stochastically perturbed around the curriculum mean. We normalize the curriculum mean to the interval $(0, 1)$:

$$\bar{\mu}_i(p) = \frac{\mu_i(p) - s_{\min}}{s_{\max} - s_{\min}}, \quad \bar{\mu}_i \in (0, 1), \tag{42}$$

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum allowed step indices. The branch position is then sampled from a Beta distribution:

$$\xi_i \sim \text{Beta}(\bar{\mu}_i\kappa,\ (1-\bar{\mu}_i)\kappa)\,, \tag{43}$$

where $\kappa > 0$ is the concentration parameter controlling the spread of the perturbation.

Properties of this parameterization. The Beta distribution $\text{Beta}(\alpha, \beta)$ with $\alpha = \bar{\mu}\kappa$ and $\beta = (1-\bar{\mu})\kappa$ has:

$$\mathbb{E}[\xi_i] = \frac{\alpha}{\alpha+\beta} = \bar{\mu}_i, \tag{44}$$

$$\text{Var}[\xi_i] = \frac{\bar{\mu}_i(1-\bar{\mu}_i)}{\kappa+1}. \tag{45}$$

The mean exactly equals the curriculum trajectory, and the variance is inversely proportional to $\kappa + 1$. As $\kappa \to \infty$, the distribution concentrates at $\bar{\mu}_i$ (deterministic curriculum); as $\kappa \to 0$, its probability mass concentrates near the endpoints 0 and 1. In practice, $\kappa \in [3, 8]$ provides sufficient stochasticity to avoid geometry overfitting while keeping branch positions close to the curriculum.

C.2.3 Discrete Mapping and Constraint Enforcement

The continuous sample ξ_i is mapped back to a discrete step index:

$$\tilde{s}_i = \lfloor s_{\min} + (s_{\max} - s_{\min}) \cdot \xi_i + 0.5 \rfloor . \tag{46}$$

The final branch indices $\{s_1, s_2, s_3\}$ are obtained by sorting $\{\tilde{s}_1, \tilde{s}_2, \tilde{s}_3\}$.

C.3 Trajectory Log-Probability

Between branch points, the denoising trajectory follows deterministic ODE integration. In the policy objective, trajectory log-probabilities are computed as the sum of the conditional log-densities of the T stochastic SDE transitions; deterministic steps determine their conditioning states and are not assigned a separate stochastic transition log-density:

$$\log P_\theta(\tau) = \sum_{i=1}^{T} \log \pi_\theta(x_{s_i^-} \mid x_{s_i}), \tag{47}$$

where each transition probability follows the Gaussian:

$$\log \pi_\theta(x_{s_i^-} \mid x_{s_i}) = -\frac{\|x_{s_i^-} - \mu_\theta(x_{s_i}, s_i)\|^2}{2\gamma_i^2} - \frac{d}{2}\log(2\pi\gamma_i^2), \tag{48}$$

with d being the latent dimension. In practice, this is computed as a per-dimension mean to maintain $O(1)$ magnitude regardless of latent resolution, which is essential for numerical stability in mixed-precision (bf16) training:

$$\overline{\log \pi}_\theta(x_{s_i^-} \mid x_{s_i}) \triangleq \frac{1}{d}\log \pi_\theta(x_{s_i^-} \mid x_{s_i}). \tag{49}$$

This per-dimension normalization is used consistently for current- and old-policy log-probabilities. It is a numerical scaling convention, not an algebraic cancellation: applying $1/d$ before Softmax changes the relative trajectory weights, and exponentiating a scaled current–old log-probability difference changes the resulting ratio. Accordingly, reward temperature and clipping settings must be interpreted under the same per-dimension convention; they are not automatically equivalent to those based on unscaled joint log-densities. Note that γ_i is a constant determined by the noise schedule and does not depend on θ; the gradient of $\log P_\theta(\tau)$ with respect to θ flows exclusively through the drift term μ_θ.

D Further Details on the Experimental Setup

D.1 Quality Metrics

We adopt the following evaluation metrics:

- **HPS-v2.1** [11]: A human preference score trained on large-scale pairwise human annotations, computed as the cosine similarity between image and text embeddings from a ViT-H/14 backbone.
- **ImageReward** [10]: A reward model trained on human preference annotations over text-to-image generations, providing a scalar score that correlates with human aesthetic and fidelity judgments.
- **PickScore** [31]: A CLIP-based [46] preference model trained on the Pick-a-Pic dataset. We adopt the flow-scaled scoring mode during training.
- **GenEval** [44]: A compositional generation benchmark that evaluates whether generated images faithfully reflect the objects, attributes, and spatial relations specified in the prompt. Each image is scored by a Mask2Former object detector combined with CLIP attribute matching, served via an HTTP endpoint. During training, we use strict scoring (binary 1 if all criteria are satisfied, 0 otherwise); during evaluation, we report the fraction of compositional criteria satisfied.
- **OCR Accuracy ($1-$NED)**: For the text-rendering setting, we measure character-level accuracy as $1 - \text{NED}$, where NED denotes the normalized edit distance between the recognized text and the ground-truth target string.

- **LGMD (Log Geometric Mean Distance)**: A latent-space diversity metric defined in the main text (Section 5). LGMD computes the logarithm of the dimension-normalized geometric mean of pairwise Euclidean distances between flattened VAE latent features. Because the geometric mean is dominated by the smallest pairwise distances, LGMD is highly sensitive to near-duplicate samples; positive values indicate healthy diversity, while negative values signal mode collapse.
- **Cosine Diversity (Cos. Div.)**: A semantic diversity metric following GARDO [15]. It computes the mean pairwise cosine distance in DINOv2 [41] ViT-L/14 feature space: $\text{Cos. Div.} = \frac{2}{N(N-1)} \sum_{i<j} (1 - \cos(\psi(x_i), \psi(x_j)))$. While LGMD captures low-level structural duplicates, Cos. Div. captures semantic layout and texture differences.

D.2 Model and Reward Specification

Table 6 lists the backbone model and reward models used in our experiments.

Table 6: Models used in our experiments and their sources.

Model	Link
FLUX.1-dev (backbone)	https://huggingface.co/black-forest-labs/FLUX.1-dev
HPS-v2.1 [11]	https://github.com/tgxs002/HPSv2
ImageReward [10]	https://huggingface.co/THUDM/ImageReward
PickScore [31]	https://huggingface.co/yuvalkirstain/PickScore_v1

Backbone. We use FLUX.1-dev, a guidance-distilled rectified flow transformer [42]. LoRA adapters are applied with rank r=64 and scaling factor α=128 (α/r=2) to all linear projections in each transformer block, including: (i) attention Q/K/V and output projections for both the image stream and the context (text-conditioned) stream, and (ii) the GEGLU gate and down projections in both the image and context feed-forward networks. All generation uses a classifier-free guidance [43] scale of 3.5 and a resolution of 512×512. Training rollouts use 6 denoising steps for efficiency; evaluation uses 28 steps for full-quality generation.

Reward normalization. In the joint preference setting, HPS-v2.1, ImageReward, and PickScore are combined at equal weight. Before summation, each reward is independently z-score normalized within the K=27 trajectory group: $\tilde{R}_m = (R_m - \mu_m)/(\sigma_m + \epsilon)$, where μ_m and σ_m are the within-group mean and standard deviation of reward m, and $\epsilon = 10^{-8}$ prevents division by zero. This normalization ensures that rewards with different scales contribute equally to the Softmax-TB advantage.

D.3 Training Pipeline

Each TMPO training iteration proceeds as follows:

1. **Prompt sampling.** Eight prompts are drawn uniformly from the training set, one per GPU.
2. **Tree rollout.** On each GPU, the stochastic tree sampler generates K=27 terminal trajectories for its prompt via T=3 branch points with B=3 children each, yielding $8 \times 27 = 216$ images per iteration. Between branch points, deterministic ODE integration advances the latent. When the first branch point falls at step 1, the B child trajectories are initialized from independent random seeds; the remaining two branch points inject CPS-type SDE noise [48] with coefficient η=0.7. Branch positions are determined by the curriculum-guided Beta schedule (Appendix C.2).
3. **Reward evaluation.** Each terminal image $x_0^{(i)}$ is decoded and scored by the reward model(s). Rewards are z-score normalized in the joint setting.
4. **Advantage computation.** The detached Softmax-TB advantage $A_i^{\perp}$ is computed per Eq. (7).
5. **IS ratio recomputation.** The current policy π_θ recomputes per-step log-probabilities at each branch point, yielding the bias-corrected IS ratio $\hat{\rho}_i$ per Eq. (32).
6. **Policy update.** The PPO-style clipped loss $\mathcal{L}_{\text{TMPO}}$ (Eq. (6)) is minimized via AdamW without an auxiliary reference-policy KL penalty. The old policy π_{old} is the frozen policy from the latest rollout; its log-probabilities are stored before the gradient update and remain fixed throughout the

IS updates. EMA-smoothed parameters (decay 0.9, update interval 8) are maintained separately and used only for evaluation and checkpoint saving.

D.4 Hyperparameter Specification

Except for task-specific adjustments noted below, TMPO hyperparameters are fixed across all training protocols. We use AdamW (β_1=0.9, β_2=0.999) with weight decay 10^{-4} and a cosine annealing schedule ($\eta_{\min}$=0.1 $\times$ lr). The learning rate is 3×10^{-5} for GenEval, OCR, and Joint Preference, and 5×10^{-5} for PickScore. We train for 1,000 steps (GenEval), 250 steps (OCR), 500 steps (PickScore), and 500 steps (Joint). Gradient norms are clipped to 0.5 except for PickScore (0.3). Training uses bf16 mixed precision throughout. Each prompt produces K=27 terminal trajectories via a three-level prefix-sharing tree with B=3 children per branch point (T=3 branch points). The training denoising horizon is 6 steps and the evaluation horizon is 28 steps. IS ratios are clipped with ε=0.2. The reward temperature β warms up linearly from 0.8 to 2.0 over 150 steps (100 for OCR). EMA-smoothed parameters (decay 0.9, update interval 8) are maintained for evaluation and checkpoint saving. The tree sampler uses CPS-type SDE noise injection [48] with noise coefficient η=0.7. Branch positions follow the curriculum-guided Beta schedule (Appendix C.2) with early positions $(1, 2, 3)$, late positions $(1, 3, 5)$, and concentration κ=6.0. LoRA is applied with rank r=64 and α=128.

D.5 Baseline Hyperparameters

For fair comparison, all baselines use the same backbone (FLUX.1-dev with LoRA r=64, α=128), the same reward models, and the same prompt set. Wherever possible, we adopt the default settings from the original papers [6, 15, 26, 30]. All baselines use K=27 trajectories per prompt to match TMPO's group size. Flow-GRPO, MixGRPO, TreeGRPO, and GARDO share a learning rate of 5×10^{-5}, gradient norm clipping of 0.3, and IS clipping ε=0.2. The KL coefficient is β_{KL}=0.03 for Flow-GRPO, MixGRPO, and TreeGRPO, and 0.04 for GARDO. All GRPO-based methods use z-score advantage normalization (TreeGRPO normalizes within the tree group). Flow-GRPO and GARDO sample K independent trajectories per prompt; MixGRPO uses mixed ODE-SDE sampling; TreeGRPO uses prefix-sharing tree rollouts. GARDO additionally employs DINOv2-based advantage reweighting for diversity preservation.

D.6 Compute Resources Specification

All experiments are conducted on 8 GPUs using HuggingFace `accelerate` for distributed training with FSDP. Training uses bf16 mixed precision with TF32-enabled cuDNN kernels. Iteration times reported in the main text are wall-clock measurements averaged over the full training run. A single TMPO training run takes approximately 9.5 wall-clock hours (76 GPU-hours) on the PickScore only protocol (500 steps), 5.3 wall-clock hours (42 GPU-hours) on the OCR only protocol (250 steps), and 25.5 wall-clock hours (204 GPU-hours) on the GenEval only protocol (1,000 steps).

E Extended Experimental Results

We report FlowRL-style training dynamics, OCR recognizer robustness, GenEval category scores, and matched-execution runtime, followed by qualitative comparisons and training snapshots.

E.1 FlowRL-style Training Dynamics

FlowRL and within-group matching. FlowRL [25] and TMPO share a Boltzmann distribution-matching motivation, but FlowRL learns an absolute prompt-conditioned flow while TMPO matches relative reward and trajectory-probability shares within the same realized group. Reward normalization changes the effective target scale across sampled groups. Fitting a separate normalizer couples this estimation to policy learning, whereas relative matching cancels the shared factor. This is analogous to using a relative group signal instead of learning an additional value estimator, although TMPO's target remains distribution-aware rather than a reward-only advantage.

We adapt FlowRL-style optimization to the FLUX.1-dev OCR setting in Section 5, using EasyOCR rewards and offline normalizer pretraining before joint policy updates. Table 7 reports OCR accuracy

over 250 training steps. The adaptation drops from 0.591 to 0.429, while TMPO reaches 0.935; FlowRL-style's final PickScore is 20.681, below the untrained backbone's 21.968. Inaccurate normalizer estimates coupled to policy learning offer a possible explanation, although this experiment does not isolate their effect from other implementation differences. The result concerns this diffusion adaptation of FlowRL.

Table 7: EasyOCR accuracy at training checkpoints (↑). The FlowRL-style diffusion adaptation uses an offline-pretrained normalizer before joint updates.

Training step	0	50	100	150	200	250
TMPO	0.591	0.700	0.811	0.862	0.931	0.935
FlowRL-style	0.591	0.463	0.398	0.421	0.448	0.429

DAG-DB and DGFS-SubTB. DAG [29] uses local detailed-balance constraints and a state-conditioned flow, while DGFS [23] fits flows across partial trajectories through subtrajectory balance. Such intermediate signals can help propagate credit over long diffusion chains. The published DAG and DGFS settings use 50-step DDIM and 100-step stochastic diffusion, respectively. By contrast, each TMPO FLUX trajectory has only three stochastic transitions. Terminal reward therefore motivates directly matching the complete short trajectory without fitting intermediate state flows. Forward-looking intermediate rewards require additional estimates, and combining multiple subtrajectory lengths adds computation. This is a setting-specific rationale for TMPO's design, not a claim that complete-trajectory matching dominates local or subtrajectory methods in general.

E.2 OCR Verification with a Different Recognizer

The main OCR setting uses EasyOCR, which can accept misspelled strings, as in the "Alanac" and "To Sumnit" examples in Figure 1.

To check robustness to the recognizer, we replace EasyOCR with PP-OCRv6-medium for both training and evaluation and retrain Flow-GRPO and TMPO, the two strongest methods by OCR accuracy in Table 1, under otherwise identical settings. Table 8 reports this separate experiment; its OCR scores should be compared within the table rather than substituted into the EasyOCR comparison.

Table 8: Separate OCR retraining and evaluation with PP-OCRv6-medium.

Method	OCR	HPS-v2.1	ImageReward	PickScore	LGMD	Cos.Div.
Flow-GRPO	0.942	0.294	1.092	21.350	-0.107	0.214
TMPO	0.945	0.308	1.098	22.311	0.107	0.245

TMPO achieves slightly higher OCR accuracy and higher preference and diversity scores in this separate recognizer setting.

E.3 GenEval Category Breakdown

Table 9 decomposes the GenEval results from Table 1. TMPO ties for the best single-object score and has the highest scores on two objects, colors, position, and attribute binding. Its counting score is 0.002 below Flow-GRPO. The category average is 0.949, consistent with the main table.

Table 9: GenEval category scores for the compositional-generation comparison in Table 1. Best category scores are bold.

Method	Single	Two	Counting	Colors	Position	Binding	Overall
FLUX.1-dev	0.968	0.795	0.723	0.771	0.199	0.426	0.647
Flow-GRPO	**1.000**	0.989	**0.943**	0.913	0.971	0.842	0.943
MixGRPO	**1.000**	0.977	0.931	0.894	0.959	0.831	0.932
TreeGRPO	**1.000**	0.987	0.938	0.903	0.969	0.837	0.939
GARDO	**1.000**	0.982	0.923	0.879	0.937	0.817	0.923
TMPO	**1.000**	**0.992**	0.941	**0.919**	**0.986**	**0.856**	**0.949**

E.4 No-Tree Runtime under Matched Execution

The 123.7 s no-tree ablation in Table 4 retains TMPO's prompt-group placement: each GPU handles one complete group of $K = 27$ trajectories, so eight GPUs process eight groups in parallel. Removing the tree removes prefix sharing while retaining this placement. Under memory constraints, each GPU then needs chunked log-probability recomputation for its independent trajectories. The independent-rollout baselines instead shard a prompt group across eight GPUs, avoiding that per-GPU chunking.

A no-tree Softmax-TB run using the baseline rollout execution scheme takes 105.9 s per iteration, close to Flow-GRPO's 108.9 s under PickScore-only training. The complete TMPO implementation takes 68.3 s. Thus, the 123.7 s entry in Table 4 measures loss of prefix reuse together with the consequences of retaining per-GPU group placement; the matched-execution timing distinguishes these implementation costs from the trajectory-matching objective.

E.5 Qualitative Comparison with Baselines

Figures 7, 8, and 9 present qualitative comparisons between TMPO and baseline methods on the GenEval (compositional image generation), OCR (visual text rendering), and PickScore (human preference alignment) protocols, respectively. Across all three tasks, TMPO generates images that are both faithful to the text prompt and visually diverse, whereas GRPO-based baselines tend to produce near-duplicate outputs or sacrifice prompt fidelity for reward maximization.

E.6 Evolution of Evaluation Images Across Training Steps

Figures 10, 11, and 12 visualize how generated images evolve across training steps under the GenEval, OCR, and PickScore protocols, respectively. As training progresses, TMPO steadily improves task-specific quality (compositional correctness, text legibility, and aesthetic appeal) while preserving sample diversity throughout the optimization process, avoiding the mode collapse commonly observed in reward-maximizing methods.

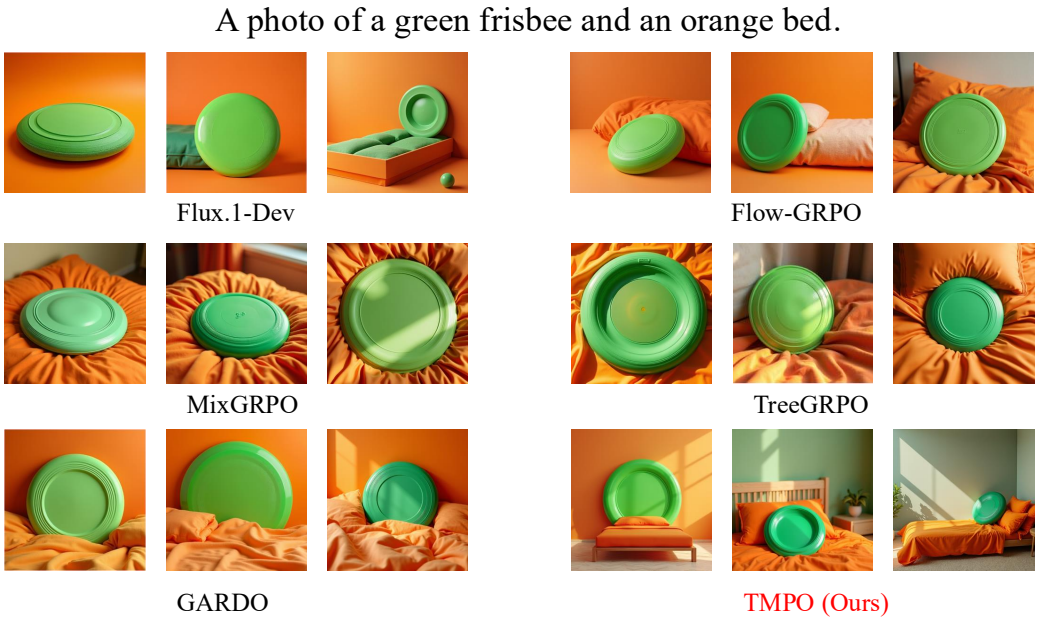


Figure 7: Qualitative comparison on **GenEval** (compositional image generation). TMPO faithfully renders all specified objects, attributes, and spatial relations while maintaining diverse compositions across samples.

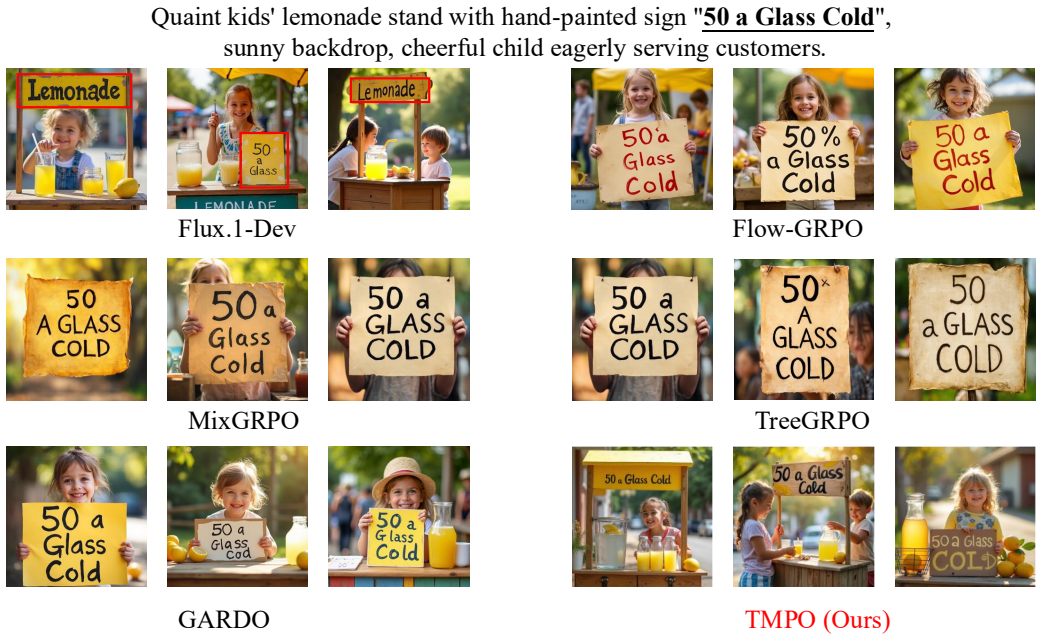


Figure 8: Qualitative comparison on **OCR** (visual text rendering). TMPO accurately renders the target text strings with high legibility while preserving visual diversity in background and style, whereas baselines either produce near-duplicate layouts or exhibit text rendering errors.

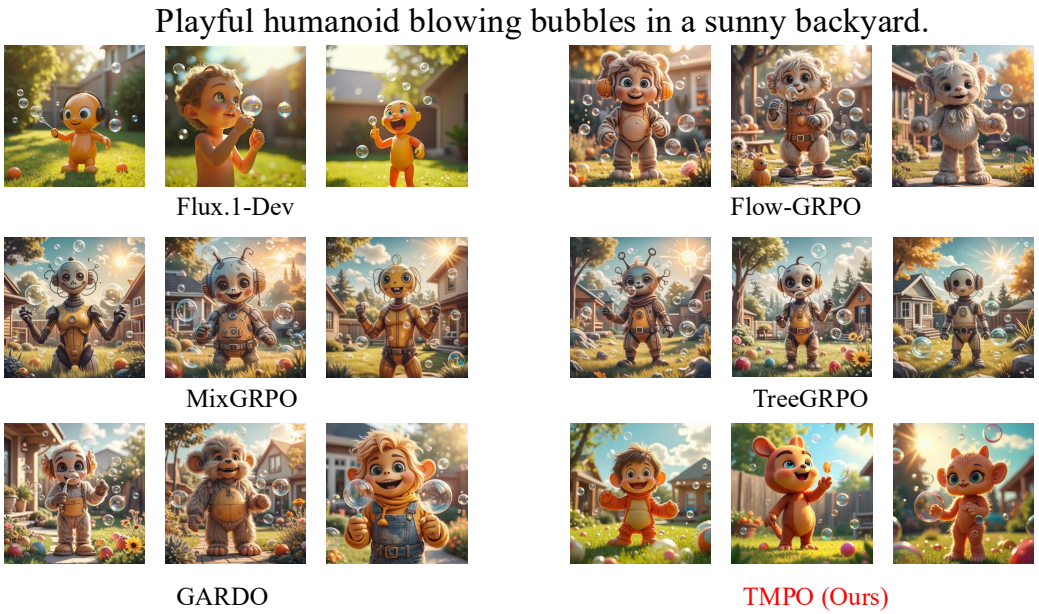


Figure 9: Qualitative comparison on **PickScore** (human preference alignment). TMPO produces aesthetically appealing and prompt-faithful images with noticeably greater diversity in composition, color palette, and viewpoint compared to baselines.

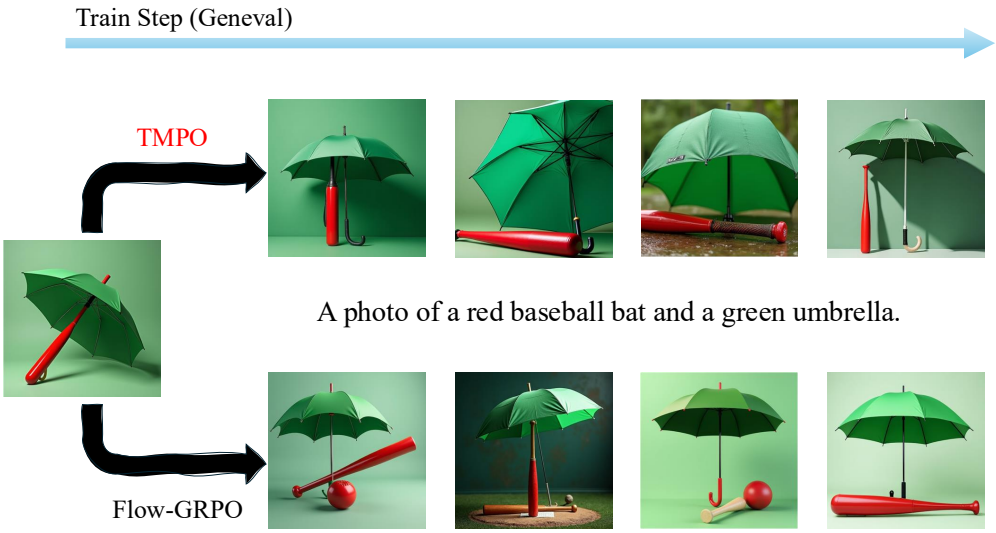


Figure 10: Evolution of generated images across training steps on **GenEval**. TMPO progressively improves compositional accuracy while maintaining diverse object arrangements and visual styles throughout training.

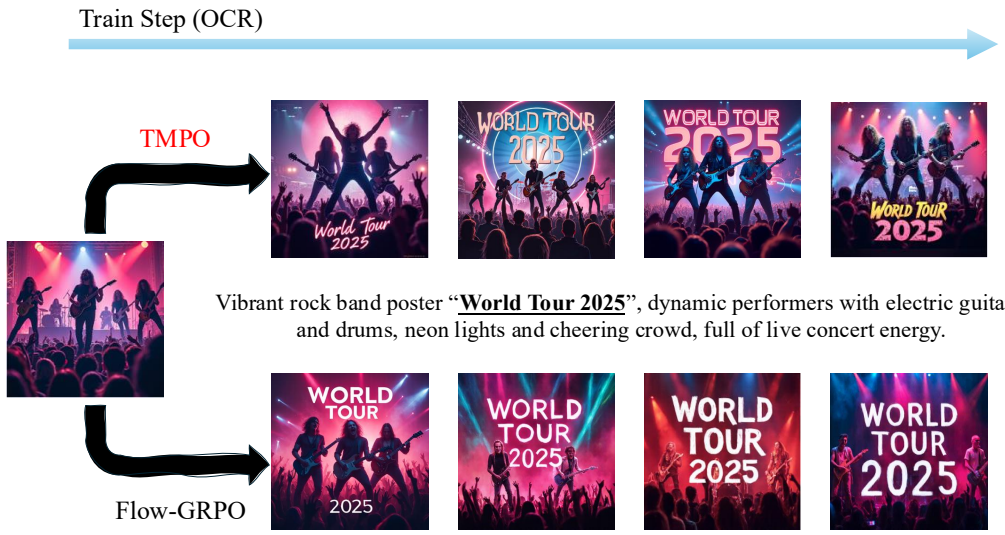


Figure 11: Evolution of generated images across training steps on **OCR**. Text rendering quality improves steadily, with the model learning to produce legible characters while retaining diverse visual layouts.

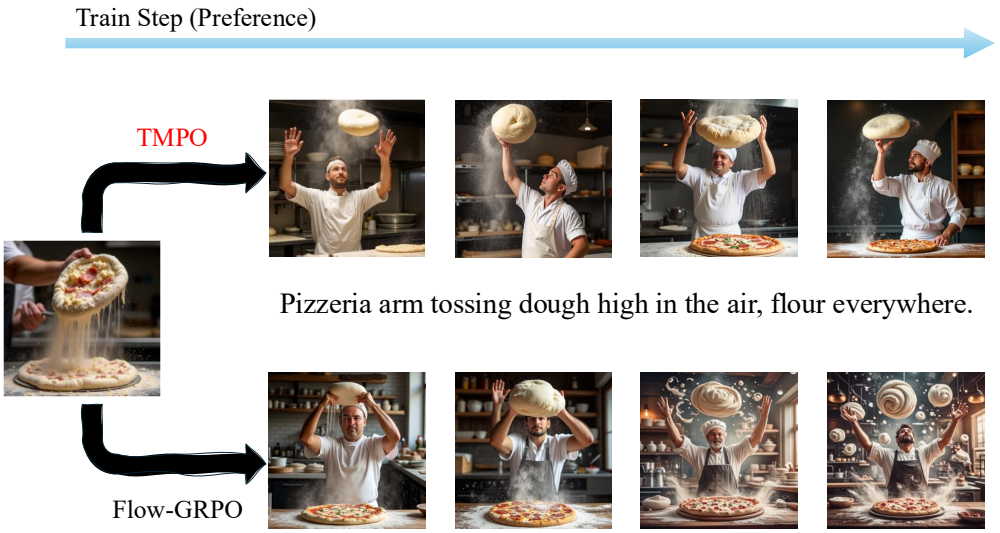


Figure 12: Evolution of generated images across training steps on **PickScore**. Image aesthetics and prompt fidelity improve progressively, while sample diversity is preserved even at later training stages.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 4 supports the trajectory-matching and within-group mode-covering claims; Section 5 evaluates alignment, diversity, and efficiency. The Conclusion states limitations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Conclusion discusses dependence on reward-model quality and the restriction to text-to-image evaluation. Video, 3D, and robotic generation remain future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Assumptions and the within-group result appear in Section 4.1. Appendices A and B provide the matching proofs, importance-sampling derivation, and RatioNorm analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5 specifies protocols, metrics, and baselines. Appendix D details the models, training pipeline, hyperparameters, and compute; Appendix E reports detailed comparisons and robustness analyses.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: https://github.com/MAIR-Lab-HUST/TMPO provides training, inline evaluation, configurations, dependencies, and commands for PickScore, OCR, and joint rewards; GenEval needs an external scorer. Trained TMPO checkpoints and baseline reproduction scripts are not bundled, so full reproduction is incomplete.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 specifies the backbone, LoRA, image resolution, and evaluation protocols. Appendix D.4 gives optimizer, learning-rate, clipping, temperature, tree, and EMA settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: Independent training runs require 42–204 GPU-hours each, so we do not report multi-run error bars (Appendix D.6). Automatic evaluation uses 10 samples per held-out prompt and multiple complementary metrics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D.6 reports 8 GPUs, FSDP, bf16, and per-run GPU-hours. Main result tables report iteration times.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics. The research reported in this manuscript uses publicly available models and datasets and reports automated evaluations only.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This algorithmic study changes the objective and sampler for an existing T2I model, without introducing a new application domain or deployment. Societal implications are inherited from the base model and the existing diffusion-alignment setting.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: TMPO uses LoRA to change the distribution of outputs from FLUX.1-dev with existing reward models. We do not release a new foundation model or scraped dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Licenses: FLUX.1-dev Non-Commercial License (weights); Apache-2.0 (HPS-v2.1/ImageReward code and weights; `ymhao/HPDv2` data); MIT (PickScore code). PickScore-v1's model card states no weight license. Sources: Table 6.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Public code: `https://github.com/MAIR-Lab-HUST/TMPO`. README and setup guide cover dependencies, model/data preparation, and training commands.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This manuscript reports automated evaluations only; no crowdsourcing or human-participant evaluation is reported.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human-participant study is reported in this manuscript; the reported experimental results use automated evaluation.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component of the core methodology in this research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.