REFORMFLOW: REFORMING FLOW REINFORCEMENT VIA DIRECTLY MATCHING REWARD DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Online reinforcement learning for flow matching is challenged by costly terminal-density evaluation. Recent approaches introduce stochastic transitions for likelihood-ratio updates or use local reward-weighted velocity regression. This motivates a terminal-sample objective that retains deterministic generation while specifying the reward-tilted distribution toward which the policy is optimized. We propose **ReFormFlow**, an on-policy framework motivated by reverse Kullback–Leibler minimization toward an exponentially reward-tilted reference distribution. An auxiliary path-space analysis motivates a velocity-regression surrogate for the terminal log-density ratio. ReFormFlow aggregates self-normalized regression contributions across re-noised timesteps into a single residual against the scaled, group-centered reward, jointly learning the velocity field and a prompt-conditioned log-partition. Unlike per-step regression, reward supervision applies to the combined contribution. Both training rollouts and inference retain deterministic ODE sampling without classifier-free guidance. Experiments cover single- and multi-reward optimization and out-of-domain evaluation. ReFormFlow is up to $4.9\times$ faster than DiffusionNFT at matched reward levels in the single-reward comparisons, measured in sampling-and-training GPU-hours. On PickScore, it achieves 24.20 in about 1k updates, compared with 23.82 in 2k updates for DiffusionNFT.

1 INTRODUCTION

The success of Reinforcement Learning (RL) in aligning Large Language Models with human preferences Christiano et al. (2017); Ziegler et al. (2019); Achiam et al. (2023); Comanici et al. (2025) has motivated its extension to continuous generative models. Within this domain, Flow Matching (FM) Lipman et al. (2022) provides a deterministic generation framework underlying modern image generators Esser et al. (2024); Black Forest Labs et al. (2025); Wu et al. (2025). However, applying RL to FM encounters a computational obstacle: evaluating the probability of a generated terminal sample. Unlike autoregressive models with directly computable token probabilities, FM induces a terminal distribution without a readily evaluable density. Computing the log-density of a clean latent x_1 requires integrating the trace of the velocity field's Jacobian along the generative trajectory Chen et al. (2018); Grathwohl et al. (2018), which is costly in high-dimensional latent spaces. Since rewards evaluate the final generated sample, this obstacle raises a central question: how can tractable velocity-field evaluations provide a terminal-distribution optimization signal that reflects the reward?

To bypass this obstacle, one line of work discretizes generation into a multi-step Markov Decision Process. Recent approaches such as Flow-GRPO and DanceGRPO Liu et al. (2025); Xue et al. (2025b) introduce stochasticity through Stochastic Differential Equation (SDE) samplers, constructing tractable Gaussian transition probabilities for policy-gradient optimization. This makes transition-level likelihood ratios available, but replaces deterministic training rollouts with stochastic dynamics that differ from Ordinary Differential Equation (ODE) deployment. Moreover, their reported configurations use Classifier-Free Guidance (CFG) Ho & Salimans (2022) at inference, requiring both conditional and unconditional model evaluations. These approaches demonstrate the applicability of policy gradients to FM, while motivating reward fine-tuning that retains deterministic rollouts and supports deployment without external guidance.

A contrasting line of work pursues implicit optimization, casting reward fine-tuning as likelihood-free velocity regression. DiffusionNFT Zheng et al. (2025) uses reward-weighted positive and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

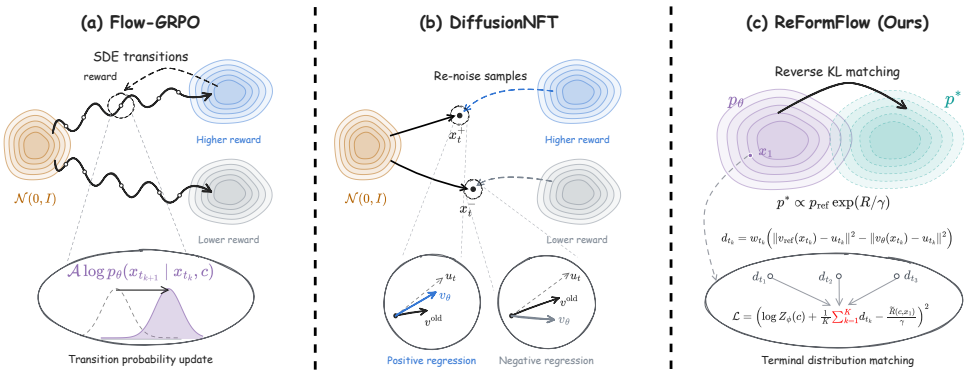


Figure 1: Conceptual comparison. (a) Flow-GRPO updates stochastic transition likelihoods using advantages. (b) DiffusionNFT uses positive and negative velocity regression. (c) ReFormFlow aggregates timestep contributions for terminal-sample regression toward a reward-tilted target.

negative velocity regression terms to avoid explicit sample-density evaluation, demonstrating training-efficiency gains over SDE-based policy gradients. Its updates are constructed by mixing the current and old-policy velocities, with an additional parameter controlling the correction scale. This local regression formulation does not directly specify the terminal density ratio that each generated sample should satisfy, nor does it constrain the contributions from different timesteps to jointly account for that ratio. We instead specify the reward-induced terminal distribution first and use it to organize regression, rather than defining updates through local velocity mixing.

We introduce **ReFormFlow**, an online framework that reforms flow reinforcement through terminal-sample distribution matching. Figure 1 contrasts the optimization signals used by these approaches. Starting from the analytic optimum of KL-regularized reward maximization, $p^*(x_1 \mid c) \propto p_{\text{ref}}(x_1 \mid c) \exp(R(c, x_1)/\gamma)$ Peters et al. (2010); Levine (2018), we adopt the terminal reverse Kullback–Leibler (KL) divergence as the ideal optimization objective. This target specifies how rewards should change sample probabilities relative to the reference, but does not itself make terminal densities tractable. Drawing on Trajectory Balance Whitammer et al. (2022), we organize the model-to-reference log-density ratio, terminal reward, and conditional normalization term into a single squared residual. ReFormFlow constructs a tractable terminal-sample estimator for this residual and learns from samples generated by the current policy. Reward feedback is incorporated into the velocity field through regression, preserving ODE rollouts during training and CFG-free deployment.

To address the inaccessible terminal density, Distribution Matching Distillation (DMD) Yin et al. (2023) replaces terminal KL with divergences between noised marginal distributions and constructs updates from per-timestep score differences. ReFormFlow instead aggregates information across timesteps into a unified optimization signal for the clean terminal sample, so that terminal rewards directly supervise sample-level deviations from the reference distribution. Our construction uses computable velocity-prediction errors to construct a surrogate for the terminal log-density ratio. A shared-diffusion auxiliary analysis, informed by Girsanov's theorem Oksendal (2013) and local Gaussian transition calculations, motivates accumulated velocity-error contributions. For each generated sample, we evaluate these contributions at multiple re-noised timesteps, aggregate them into this surrogate, and compare it with the reward in one squared residual. The timesteps thus contribute jointly to the sample-level residual, without each local contribution having to match the full terminal reward separately. Actual training rollouts and inference remain deterministic ODEs, with a learnable conditional normalization term, group-relative reward centering Shao et al. (2024), and adaptive scaling supporting online optimization. **We summarize our contributions as follows:**

- We introduce **ReFormFlow**, an on-policy FM framework motivated by direct distribution matching through reverse KL. It couples an analytic reward-tilted target with terminal-sample regression while retaining deterministic ODE training and inference.
- We derive a velocity-regression surrogate for terminal density-ratio estimation, complemented by a learnable conditional log-partition, group-relative rewards, and adaptive reward scaling to support on-policy optimization across different reward tasks.

- ReFormFlow achieves competitive or improved quality across multiple tasks and is up to $4.9\times$ faster than DiffusionNFT at matched reward levels in the reported single-reward comparisons on SD3.5-Medium Esser et al. (2024), measured in sampling-and-training GPU-hours. It reaches a PickScore of 24.20 in about 1k steps. Multi-reward evaluations use 40-step training rollouts, with additional out-of-domain assessment.

2 Related Work

Reward-based Fine-tuning and Guidance. Reward-based adaptation of diffusion and flow models includes weighted regression, differentiable reward optimization, and inference-time guidance. Early approaches such as Reward-Weighted Regression (RWR) Lee et al. (2023); Dong et al. (2023) favor high-reward samples through weighted or filtered fine-tuning. Methods that differentiate rewards through the sampling process Xu et al. (2023); Prabhudesai et al. (2024); Clark et al. (2023) instead require differentiable feedback and incur backpropagation costs. Inference-time guidance Janner et al. (2022); Jin et al. (2025) and training-free methods such as DyMO Xie & Gong (2025) steer generation without policy fine-tuning, but introduce additional computation during sampling. Recent methods incorporate reward feedback into objectives related to pretraining. Advantage Weighted Matching (AWM) Xue et al. (2025a) formulates policy-gradient optimization through advantage-weighted score or flow matching, connecting RL updates to the pretraining objective. DiffusionNFT Zheng et al. (2025) uses reward-weighted positive and negative velocity regression to perform likelihood-free policy optimization. These regression methods avoid explicit terminal-density evaluation.

Preference and Transition-based Policy Optimization. Another line of work adapts preference learning and policy optimization to iterative generation. D3PO Yang et al. (2024) and Diffusion-DPO Wallace et al. (2024) learn from preference pairs, while SPO Liang et al. (2025) introduces step-aware preference supervision. Online methods such as DDPO Black et al. (2023) and DPOK Fan et al. (2023) formulate denoising as a multi-step MDP. For FM, Flow-GRPO Liu et al. (2025) and DanceGRPO Xue et al. (2025b) use stochastic transitions to obtain likelihood ratios for policy updates. Subsequent work refines how exploration and credit are distributed across timesteps. MixGRPO Li et al. (2025b) combines ODE and SDE sampling, while TempFlow-GRPO He et al. (2026) combines trajectory branching for process rewards, noise-aware timestep weighting, and seed grouping to distinguish initialization effects from exploration. These flow-based policy-gradient methods use stochastic training transitions. ReFormFlow instead retains deterministic ODE rollouts and regresses cross-timestep density-ratio estimates against terminal rewards.

3 Preliminaries

3.1 Flow Matching and Deterministic Density Evolution

For $x_0 \sim p_0 = \mathcal{N}(0, I)$ and $x_1 \sim p_1(\cdot \mid c)$ drawn from the noise prior and conditional data distribution, respectively, we use the linear FM path Lipman et al. (2022); Liu et al. (2023):

$$x_t = tx_1 + (1-t)x_0, \qquad t \in [0,1]. \tag{1}$$

The sampled pair defines the velocity target $u_t = x_1 - x_0$. The learned field $v_\theta(x_t, c, t)$ generates samples through the ODE $\dot{x}_t = v_\theta(x_t, c, t)$, starting from $x_0 \sim p_0$.

Along the model ODE trajectory, rather than the linear interpolation above, the continuous change-of-variables formula Chen et al. (2018); Grathwohl et al. (2018) gives the terminal log-density:

$$\log p_\theta(x_1|c) = \log p_0(x_0) - \int_0^1 \mathrm{Tr}\left(\nabla_{x_t} v_\theta(x_t, c, t)\right) \mathrm{d}t, \tag{2}$$

where $x_1 \sim p_\theta(\cdot \mid c)$ is now a generated endpoint. Evaluating the Jacobian-trace integral is costly in high-dimensional latent spaces, motivating alternatives to direct density evaluation.

3.2 Reinforcement Learning for Flow Matching

Policy Gradient Algorithms via SDE Exploration. Flow-GRPO and DanceGRPO Liu et al. (2025); Xue et al. (2025b) introduce stochastic transitions to obtain likelihood ratios for policy

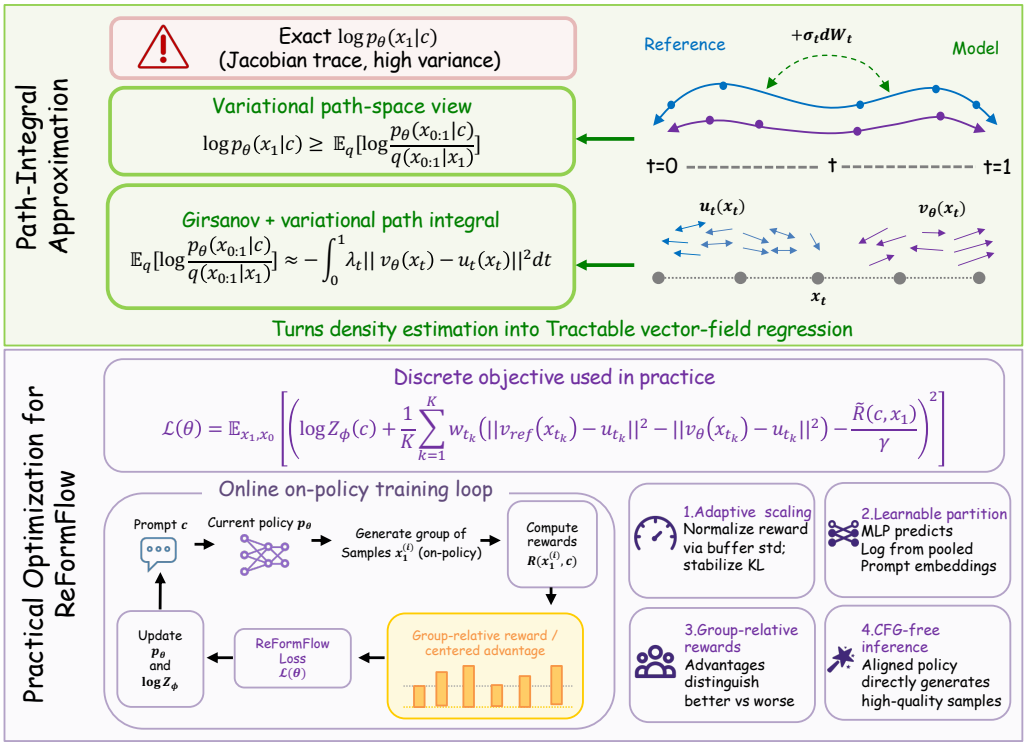


Figure 2: Overview of **ReFormFlow**. **(Top)** The auxiliary path-space formulation motivates velocity regression; the schematic regression expression omits additive terms and path expectations. **(Bottom)** On-policy ODE samples are re-noised, and timestep contributions weighted by w_t are aggregated into a single residual against the group-centered reward $\widetilde{R}$, jointly updating the policy and log head.

updates. Under our noise-to-data time convention, a Gaussian-interpolation-based auxiliary SDE is

$$\mathrm{d}x_t = \left[v_\theta(x_t, c, t) + \frac{g_t^2}{2(1-t)}\left(t v_\theta(x_t, c, t) - x_t\right)\right]\mathrm{d}t + g_t \mathrm{d}W_t. \tag{3}$$

Here g_t controls the diffusion strength and W_t is a standard Wiener process. For $0 < t < 1$, the drift preserves the Gaussian interpolation marginals when the velocity is their exact marginal field; this correspondence is not automatic for an approximate network (Appendix A.3). With $g_t > 0$, Euler–Maruyama discretization yields Gaussian transition densities for likelihood-ratio updates such as PPO Schulman et al. (2017) and GRPO Shao et al. (2024).

Reinforcement Learning as Supervised Fine-tuning. DiffusionNFT Zheng et al. (2025) incorporates rewards through positive and negative velocity regression. Omitting the shared arguments (x_t, c, t), its mixed velocities are

$$v_\theta^{\pm} := v^{\text{old}} \pm \beta\big(v_\theta - v^{\text{old}}\big), \tag{4}$$

where v^{old} is the old-policy velocity and β controls the mixing scale. The reward-weighted loss is

$$\mathcal{L}_{\text{NFT}}(\theta) = \mathbb{E}\left[r\|v_\theta^+ - u_t\|^2 + (1-r)\|v_\theta^- - u_t\|^2\right], \tag{5}$$

where $r \in [0,1]$ is the normalized relative reward of x_1 and u_t is the re-noising target.

4 Method

4.1 Direct Regression via On-Policy Reverse KL Optimization

For a prompt c, KL-regularized reward maximization optimizes $\mathbb{E}_{p_\theta}[R(c, x_1)] - \gamma D_{\text{KL}}(p_\theta \| p_{\text{ref}})$, with a fixed reference distribution and $\gamma > 0$. Its optimum over distributions is the reward-tilted

target Peters et al. (2010); Levine (2018):

$$p^*(x_1 \mid c) = \frac{1}{Z(c)} p_{\text{ref}}(x_1 \mid c) \exp\left(\frac{R(c, x_1)}{\gamma}\right), \tag{6}$$

where $Z(c)$ normalizes the density. This target also appears in reward and preference optimization for diffusion and flow models Domingo i Enrich et al. (2025); Li et al. (2025a). We adopt $D_{\mathrm{KL}}(p_\theta \| p^*)$ as the ideal objective, which permits sampling from the current policy. Appendix A.1 gives the connection to reward maximization.

Following the trajectory-balance objective of GFlowNets Bengio et al. (2021); Whitammer et al. (2022), we formulate a terminal-sample squared residual that combines the model-to-reference log-density ratio, reward, and conditional log-partition:

$$\min_\theta \mathbb{E}_{x_1 \sim p_\theta}\left[\left(\log Z_\phi(c) + \log p_\theta(x_1|c) - \log p_{\text{ref}}(x_1|c) - \frac{R(c, x_1)}{\gamma}\right)^2\right]. \tag{7}$$

Samples are drawn from the current policy and held fixed during differentiation, with no gradient through the sampling process. Learning the conditional log-partition $\log Z_\phi(c)$ alongside the policy follows the normalization-learning approach of GFlowNets and TB Bengio et al. (2021); Whitammer et al. (2022), also used in FlowRL for LLM reasoning Zhu et al. (2025).

The on-policy gradient connection between TB and reverse KL is established in prior work Whitammer et al. (2023). With exact terminal densities and fixed reward, reference, and temperature, the expected policy semi-gradient of Eq. 7, with samples and ϕ fixed, equals twice the reverse-KL gradient. This holds for any sample-independent $\log Z_\phi(c)$, without requiring it to equal $\log Z(c)$. Appendix A.2 restates the derivation in our notation. This relation motivates on-policy terminal-sample regression. The next section constructs a tractable density-ratio surrogate for this loss.

4.2 Terminal Density-Ratio Estimation

Directly matching the reward-tilted target in Eq. 7 requires evaluating terminal densities. Unlike autoregressive LLMs, FM does not provide these densities directly. We first consider a per-step approximation that regresses local contributions against the terminal reward. We then develop a terminal log-density-ratio surrogate by aggregating these contributions across timesteps, so that reward regression constrains their combined effect rather than each timestep individually.

Per-step distribution matching. Combining DMD's per-timestep distribution matching Yin et al. (2023) with GRPO-style application of terminal rewards across steps Shao et al. (2024); Liu et al. (2025), we construct a *Per-Step Reg* baseline. At each re-noised timestep, we use the weighted velocity-error difference as a local surrogate for the model-to-reference log-density ratio, forming a separate squared residual against the same scaled, group-centered reward used by ReFormFlow:

$$\mathcal{L}_{\text{Per-Step}} = \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\left(\log Z_\phi(c) + w_{t_k}\left[\|v_{\text{ref}} - u_{t_k}\|_2^2 - \|v_\theta - u_{t_k}\|_2^2\right] - \frac{\widetilde{R}(c, x_1)}{\gamma}\right)^2\right]. \tag{8}$$

Each generated endpoint is re-noised at K timesteps using the linear interpolation in Section 3.1. Shared velocity arguments are omitted for brevity. The weight implements DMD-inspired self-normalized x_1 regression in velocity space Yin et al. (2023):

$$w_t = \frac{1 - t}{\mathrm{sg}[\mathrm{mean}(|v_\theta(x_t, c, t) - u_t|)]}. \tag{9}$$

where mean averages over latent coordinates and sg denotes stop-gradient.

Terminal-sample regression. A local regression term does not by itself specify the terminal log-density ratio required by Eq. 7. ReFormFlow instead combines model-to-reference regression-error contributions across re-noised timesteps before forming a single reward-conditioned residual. We next motivate this construction through a shared-diffusion path-space representation. For this construction, we start from the variational expression over the trajectory $x_{0:1}$:

$$\log p_\theta(x_1|c) = \log \mathbb{E}_{q(\cdot|x_1)}\left[\frac{p_\theta(x_{0:1}|c)}{q(x_{0:1}|x_1)}\right] \geq \mathbb{E}_{q(\cdot|x_1)}\left[\log \frac{p_\theta(x_{0:1}|c)}{q(x_{0:1}|x_1)}\right], \tag{10}$$

Table 1: Our evaluation results compared with DiffusionNFT. Gray-colored : In-domain target task.

Method	#Iter	Task Metric			DrawBench Metric				
		GenEval	OCR	PickScore	ClipScore	HPSv2.1	Aesthetic	ImgRwd	PickScore
SD3.5-M + CFG	—	0.63	0.59	21.75	0.285	0.279	5.36	0.85	22.34
DiffusionNFT	1k	**0.98**	0.35	20.19	0.274	0.240	4.99	0.40	21.78
ReFormFlow	**0.8k**	**0.98**	0.37	20.25	0.278	0.247	5.05	0.45	21.86
DiffusionNFT	0.4k	0.20	**0.98**	19.15	0.233	0.168	4.79	-1.27	20.15
ReFormFlow	**0.3k**	0.23	**0.98**	19.26	0.238	0.173	5.08	-1.23	20.43
DiffusionNFT	2.0k	0.55	0.67	23.82	0.272	0.316	6.19	1.31	24.09
ReFormFlow	**1.0k**	0.65	0.74	**24.20**	0.276	0.315	6.27	1.37	24.34

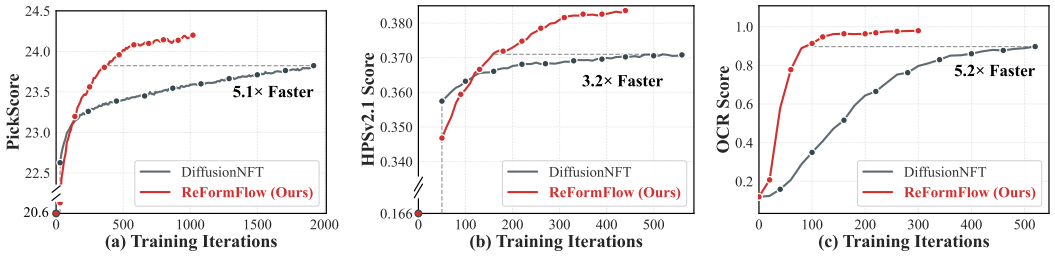


Figure 3: Head-to-head comparison between ReFormFlow and DiffusionNFT on single-reward optimization tasks: (a) PickScore, (b) HPSv2.1, and (c) OCR.

where $q(x_{0:1}|x_1)$ is the reference noise-addition process. In the deterministic limit, distinct ODE path measures can be singular, preventing a direct change-of-measure argument.

For shared-diffusion auxiliary processes used only for analysis, Girsanov's theorem Oksendal (2013) relates path-space log-density ratios to drift differences. This motivates approximating the difference of the policy and reference ELBOs through weighted velocity regression:

$$\mathbb{E}_{q(\cdot|x_1)}\left[\log \frac{p_\theta(x_{0:1} \mid c)}{p_{\text{ref}}(x_{0:1} \mid c)}\right] \approx \mathbb{E}_{q(\cdot|x_1)}\left[\int_0^1 \lambda_t \left(\|v_{\text{ref}} - u_t\|^2 - \|v_\theta - u_t\|^2\right) \mathrm{d}t\right], \tag{11}$$

where $\lambda_t = \frac{1}{2g_t^2}\left(1 + \frac{g_t^2 t}{2(1-t)}\right)^2$ is the local energy coefficient associated with the drift form in Eq. 3. The local calculation in Appendix A.3 does not establish this regression approximation for endpoint-conditioned paths. To approximate the terminal log-density ratio, we additionally omit the posterior-gap difference between the two models, as detailed in Appendix A.3.

4.3 Practical Optimization for ReFormFlow

Substituting the terminal density-ratio surrogate from Section 4.2 into the on-policy regression objective (Eq. 7) gives a practical flow regression loss. We approximate the temporal aggregation by averaging contributions at K re-noised timesteps. Each contribution uses the self-normalized weight w_{t_k} defined in Eq. 9, as in Per-Step Reg. This empirical weighting is not an exact discretization of λ_t; Appendix A.3 details the practical surrogate and its assumptions.

The distinction from Per-Step Reg is where terminal reward supervision enters the objective. Per-Step Reg forms a separate squared reward residual for each local estimate, whereas ReFormFlow combines the timestep-level contributions associated with the same endpoint. Regressing this aggregate against the within-prompt group-centered reward $\widetilde{R}$, defined below, gives the discrete on-policy objective:

$$\mathbb{E}_{\substack{x_1 \sim p_{\bar{\theta}} \\ x_0 \sim p_0}}\left[\left(\log Z_\phi(c) + \frac{1}{K}\sum_{k=1}^{K} w_{t_k}\left(\|v_{\text{ref}}(x_{t_k}) - u_{t_k}\|^2 - \|v_\theta(x_{t_k}) - u_{t_k}\|^2\right) - \frac{\widetilde{R}(c, x_1)}{\gamma}\right)^2\right]. \tag{12}$$

Here $\bar{\theta}$ denotes the policy used to collect the current endpoints; the sampling distribution and endpoints are held fixed during each update. The inner average provides a model-to-reference density-ratio surrogate for each endpoint, while the outer squared residual relates this aggregate to the scaled

reward and conditional log-partition. The auxiliary SDE is used only for the path-space analysis. We generate the endpoints with a deterministic ODE solver and evaluate the regression terms at re-noised versions of these endpoints. Both training rollouts and inference retain deterministic ODE sampling.

Practical implementation. For **adaptive γ scaling**, we use $\widetilde{R}(c, x_1) = R(c, x_1) - \bar{R}_c$ in Eq. 12, where $\bar{R}_c$ is the current on-policy group's mean reward for prompt c, and set γ to the standard deviation of raw rewards across the current on-policy buffer. This combines within-prompt centering with global scaling, as in Flow-GRPO's advantage normalization with its global-standard-deviation option Liu et al. (2025). The **learnable log-partition** $\log Z_\phi(c)$ is predicted by an MLP from pooled prompt embeddings and jointly optimized with the velocity model; Appendix B.2 provides implementation details and mean-output trajectories for numerical inspection (Figure 9). For **CFG-free inference**, we directly use the trained conditional velocity field without guidance extrapolation. CFG Ho & Salimans (2022) admits the score-level interpretation $p_t^{\mathrm{CFG}}(x_t \mid c) \propto p_t(x_t \mid c)[p_t(x_t \mid c)/p_t(x_t)]^s$, with additional guidance strength s, whereas ReFormFlow uses terminal-sample regression toward the reward-tilted target specified by $\exp(R/\gamma)$ in Eq. 6.

5 Experiments

We assess ReFormFlow through three primary evaluation axes: (1) a direct comparative analysis against DiffusionNFT under single-reward optimization scenarios, (2) the effectiveness of sequential multi-reward training, and (3) systematic ablation studies to validate our core algorithmic designs.

5.1 Experimental Setup

Our controlled fine-tuning experiments use SD3.5-Medium Esser et al. (2024) at a 512×512 resolution, following the experimental settings of FlowGRPO Liu et al. (2025) and DiffusionNFT Zheng et al. (2025) where applicable. Table 2 also includes other base models and evaluations of official checkpoints, with their sources and resolution differences marked separately.

Tasks and Evaluation Metrics. We divide our text-to-image alignment tasks into two main categories: (1) *Rule-based constraints*, where we evaluate spatial compositional reasoning via GenEval Ghosh et al. (2023) and text generation fidelity using OCR accuracy. (2) *Model-based human preference*, deploying PickScore Kirstain et al. (2023) as the principal reward feedback. To measure out-of-domain robustness, we further test the fine-tuned models on the DrawBench Saharia et al. (2022) dataset, calculating comprehensive metrics including PickScore, CLIPScore Hessel et al. (2021), HPSv2.1 Wu et al. (2023), Aesthetic Score Schuhmann (2022), and ImageReward Xu et al. (2023).

Training and Evaluation. Our fine-tuning runs use LoRA ($\alpha = 64, r = 32$), with 48 prompts and $G = 24$ samples per prompt in each rollout collection. Single-reward training rollouts use 10 solver steps; the DiffusionNFT and ReFormFlow multi-reward results in Table 2 use 40-step training rollouts. All reported ReFormFlow and DiffusionNFT checkpoints are evaluated using a 40-step first-order ODE solver. Training rollout steps and training iterations are reported separately.

5.2 Head-to-Head Comparison on Single Rewards

Table 1 reports task-specific checkpoints, while Figure 3 shows reward trajectories. On PickScore, ReFormFlow attains 24.20 in about 1k iterations, compared with 23.82 at 2k iterations for DiffusionNFT. Appendix Figure 7 reports the corresponding curves against GPU-hours. Additional comparisons with AWM Xue et al. (2025a) and TempFlow-GRPO He et al. (2026) across PickScore, HPSv2.1, and OCR are reported against GPU-hours in Appendix B.1 (Figure 8). For OCR, Table 1 uses DiffusionNFT's task-specific configuration ($\beta = 0.1$ and a delayed old-policy EMA), which reaches 0.98 at 400 iterations; ReFormFlow reaches 0.98 at 300 iterations. The OCR curves in Figures 3(c) and 7(c) instead use DiffusionNFT's general configuration ($\beta = 1$), as detailed in Appendix B.2.

Table 1 also evaluates transfer to DrawBench. ReFormFlow improves most of the reported DrawBench metrics over the corresponding DiffusionNFT checkpoints. However, the GenEval- and OCR-optimized checkpoints have lower aesthetic scores than the base model with CFG. These results motivate the sequential multi-reward experiments below.

Table 2: Multi-reward evaluation. DiffusionNFT and ReFormFlow use 40-step training rollouts and 40-step ODE evaluation. †: our evaluation of official checkpoints; ‡: 1024×1024 evaluation. Each row reports one checkpoint across all metrics. **Bold**: best; underline: second best, including ties.

Model	#Iter	Task Metric			DrawBench Metric				
		GenEval	OCR	PickScore	ClipScore	HPSv2.1	Aesthetic	ImgRwd	PickScore
SD-XL‡	—	0.55	0.14	21.74	0.287	0.280	5.60	0.76	22.42
SD3.5-L‡	—	0.71	0.68	22.24	0.289	0.288	5.50	0.96	22.91
FLUX.1-Dev	—	0.66	0.59	22.55	**0.295**	0.274	5.71	0.96	22.84
SD3.5-M (w/o CFG)	—	0.24	0.12	19.90	0.237	0.204	5.13	-0.58	20.51
+ CFG	—	0.63	0.59	21.75	0.285	0.279	5.36	0.85	22.34
+ FlowGRPO†	>5k	**0.95**	0.66	21.67	0.293	0.274	5.32	1.06	22.37
	2k	0.66	**0.92**	21.80	0.290	0.280	5.32	0.95	22.44
	4k	0.54	0.68	23.33	0.280	0.316	5.90	1.29	23.50
+ DiffusionNFT†	1.7k	0.94	0.91	23.36	0.293	0.331	6.01	1.49	23.80
+ ReFormFlow	1.5k	0.93	0.91	**23.52**	0.293	**0.338**	**6.09**	**1.52**	**24.11**

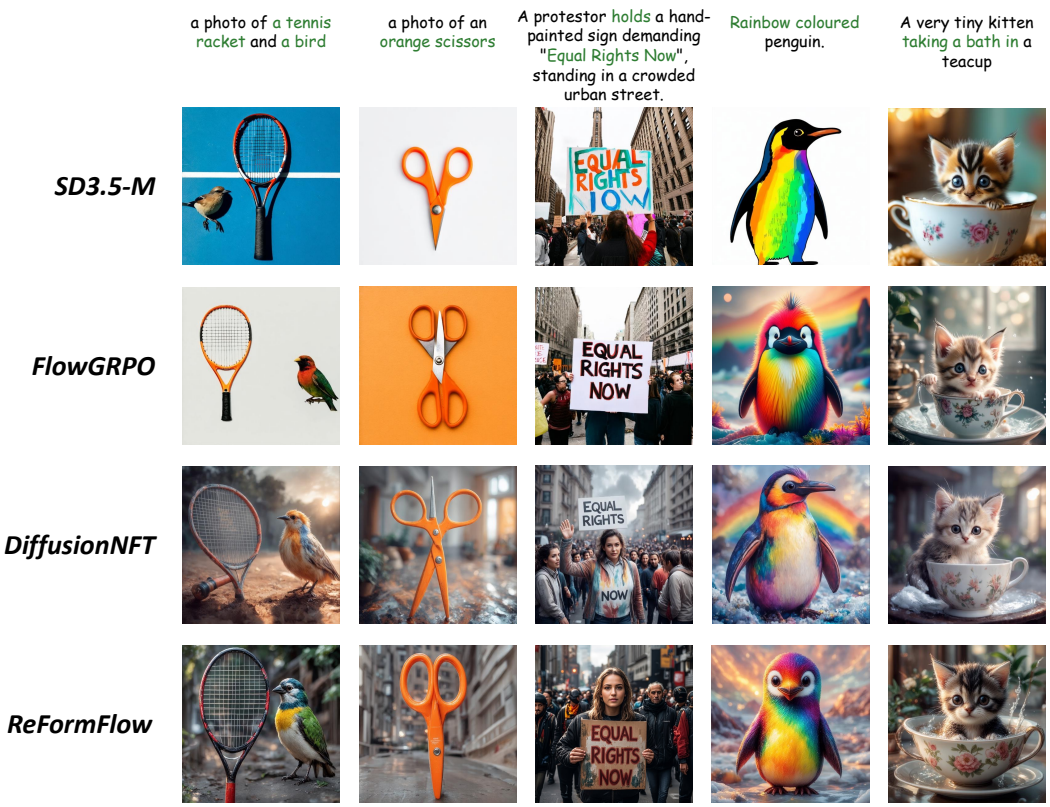


Figure 4: Qualitative comparison on prompts sampled from GenEval, OCR, PickScore, and DrawBench, illustrating visual differences between ReFormFlow and the baselines.

5.3 Multi-Reward Sequential Training

We optimize multiple objectives in one model through sequential fine-tuning. Starting from SD3.5-Medium configured for CFG-free generation, we train with PickScore, CLIPScore, and HPSv2.1 on Pick-a-Pic Kirstain et al. (2023), then alternate preference-training stages with stages that additionally optimize GenEval or OCR, as described in Appendix B.2. Each row in Table 2 evaluates the same checkpoint across all task and DrawBench metrics, rather than combining results from separate task experts. Figure 4 provides qualitative comparisons.

Both methods use 40-step training rollouts in Table 2; the DiffusionNFT results come from our evaluation of its official checkpoint. ReFormFlow scores higher on five of the eight metrics, matches OCR and CLIPScore, and scores lower on GenEval (0.93 versus 0.94), after about 1.5k training iterations compared with DiffusionNFT's 1.7k. Thus, the comparison uses the same rollout step count while reporting the different iteration budgets explicitly.

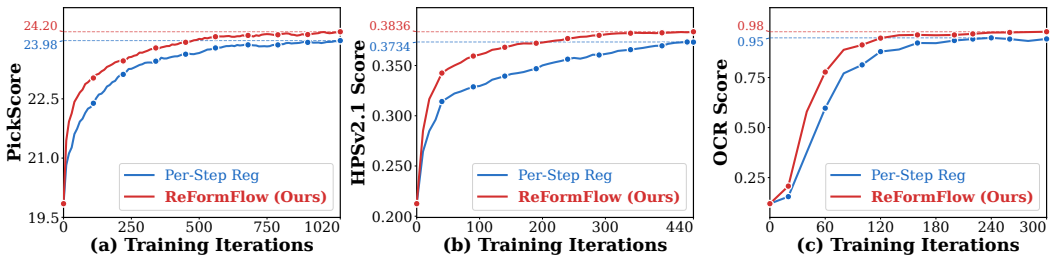


Figure 5: Terminal-sample regression versus Per-Step Reg on (a) PickScore, (b) HPSv2.1, and (c) OCR. Dashed lines mark the best scores under matched iteration budgets.

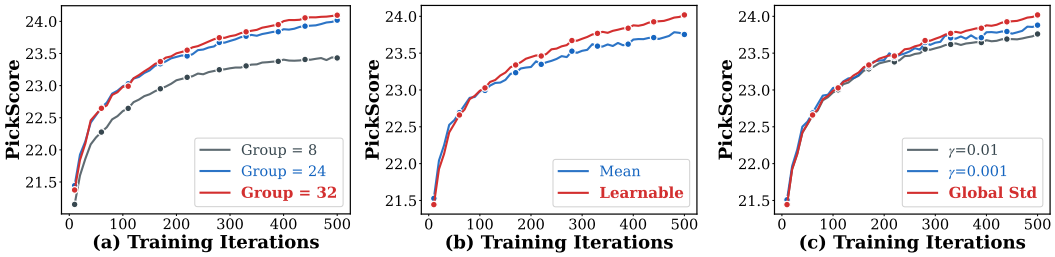


Figure 6: Component ablations on PickScore: (a) group size G, (b) learnable log-partition versus a prompt-mean baseline, and (c) adaptive γ scaling versus fixed temperatures.

5.4 Ablation Study

We examine whether reward supervision should constrain individual timesteps or their aggregate across three tasks. We then assess how group size, the learnable log-partition, and adaptive reward scaling affect PickScore optimization.

Terminal-Sample versus Per-Step Regression. Figure 5 compares Eqs. 12 and 8. ReFormFlow squares the reward residual after aggregating timestep contributions; Per-Step Reg squares a separate residual at each timestep. Under matched iteration budgets, the best scores are 24.20 versus 23.98 on PickScore, 0.3836 versus 0.3734 on HPSv2.1, and 0.98 versus 0.95 on OCR.

Group Size. Figure 6(a) shows that $G = 8$ attains lower PickScore than $G = 24$ and $G = 32$. Increasing the group size from 24 to 32 gives a smaller additional gain. We use $G = 24$ as default.

Learnable Partition Function. In Figure 6(b), the learnable $\log Z_\phi(c)$ reaches a higher PickScore than the prompt-mean baseline, with the difference widening in the later iterations.

Adaptive γ Scaling. Figure 6(c) compares global-standard-deviation scaling with fixed $\gamma \in \{0.01, 0.001\}$. Adaptive scaling achieves the highest PickScore by the end of the displayed interval; the fixed-temperature variants also continue to improve.

6 Conclusion

We presented **ReFormFlow**, an on-policy framework for reward fine-tuning of flow matching models. A reward-tilted reference distribution defines the ideal target, while auxiliary path-space analysis motivates a velocity-regression surrogate. ReFormFlow aggregates self-normalized contributions across re-noised timesteps into a single terminal reward residual, jointly learning the velocity field and a conditional log-partition. Training rollouts and inference retain deterministic ODE sampling without classifier-free guidance. Experiments support this formulation through single- and multi-reward comparisons and timestep-aggregation ablations, without establishing equivalence to exact reverse-KL optimization. Future work will investigate integrating distribution matching distillation (DMD) with ReFormFlow for reward-aware few-step generation. We will study how to coordinate teacher-distribution matching and terminal reward regression on shared on-policy samples, and how to adapt timestep aggregation and normalization to distilled generators. This direction will examine the trade-offs among reward improvement, fidelity to the teacher distribution, sample diversity, and sampling cost, toward a joint DMD–RL training framework.

AI USE STATEMENT

We used generative AI tools to assist with manuscript preparation, primarily for grammar correction, refinement of wording, and LaTeX formatting. AI-assisted checks do not replace author verification. The authors take responsibility for the final manuscript, including its technical claims and derivations.

ETHICS STATEMENT

Our experiments use publicly available pretrained models and established reward functions to evaluate ReFormFlow. We have not identified additional ethical concerns specific to this study beyond the general risks associated with text-to-image generation and reward-based optimization.

REPRODUCIBILITY STATEMENT

We document the practical optimization objective in Section 4.3 and provide the supporting assumptions, derivations, and approximation limitations in Appendix A. Section 5.1 describes the experimental tasks and evaluation metrics. Appendix B.2 details the sampling and evaluation protocols, optimizer and LoRA settings, conditional log-partition head, baseline configurations, and multi-reward training procedure; shared training settings are summarized in Table 3. Hardware specifications and the convention used to calculate sampling-and-training GPU-hours are provided in Appendix B.1. These descriptions distinguish the single-reward and multi-reward settings and identify the baseline configurations used in the reported comparisons.

REFERENCES

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

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

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

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

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Carles Domingo i Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations*, 2025.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

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

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

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

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models. In *International Conference on Learning Representations*, volume 2026, pp. 132869–132895, 2026.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

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

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Luozhijie Jin, Zijie Qiu, Jie Liu, Zijie Diao, Lifeng Qiao, Ning Ding, Alex Lamb, and Xipeng Qiu. Inference-time alignment control for diffusion models with reinforcement learning guidance. *arXiv preprint arXiv:2508.21016*, 2025.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Binxu Li, Minkai Xu, Jiaqi Han, Meihua Dang, and Stefano Ermon. Divergence minimization preference optimization for diffusion model alignment. *arXiv preprint arXiv:2507.07510*, 2025a.

Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025b.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Mihir Prabhudesai, Russell Mendonca, Zheyang Qin, Katerina Fragkiadaki, and Deepak Pathak. Video diffusion alignment via reward gradients. *arXiv preprint arXiv:2407.08737*, 2024. URL https://arxiv.org/abs/2407.08737.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Christoph Schuhmann. Laion-aesthetics. https://laion.ai/blog/laion-aesthetics/, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Esmeralda S. Whitammer, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Esmeralda S. Whitammer, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. In *International Conference on Learning Representations*, 2023.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

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.

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

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

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv preprint arXiv:2509.25050*, 2025a.

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*, 2025b.

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

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *arXiv preprint arXiv:2311.18828*, 2023.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025.

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, Qizheng Zhang, Lin Chen, Fanghao Shao, Bo Xue, Yunchong Song, Zhenjie Yang, Ganqu Cui, Ning Ding, Jianfeng Gao, Xiaodong Liu, Bowen Zhou, Hongyuan Mei, and Zhouhan Lin. FlowRL: Matching reward distributions for LLM reasoning. *arXiv preprint arXiv:2509.15207*, 2025.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

APPENDIX CONTENTS

A Extended Theoretical Derivations and Proofs 14
A.1 Analytical Derivation of the Optimal Policy 14
A.2 Connection Between Squared-Residual Updates and Reverse KL 15
A.3 Density Estimation for ReFormFlow 16
B Experiment Details 18
B.1 Training Efficiency and Hardware Configuration 18
B.2 Experimental Setup 19
C Limitations and Future Work 21
D Additional Results 22

A EXTENDED THEORETICAL DERIVATIONS AND PROOFS

A.1 ANALYTICAL DERIVATION OF THE OPTIMAL POLICY

We recall the reward-tilted optimum of KL-regularized reward maximization Peters et al. (2010); Levine (2018), using the terminal-sample notation of the main text. For a fixed prompt c, reference density $p_{\mathrm{ref}}(\cdot \mid c)$, reward R, and $\gamma > 0$, consider

$$J_c(p) = \mathbb{E}_{x_1 \sim p(\cdot|c)}[R(c, x_1)] - \gamma D_{\mathrm{KL}}\big(p(\cdot \mid c) \| p_{\mathrm{ref}}(\cdot \mid c)\big). \tag{13}$$

We maximize $J_c(p)$ over densities absolutely continuous with respect to the reference, subject to $p(x_1 \mid c) \geq 0$ and $\int p(x_1 \mid c)\, \mathrm{d}x_1 = 1$. Introducing a multiplier $\lambda(c)$ for normalization gives the Lagrangian functional

$$\begin{aligned} \mathcal{F}_c[p, \lambda] = & \int p(x_1 \mid c) R(c, x_1)\, \mathrm{d}x_1 \\ & - \gamma \int p(x_1 \mid c) \log \frac{p(x_1 \mid c)}{p_{\mathrm{ref}}(x_1 \mid c)}\, \mathrm{d}x_1 \\ & + \lambda(c) \left(1 - \int p(x_1 \mid c)\, \mathrm{d}x_1\right). \end{aligned} \tag{14}$$

On the support of the reference, taking the functional derivative with respect to a positive density and setting it to zero yields

$$\frac{\delta \mathcal{F}_c}{\delta p(x_1 \mid c)} = R(c, x_1) - \gamma \left(\log \frac{p(x_1 \mid c)}{p_{\mathrm{ref}}(x_1 \mid c)} + 1\right) - \lambda(c) = 0. \tag{15}$$

Solving this stationarity condition gives the candidate density

$$\log \frac{p^*(x_1 \mid c)}{p_{\mathrm{ref}}(x_1 \mid c)} = \frac{R(c, x_1)}{\gamma} - 1 - \frac{\lambda(c)}{\gamma}, \tag{16}$$

$$p^*(x_1 \mid c) = p_{\mathrm{ref}}(x_1 \mid c) \exp\left(\frac{R(c, x_1)}{\gamma}\right) \exp\left(-1 - \frac{\lambda(c)}{\gamma}\right). \tag{17}$$

The final factor is independent of x_1. Enforcing normalization determines its inverse as

$$Z(c) = \exp\left(1 + \frac{\lambda(c)}{\gamma}\right) = \int p_{\mathrm{ref}}(x_1 \mid c) \exp\left(\frac{R(c, x_1)}{\gamma}\right) \mathrm{d}x_1. \tag{18}$$

Provided $0 < Z(c) < \infty$, the normalized candidate is therefore

$$p^*(x_1 \mid c) = \frac{p_{\mathrm{ref}}(x_1 \mid c) \exp(R(c, x_1)/\gamma)}{Z(c)}. \tag{19}$$

To verify global optimality, assume that the reward expectation and KL terms are finite for the densities considered, including p^*. Substitution into the reverse KL yields

$$D_{\mathrm{KL}}(p \| p^*) = D_{\mathrm{KL}}(p \| p_{\mathrm{ref}}) - \frac{1}{\gamma} \mathbb{E}_p[R(c, x_1)] + \log Z(c), \tag{20}$$

$$J_c(p) = \gamma \log Z(c) - \gamma D_{\mathrm{KL}}(p \| p^*). \tag{21}$$

Non-negativity of KL establishes p^* as the maximizing distribution. Restricting p to a model family instead gives reverse-KL minimization within that family. Averaging over a fixed prompt distribution preserves this identity and does not require an infinite offline dataset. Related reward-tilted targets are used in Adjoint Matching and DMPO Domingo i Enrich et al. (2025); Li et al. (2025a).

A.2 Connection Between Squared-Residual Updates and Reverse KL

The expected-gradient connection between on-policy Trajectory Balance and reverse KL is established in prior work (Whitammer et al., 2023, Proposition 1, Eq. (8)). For completeness, we restate this connection for terminal densities in our notation, making the sampling and differentiation conditions explicit. The squared residual and learned normalization follow TB Whitammer et al. (2022); a related formulation is used for LLM reasoning in FlowRL Zhu et al. (2025).

Setup and differentiation convention. Fix c and suppress it below. Assume that p_θ is positive and differentiable on a parameter-independent support, that p_{ref} is positive there, and that the expectations below are finite and permit differentiation under the integral. The reference, reward, and γ are fixed. Let $\ell_\phi = \log Z_\phi(c)$ be independent of x_1 and of θ when taking the policy gradient. Define

$$\delta^*_\theta(x_1) = \log \frac{p_\theta(x_1)}{p^*(x_1)}, \qquad \delta_{\theta,\phi}(x_1) = \delta^*_\theta(x_1) + \ell_\phi - \log Z(c). \tag{22}$$

For a current sampling policy $p_{\bar\theta}$, the squared loss is

$$\mathcal{L}_{\mathrm{sq}}(\theta, \phi; \bar\theta) = \mathbb{E}_{x_1 \sim p_{\bar\theta}}[\delta_{\theta,\phi}(x_1)^2]. \tag{23}$$

We hold $\bar\theta$ and the sampled endpoints fixed during differentiation, then evaluate the derivative at $\theta = \bar\theta$. This is the on-policy semi-gradient convention: gradients flow through the evaluated log-density, not through sample generation.

Reverse-KL gradient. Write $s_\theta(x_1) = \nabla_\theta \log p_\theta(x_1)$ for the parameter score, not the spatial score $\nabla_{x_1} \log p_\theta$. Normalization gives

$$\mathbb{E}_{p_\theta}[s_\theta] = \int \nabla_\theta p_\theta(x_1)\, \mathrm{d}x_1 = \nabla_\theta 1 = 0. \tag{24}$$

Since p^* is fixed, $\nabla_\theta \delta^*_\theta = s_\theta$, so

$$\begin{aligned} \nabla_\theta D_{\mathrm{KL}}(p_\theta \| p^*) &= \nabla_\theta \mathbb{E}_{p_\theta}[\delta^*_\theta] \\ &= \mathbb{E}_{p_\theta}[\delta^*_\theta s_\theta] + \mathbb{E}_{p_\theta}[s_\theta] = \mathbb{E}_{p_\theta}[\delta^*_\theta s_\theta]. \end{aligned} \tag{25}$$

Squared-residual gradient. With the sampling distribution fixed, $\nabla_\theta \delta_{\theta,\phi} = s_\theta$. Consequently,

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{sq}}(\theta, \phi; \bar\theta)\big|_{\theta=\bar\theta} &= 2\mathbb{E}_{p_{\bar\theta}}[\delta_{\bar\theta,\phi} s_{\bar\theta}] \\ &= 2\mathbb{E}_{p_{\bar\theta}}[\delta^*_{\bar\theta} s_{\bar\theta}] + 2(\ell_\phi - \log Z(c))\mathbb{E}_{p_{\bar\theta}}[s_{\bar\theta}] \\ &= 2\, \nabla_\theta D_{\mathrm{KL}}(p_\theta \| p^*)|_{\theta=\bar\theta}\,. \end{aligned} \tag{26}$$

The normalization offset vanishes by Eq. 24, regardless of its current value. Thus it can change the stochastic gradient variance without changing the expected policy gradient; arbitrary offsets are not guaranteed to reduce variance. The identity concerns the θ update, not the joint (θ, ϕ) gradient of a KL objective.

Scope of the connection. The result is an equality of expected gradients up to a factor of two, not an equality of scalar objectives. Differentiating the sampling distribution as well would instead give

$$\nabla_\theta \mathbb{E}_{p_\theta}[\delta^2_{\theta,\phi}] = 2\nabla_\theta D_{\mathrm{KL}}(p_\theta \| p^*) + \mathbb{E}_{p_\theta}[\delta^2_{\theta,\phi} s_\theta]. \tag{27}$$

Similarly, uncorrected off-policy sampling does not generally satisfy Eq. 26. The derivation assumes exact terminal log-densities and a fixed reward and temperature during the update. Replacing the log-density ratio by a velocity-regression estimator yields a practical surrogate whose approximation error is separate from this identity.

A.3 Density Estimation for ReFormFlow

Starting from the terminal-sample regression in Eq. 23, we next describe density evaluation and its path-space approximation. The time convention follows Section 3.1.

Deterministic Density Evolution. In the standard Flow Matching framework, the diffusion coefficient $g_t = 0$, degenerating the SDE into a deterministic ODE: $\frac{\mathrm{d}x_t}{\mathrm{d}t} = v_\theta(x_t, c, t)$. Accordingly, the Fokker-Planck equation governing the probability density $p_\theta(x_t, t)$ simplifies to the continuity equation:

$$\frac{\partial p_\theta(x_t,t)}{\partial t} + \nabla_{x_t} \cdot (p_\theta(x_t,t) v_\theta(x_t,c,t)) = 0.$$

Applying the vector calculus identity $\nabla \cdot (\phi \mathbf{A}) = \phi(\nabla \cdot \mathbf{A}) + \mathbf{A} \cdot \nabla\phi$, this expands to:

$$\frac{\partial p_\theta(x_t,t)}{\partial t} + p_\theta(x_t,t)(\nabla_{x_t} \cdot v_\theta) + v_\theta \cdot \nabla_{x_t} p_\theta(x_t,t) = 0.$$

Using the chain rule, the total derivative of the density along the flow is $\frac{\mathrm{d}p_\theta(x_t,t)}{\mathrm{d}t} = \frac{\partial p_\theta(x_t,t)}{\partial t} + \nabla_{x_t} p_\theta(x_t,t) \cdot \frac{\mathrm{d}x_t}{\mathrm{d}t}$. Substituting $\frac{\mathrm{d}x_t}{\mathrm{d}t} = v_\theta$ yields:

$$\frac{\mathrm{d}p_\theta(x_t,t)}{\mathrm{d}t} = -p_\theta(x_t,t)\mathrm{Tr}(\nabla_{x_t} v_\theta(x_t,c,t)).$$

Applying the logarithmic derivative property gives $\frac{\mathrm{d}\log p_\theta(x_t,t)}{\mathrm{d}t} = -\mathrm{Tr}(\nabla_{x_t} v_\theta(x_t,c,t))$. Integrating over time from $t = 0$ to $t = 1$ provides the exact terminal log-likelihood:

$$\log p_\theta(x_1|c) = \log p_0(x_0) - \int_0^1 \mathrm{Tr}(\nabla_{x_t} v_\theta(x_t,c,t))\mathrm{d}t.$$

Auxiliary SDE and the Time Convention. Direct Jacobian-trace evaluation is costly in high-dimensional latent spaces. Shared-diffusion auxiliary processes motivate an alternative regression construction, since distinct deterministic ODE path measures need not admit a density ratio. We first specify the time convention underlying the auxiliary drift.

For the independent Gaussian interpolation $x_t = (1-t)x_0 + tx_1$, let $v_t(x,c) = \mathbb{E}[x_1 - x_0 \mid x_t = x, c]$ be its exact marginal velocity. For $0 < t < 1$, differentiation of the Gaussian conditional density gives

$$\nabla_x \log p_t(x \mid c) = -\frac{\mathbb{E}[x_0 \mid x_t = x, c]}{1-t} = \frac{t v_t(x,c) - x}{1-t}.$$

The Fokker–Planck equation preserves these interpolation marginals when the SDE drift is $v_t + (\sigma_t^2/2)\nabla_x \log p_t$. Replacing v_t by v_θ gives the auxiliary parameterization in Eq. 3, with $\sigma_t = g_t$. For an approximate network, this substitution does not establish equality between the auxiliary SDE and model ODE marginals. The formulas below are local to $0 < t < 1$; a full-interval path construction additionally requires integrable coefficients and valid endpoint limits. We use a positive diffusion coefficient, not a proved zero-diffusion limit.

Local Gaussian Transition Calculation. For a prescribed target field u_t, define the effective auxiliary drifts

$$\hat{v}_\theta = v_\theta + \frac{\sigma_t^2}{2(1-t)}(t v_\theta - x_t), \qquad \hat{u}_t = u_t + \frac{\sigma_t^2}{2(1-t)}(t u_t - x_t).$$

At a fixed current state, let $P_{\theta,t}$ and Q_t denote the Euler–Maruyama transition densities with means $x_t + \hat{v}_\theta \Delta t$ and $x_t + \hat{u}_t \Delta t$, respectively, and shared covariance $\sigma_t^2 \Delta t I$. These local kernels are distinct from the endpoint-conditioned variational distribution q below. Writing $\Delta x = x_{t+\Delta t} - x_t$, their log-ratio is

$$\begin{aligned}\log \frac{P_{\theta,t}}{Q_t} &= \frac{\|\Delta x - \hat{u}_t \Delta t\|^2 - \|\Delta x - \hat{v}_\theta \Delta t\|^2}{2\sigma_t^2 \Delta t} \\ &= \frac{\Delta x^T(\hat{v}_\theta - \hat{u}_t)}{\sigma_t^2} + \frac{\Delta t}{2\sigma_t^2}(\|\hat{u}_t\|^2 - \|\hat{v}_\theta\|^2).\end{aligned}$$

Under $Q_t(\cdot \mid x_t)$, $\mathbb{E}[\Delta x \mid x_t] = \hat{u}_t \Delta t$. Hence

$$\mathbb{E}_{Q_t(\cdot|x_t)}\left[\log \frac{P_{\theta,t}}{Q_t}\right] = -\frac{\Delta t}{2\sigma_t^2}\|\hat{v}_\theta - \hat{u}_t\|^2$$
$$= -\lambda_t \|v_\theta - u_t\|^2 \Delta t, \qquad \lambda_t = \frac{1}{2\sigma_t^2}\left(1 + \frac{\sigma_t^2 t}{2(1-t)}\right)^2.$$

The same calculation applies to the reference velocity. Subtracting the two local expected log-ratios gives $\lambda_t(\|v_{\text{ref}} - u_t\|^2 - \|v_\theta - u_t\|^2)\Delta t$.

A continuous-time Girsanov change of measure Oksendal (2013) additionally requires well-posed, non-explosive auxiliary SDEs with a shared positive diffusion coefficient and compatible initial laws. Novikov's condition on the diffusion-normalized drift difference under the target path measure is sufficient for the stochastic exponential to define a valid change of measure; differing initial laws also contribute their log-density ratio. These conditions do not identify the auxiliary terminal marginal with the ODE density.

For endpoint-conditioned paths, we use the local quadratic form to motivate the approximation

$$\mathbb{E}_{q(\cdot|x_1,c)}\left[\log \frac{P_\theta(z, x_1 \mid c)}{P_{\text{ref}}(z, x_1 \mid c)}\right] \approx \mathbb{E}_{q(\cdot|x_1,c)}\left[\int_0^1 \lambda_t \left(\|v_{\text{ref}} - u_t\|^2 - \|v_\theta - u_t\|^2\right) \mathrm{d}t\right],$$

where z collects the nonterminal path variables and the joint models are specified below. Conditioning on the endpoint generally changes the transition law, so the preceding current-state conditional expectation does not prove this approximation. Boundary terms and the conditional path construction remain to be established. In particular, using $u_t = x_1 - x_0$ on re-noised linear interpolants is a regression construction, not an identification of that target with a proved auxiliary bridge drift.

Posterior Gap and the Density-Ratio Approximation. Fix c and x_1, and let z collect the auxiliary trajectory variables other than the fixed endpoint. For $j \in \{\theta, \text{ref}\}$, consider a joint path model $P_j(z, x_1 \mid c)$ whose terminal marginal is $p_j(x_1 \mid c)$. Let $q(z \mid x_1, c)$ be the same variational distribution for both models; its dependence on c is suppressed where omitted. Assume compatible supports and finite expectations and KL divergences. Define

$$\mathcal{E}_j(x_1, c) = \mathbb{E}_q\left[\log \frac{P_j(z, x_1 \mid c)}{q(z \mid x_1, c)}\right], \tag{28}$$
$$G_j(x_1, c) = D_{\text{KL}}(q(z \mid x_1, c) \| P_j(z \mid x_1, c)). \tag{29}$$

Using $P_j(z \mid x_1, c) = P_j(z, x_1 \mid c)/p_j(x_1 \mid c)$ gives

$$G_j(x_1, c) = \mathbb{E}_q\left[\log \frac{q(z \mid x_1, c) p_j(x_1 \mid c)}{P_j(z, x_1 \mid c)}\right] = \log p_j(x_1 \mid c) - \mathcal{E}_j(x_1, c). \tag{30}$$

Thus, the gap is the KL divergence between the chosen conditional path distribution and the model's path posterior, not a terminal policy-to-reference KL. Subtracting the two identities yields

$$\log \frac{p_\theta(x_1 \mid c)}{p_{\text{ref}}(x_1 \mid c)} = \mathcal{E}_\theta(x_1, c) - \mathcal{E}_{\text{ref}}(x_1, c) + G_\theta(x_1, c) - G_{\text{ref}}(x_1, c). \tag{31}$$

Each G_j is nonnegative, but their difference has no fixed sign. If the two path models coincide at initialization, their gaps coincide; this does not control the gap difference after policy updates. For fixed q and reference model, G_{ref} is independent of θ but can still depend on the endpoint. We omit the intractable difference $G_\theta - G_{\text{ref}}$ in the practical surrogate, without assuming that it vanishes or remains small. Subtracting two ELBOs therefore supplies no general upper or lower bound on the terminal log-density ratio.

Two distinct approximations therefore enter the regression construction: omitting the posterior-gap difference in Eq. 31, and replacing the ELBO difference by the quadratic velocity expression above. The latter also requires a conditional path construction and correspondence with the ODE terminal marginals, neither of which follows from the local transition algebra. We provide no bound on either discrepancy; the posterior KL gap does not account for the regression approximation.

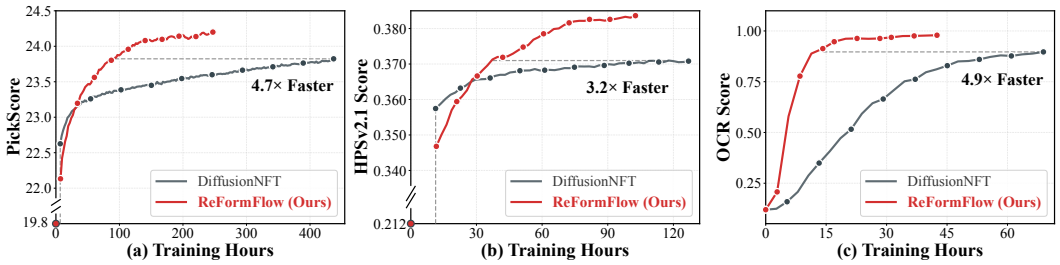


Figure 7: ReFormFlow versus DiffusionNFT on (a) PickScore, (b) HPSv2.1, and (c) OCR, against GPU-hours. Speedup annotations divide DiffusionNFT's cost to reach its highest recorded score by ReFormFlow's cost at the first evaluated checkpoint reaching that score.

Practical Regression Surrogate. The implemented loss uses ODE-generated endpoints and fresh Gaussian re-noising, rather than sampled auxiliary SDE trajectories. Let $\bar{\theta}$ denote the policy used to collect the current endpoints. Independently draw $x_0 \sim p_0$ for re-noising and set $x_{t_k} = t_k x_1 + (1 - t_k)x_0$ and $u_{t_k} = x_1 - x_0$. We aggregate the regression terms with the self-normalized weights w_{t_k} from Eq. 9. This empirical weighting is not a discretization of λ_t; the re-noising distribution is also not established as the endpoint-conditioned auxiliary path distribution used above. With shared velocity arguments (x_{t_k}, c, t_k) omitted, the practical objective is

$$\mathcal{L}_{\text{practical}}(\theta, \phi; \bar{\theta}) = \mathbb{E}_{\substack{x_1 \sim p_{\bar{\theta}}(\cdot|c) \\ x_0 \sim p_0}} \left[\left(\log Z_\phi(c) - \frac{\widetilde{R}(c, x_1)}{\gamma} + \frac{1}{K} \sum_{k=1}^{K} w_{t_k} \left[\|v_{\text{ref}} - u_{t_k}\|^2 - \|v_\theta - u_{t_k}\|^2 \right] \right)^2 \right].$$

This is Eq. 12, with $\widetilde{R}(c, x_1) = R(c, x_1) - \bar{R}_c$ and γ set to the standard deviation of raw rewards across the current buffer. The sampling distribution, sampled endpoints, centered rewards, and temperature are held fixed during differentiation; the denominator of w_{t_k} is detached as specified in Eq. 9. The policy and log-partition head are optimized jointly.

Finite-sample aggregation inside a squared residual is another distinction from regression on a conditional expectation. Even an unbiased, finite-variance estimator contributes its conditional variance to the expected squared residual, and that variance can depend on θ. We do not assert that the practical weighted re-noising average is an unbiased estimator of the auxiliary path integral, or that its update retains the exact-density reverse-KL gradient identity. Both training rollouts and inference use ODE sampling independently of the auxiliary SDE analysis.

B Experiment Details

B.1 Training Efficiency and Hardware Configuration

Our fine-tuning runs use a compute node with 8 NVIDIA H200 (141GB) GPUs and Distributed Data Parallel (DDP).

Figure 7 compares ReFormFlow and DiffusionNFT on the single-reward configurations in Section 5.2, including the general DiffusionNFT configuration for OCR. The horizontal axes in Figures 7 and 8, labeled "Training Hours," report sampling-and-training cost in GPU-hours. We compute this cost by multiplying each method's per-update time by the update count and eight GPUs, then dividing by 3,600. The ReFormFlow–DiffusionNFT comparison concerns 10-step single-reward training, not the 40-step multi-reward checkpoints in Table 2.

Computational cost. We measure per-update computational cost in aggregate GPU-hours across eight GPUs, amortizing sampling costs over the associated training updates and including an additional 70 seconds of reward computation per update for TempFlow-GRPO before GPU aggregation. ReFormFlow requires 0.2833, 0.1417, 0.2422, and 0.2333 GPU-hours per update on GenEval, OCR,

PickScore, and HPSv2.1, respectively. These costs are 11.3%, 6.9%, 6.4%, and 3.0% higher than those of DiffusionNFT. Compared with AWM, ReFormFlow reduces the per-update cost by 28.6% on GenEval and 25.0% on OCR, while increasing it by 4.0% on PickScore and 3.1% on HPSv2.1. Relative to TempFlow-GRPO, ReFormFlow reduces the cost by 46.8%, 80.5%, 54.1%, and 56.6% on the four tasks, respectively. These measurements establish a lower per-update cost than TempFlow-GRPO under the evaluated configurations, but not a uniform advantage over DiffusionNFT or AWM. End-to-end training efficiency must additionally account for the number of updates required to reach a comparable reward level.

Additional Baseline Comparisons. Figure 8 compares ReFormFlow with AWM Xue et al. (2025a), DiffusionNFT Zheng et al. (2025), and TempFlow-GRPO He et al. (2026) on single-reward optimization with PickScore, HPSv2.1, and OCR. We report reward trajectories against GPU-hours to compare their training efficiency.

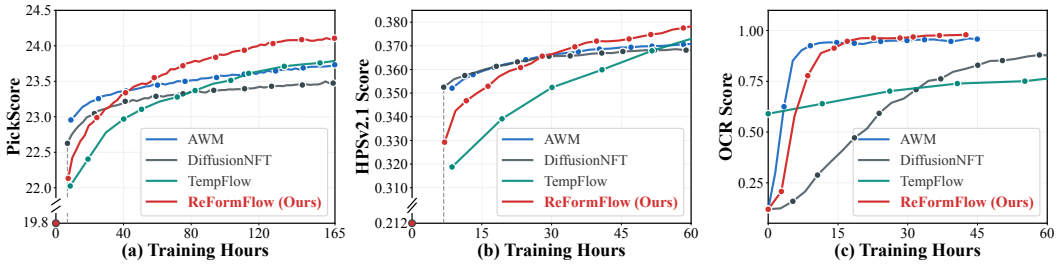


Figure 8: Additional single-reward comparisons with AWM, DiffusionNFT, and TempFlow-GRPO on (a) PickScore, (b) HPSv2.1, and (c) OCR. Horizontal axes report GPU-hours under the timing convention described above; displayed intervals and initial scores differ across methods.

B.2 Experimental Setup

Training Configurations. Each rollout collection contains 48 distinct prompts with 24 samples per prompt, yielding 1,152 generated endpoints. This count specifies the collected sample set, not an optimizer minibatch or an epoch. Table 3 lists the single-reward sampler and shared optimization settings. Multi-reward training uses 40-step rollouts, as specified below. We retain the generated endpoints and construct regression inputs by re-noising them with fresh Gaussian noise, rather than replaying the ODE rollout trajectory. The self-normalized weights in Eq. 9 act on these regression terms and are not part of the ODE solver's timestep schedule.

Table 3: Single-reward sampling and shared optimization settings for ReFormFlow.

Setting	Value
Single-reward rollout	10-step second-order DPM-Solver++
Evaluation	40-step first-order ODE solver
Training timestep fraction	0.99
Prompt groups per collection	48
Samples per prompt	24
Optimizer	AdamW, betas $(0.9, 0.999)$, weight decay 10^{-4}, optimizer epsilon 10^{-8}
Learning rate	Constant 3×10^{-4} for both the policy and $\log Z_\phi$ head
Gradient norm clipping	1.0
LoRA rank and scale	$r = 32$, $\alpha = 64$
LoRA initialization	Gaussian
LoRA placement	QKV and output projections in both streams of MMDiT joint attention

Conditional Log-Partition Head. The head parameterized by ϕ takes a 2,048-dimensional pooled prompt embedding formed from the 768-dimensional CLIP-L and 1,280-dimensional CLIP-G embeddings. It applies LayerNorm to this input, followed by three linear layers with dimensions

$2048 \to 1024 \to 1024 \to 1$. SiLU activations follow the first two linear layers, while the final layer outputs one unconstrained scalar $\log Z_\phi(c)$ per prompt. Thus, the head has two hidden layers, each of width 1,024. We optimize ϕ jointly with the policy's trainable parameters using the AdamW settings in Table 3; both use the same constant learning rate of 3×10^{-4}.

Mean Log-Partition Outputs. To inspect the numerical behavior of the head, Figure 9 shows its logged mean output over the same iteration windows as Figure 5: 0–1020 for PickScore, 0–440 for HPSv2.1, and 0–300 for OCR. The observed ranges are approximately $[-0.1451, 0.2771]$, $[-0.2451, 0.1830]$, and $[-0.6833, 0.2286]$, respectively. All displayed values are finite, including the initial transients; OCR exhibits a wider range than the two preference tasks. This diagnostic characterizes the logged mean only and does not establish the spread of individual prompt outputs or overall training stability.

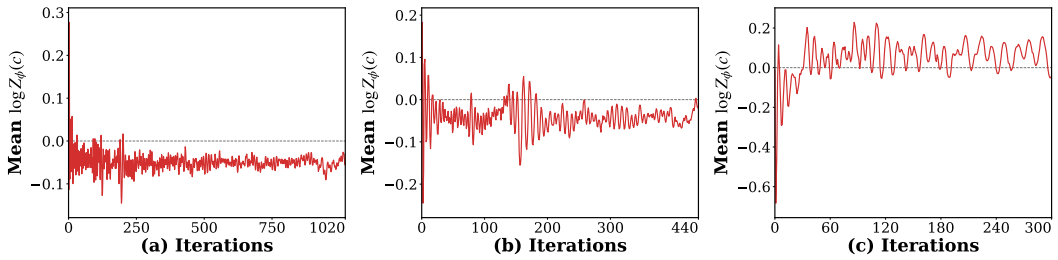


Figure 9: Mean outputs of the conditional log-partition head during (a) PickScore, (b) HPSv2.1, and (c) OCR training. Unsmoothed trajectories retain the full observed range, including initial transients, with task-specific vertical scales. Dashed lines mark zero.

Single-Reward Optimization. For GenEval, OCR, and PickScore, ReFormFlow uses the 10-step rollout configuration in Table 3. We subtract the current group's mean reward for each prompt and divide by the global standard deviation of raw rewards across the current on-policy buffer, as specified in Eq. 12. There is no additional multiplication by group size. The standard deviation is computed from raw rewards, rather than separately within each prompt group or from the full regression residual.

DiffusionNFT Baseline Configurations. The schedule η_i governs the lagging old-policy EMA in DiffusionNFT, not a ReFormFlow update. For the single-reward checkpoint comparisons in Table 1, GenEval and PickScore use $\beta = 1$ and $\eta_i = \min(0.001i, 0.5)$, where i is the baseline update index. The OCR checkpoint uses $\beta = 0.1$ and the delayed schedule

$$\eta_i = \begin{cases} 0, & i < 75, \\ \min(0.0075(i-75), 0.999)\,, & i \geq 75. \end{cases}$$

The OCR learning curves in Figures 3(c) and 7(c) instead use the general configuration, $\beta = 1$ and $\eta_i = \min(0.001i, 0.5)$. Thus, those curves and the task-specific OCR checkpoint row use different DiffusionNFT hyperparameters. ReFormFlow samples directly from its current policy and does not maintain this lagging old policy. Our single-reward DiffusionNFT reproductions use 10-step training rollouts and 40-step first-order ODE evaluation. The multi-reward DiffusionNFT row in Table 2 instead comes from our local evaluation of its official checkpoint, trained with 40-step rollouts for 1.7k iterations and marked by †. We evaluate this checkpoint using the same 40-step first-order ODE protocol as ReFormFlow.

Sensitivity to the DiffusionNFT OCR Configuration. Table 4 compares the two DiffusionNFT OCR configurations defined above and includes ReFormFlow as a reference. The general GenEval/PickScore configuration ($\beta = 1$ and $\eta_i = \min(0.001i, 0.5)$) obtains an OCR score of 0.89 after 520 updates, compared with 0.98 after 400 updates for the OCR-specific configuration ($\beta = 0.1$ and delayed EMA). This comparison indicates sensitivity to the jointly selected (β, η_i) configuration, but does not isolate either parameter's individual contribution or their interaction. The table reports the different iteration budgets and all evaluation metrics; the lower OCR score is not a uniform decline across metrics. ReFormFlow reaches an OCR score of 0.98 after 300 updates.

Table 4: Single-reward OCR comparison of DiffusionNFT configurations and ReFormFlow. The OCR-specific DiffusionNFT and ReFormFlow rows are reproduced from Table 1. All runs use 10-step training rollouts and 40-step ODE evaluation.

Method / configuration	#Iter	Task Metric			DrawBench Metric				
		GenEval	OCR	PickScore	CLIPScore	HPSv2.1	Aesthetic	ImgRwd	PickScore
DiffusionNFT (general)	0.52k	0.24	0.89	19.38	0.240	0.183	4.90	-0.92	20.27
DiffusionNFT (OCR-specific)	0.4k	0.20	0.98	19.15	0.233	0.168	4.79	-1.27	20.15
ReFormFlow	0.3k	0.23	0.98	19.26	0.238	0.173	5.08	-1.23	20.43

Multi-Reward Optimization. For joint objective alignment, we implement a multi-stage curriculum learning protocol analogous to DiffusionNFT. The curriculum is constructed upon three dataset-specific configurations: (1) a **3-reward** combination (PickScore, CLIPScore, and HPSv2.1) evaluated on the Pick-a-Pic dataset; (2) a **4-reward** combination (the previous three augmented with GenEval) on the GenEval dataset; and (3) an alternative **4-reward** combination (the initial three augmented with OCR) on the OCR dataset.

The ReFormFlow checkpoint in Table 2 is trained for about 1.5k iterations using 40-step second-order DPM-Solver++ rollouts. The stage order alternates preference-only Configuration (1) with GenEval-augmented Configuration (2), followed by OCR-augmented Configuration (3). The optimizer, LoRA, prompt-group size, and other settings follow Table 3; evaluation uses a 40-step first-order ODE solver. Within each stage, active rewards receive equal weights after dividing PickScore by 26. This fixed rescaling is not a bound on its numerical range.

We carry the trainable model parameters θ, frozen reference-policy parameters, optimizer momentum, and gradient scaler states across stages. Each multi-reward row reports all metrics from one checkpoint, rather than collecting the best scores from separate task-specific models. The reported ReFormFlow and official DiffusionNFT checkpoints use 40-step training rollouts, with their respective 1.5k and 1.7k iteration budgets listed in Table 2.

C Limitations and Future Work

While ReFormFlow establishes a theoretically grounded, on-policy optimization paradigm for continuous normalizing flows, several structural and empirical limitations remain.

Limitations. First, the strict on-policy nature of the algorithm necessitates continuous trajectory generation from the actively updating policy p_θ during training. Although a pure-ODE sampling scheme is employed to minimize per-step latency, the overall computational cost of continuous data collection remains higher than that of off-policy regression baselines that utilize static offline datasets. Second, optimization relies on external scalar rewards, so the learned policy may exploit reward-model errors or inherit their misalignment. Higher optimization rewards need not imply improvements in perceptual quality or out-of-domain performance. Third, the practical objective replaces endpoint density evaluation with self-normalized velocity regression on re-noised samples. The omitted posterior-gap difference, the conditional path approximation, and the empirical weighting are not bounded by our analysis. Consequently, the exact-density reverse-KL gradient identity does not establish an equivalent identity for the implemented update.

Future Work. Future research will focus on addressing these identified constraints. To mitigate the computational overhead of data collection, developing a hybrid optimization scheme that incorporates importance-sampled experience replay could theoretically reduce the required frequency of online generation. Furthermore, transitioning from explicit scalar reward optimization to direct preference-based learning mechanisms could decrease the susceptibility to reward hacking and improve alignment with human-centric objectives. Finally, constructing compatible conditional path measures and bounding the discrepancy between weighted regression and terminal density ratios would clarify when the practical update approximates reverse-KL optimization.

D ADDITIONAL RESULTS

We provide additional qualitative comparisons between the base model, FlowGRPO, and our proposed method in Figure 10, Figure 11, Figure 12 and Figure 13, corresponding to the GenEval, OCR, PickScore and DrawBench benchmarks, respectively.



Figure 10: Qualitative comparison on prompts sampled from the GenEval benchmark.

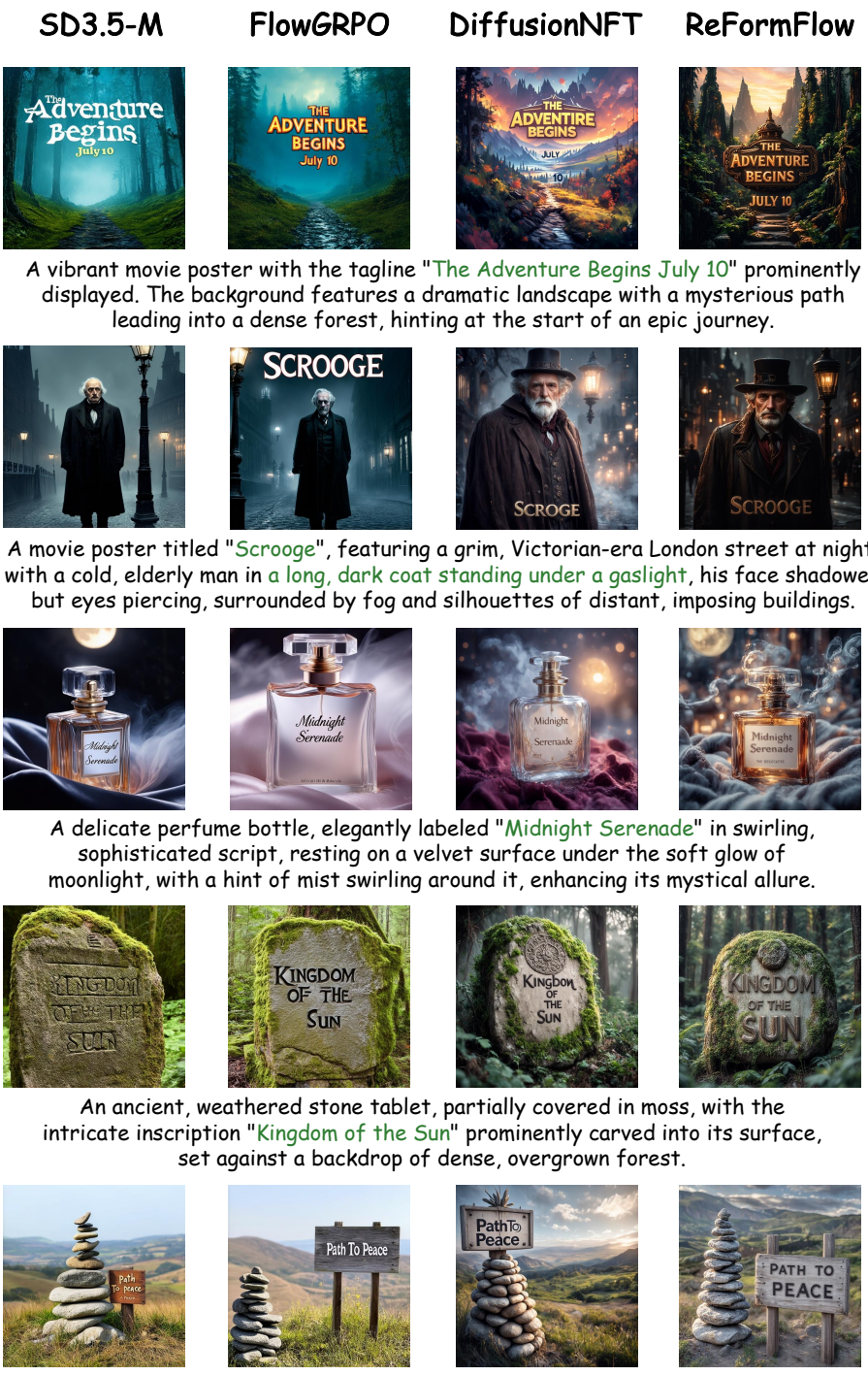


Figure 11: Qualitative comparison on prompts sampled from the OCR benchmark.

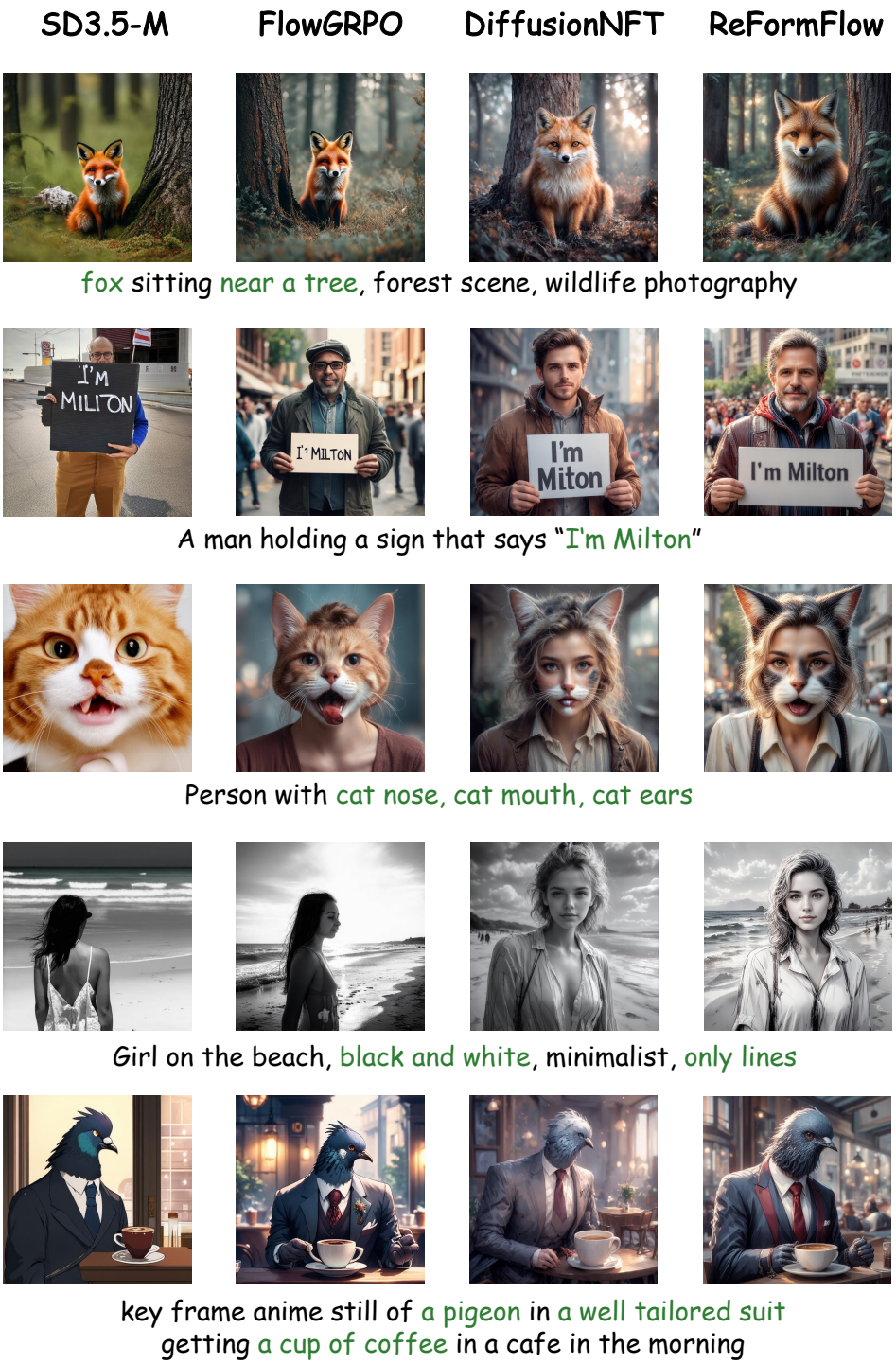


Figure 12: Qualitative comparison on prompts sampled from the PickScore benchmark.



Figure 13: Qualitative comparison on prompts sampled from the DrawBench benchmark.