FLOWSRL: STABLE AND SCALABLE REINFORCEMENT LEARNING FOR DIFFUSION AND FLOW MODELS

Anonymous authors
Paper under double-blind review

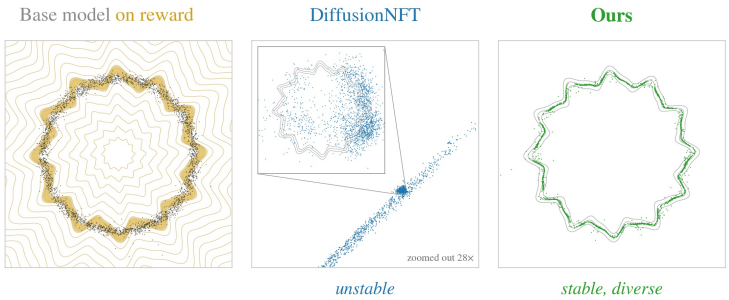


Figure 1: **Online RL in 2D.** A ring-shaped base model trained with a reward peaking on a thin zigzag. Prior methods are unstable, while FlowSRL is stable and can also be made diverse.

ABSTRACT

Online reinforcement learning has established itself as the standard approach for post-training diffusion and flow-based generative models with arbitrary reward signals. Despite their limited success, they often rely on various training tricks to remain stable, and inevitably collapse the diversity of the generated samples. In this work, we revisit both problems. To address the instability, we introduce *deviation clipping*, a simple trust region on the velocity update that stabilizes training. Anchoring the update to a learned flow-matching baseline rather than to the policy itself stabilizes it further and speeds up convergence. The resulting algorithm, *FlowSRL*, matches or outperforms prior diffusion RL methods on standard text-to-image benchmarks at significantly smaller batch sizes and roughly $2\times$ faster convergence. To address the diversity collapse, we study two complementary mechanisms, a passive one that restricts the policy to refining trajectories the pretrained model begins, and an active entropy reward bonus that incentivizes exploration beyond the base model's distribution. Finally, we apply FlowSRL to hard reward problem and show that, combined with search during the rollout stage, it is able to solve them reliably where the existing baselines without search struggled.

1 INTRODUCTION

Diffusion and flow-based generative models (Ho et al., 2020; Song et al., 2020; Lipman et al., 2022; Liu et al., 2022) have become the dominant approach to image and video generation, and reinforcement learning (RL) from reward models is being increasingly used to align them with human preferences (Xu et al., 2023) and other task-specific objectives (Black et al., 2024; Fan et al., 2023; Liu et al., 2025; Xue et al., 2025b; Zheng et al., 2026). Two recent methods, DiffusionNFT (Zheng et al., 2026) and Reinforce Adjoint Matching (RAM) (Bergmeister et al., 2026), avoid the reverse-process likelihoods used by diffusion policy-gradient methods. Instead, they directly update the velocity field using reward-weighted regression targets. This makes them simpler and cheaper, since rollouts can use any sampler, including fast deterministic solvers, and rewards climb in fewer training steps.

However in practice both methods require several additional mechanisms to combat training instabilities, including adaptively reweighting the loss, generating samples from a carefully tuned exponentially decayed copy of the policy, and very large training batch sizes (Zheng et al., 2026; Bergmeister et al., 2026). These add complexity and sensitive hyperparameters that must be tuned

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.

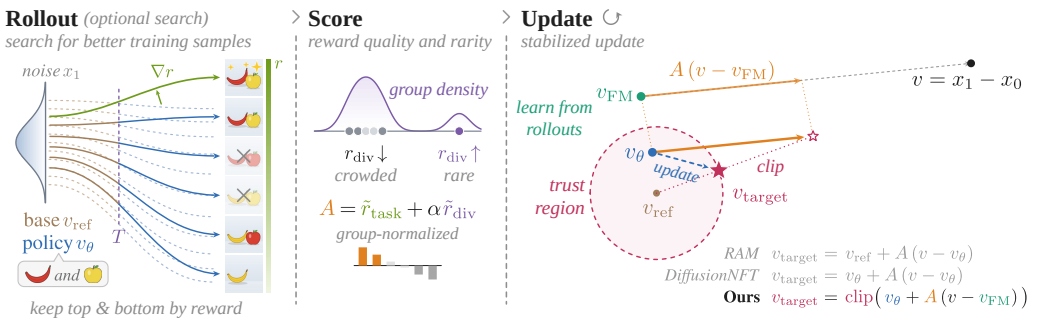


Figure 2: **Overview of FlowSRL.** Online flow RL splits into generating rollout samples, scoring them, and updating the model on the resulting data. We propose an update rule that is stable by design and needs no additional empirical techniques to keep training on track, which trains faster at smaller batch sizes, and faster still when combined with search during the rollout stage. To address diversity collapse, we propose two techniques that maintain and at times surpass the diversity of the base model, a passive one that leaves the trajectory above a refinement horizon T to the base model, and an active reward term r_{div} that favors rare samples within a group.

across rewards and models, and even with them in place we find both methods can become unstable under prolonged training. Furthermore, as in online RL more broadly, post-training can severely reduce sample diversity as the policy concentrates on a small number of high-reward modes. KL regularization is commonly used against both problems (Zheng et al., 2026; Xue et al., 2025a; Choi et al., 2026), but we find that no weight both preserves diversity and reaches a competitive average reward.

In this work, we revisit both problems. We begin with 2D toy problems, where the failure modes are immediately visible, and observe that prior reward-guided velocity matching methods are extremely unstable, with reward improving at first but collapsing early in training. KL regularization by itself does not resolve the underlying instability, so we propose *deviation clipping*, a hard per-sample trust region around the pretrained velocity field, as a simple fix. Deviation clipping stabilizes training across all rewards we consider and removes the need for most other stabilization tricks. Moving to images, we find a second improvement in the regression update itself. Prior methods define each sample's update direction relative to the policy's own velocity; replacing it with the flow matching velocity of the rollouts, estimated by a second LoRA adapter, speeds up and stabilizes training further, and we justify this choice theoretically. We call the resulting algorithm *FlowSRL*. On standard text-to-image benchmarks, it matches or outperforms prior likelihood-free diffusion RL methods while converging roughly $2\times$ faster and remaining stable at substantially smaller batch sizes.

We next turn to diversity collapse. Neither deviation clipping nor KL regularization incentivizes the policy to stay diverse, and on the 2D problem sweeping the clipping radius and the KL weight only recover the 2 extrema of insufficient reward optimization or concentration to a few high-reward modes, far from the exact KL-regularized optimum. We propose two mechanisms that mitigate diversity collapse in fundamentally different ways. The first is passive and retains most of the pretrained model's diversity. Inspired by prior work (Sorokin et al., 2025; Jena et al., 2025) that generates at high noise levels with the pretrained model, it trains the policy to refine only the rest of the trajectory, conditioned on the *refinement horizon* at which it takes over, turning the reward-diversity trade-off into a parameter that can be set at inference. The second is active and incentivizes exploration by adding an entropy bonus to the rewards within each rollout group , which has the potential to move beyond what the pretrained model covers and discover new modes.

Finally, the stability of FlowSRL at small update batch sizes changes how training compute can be allocated. Inspired by Li et al. (2026a), we find that training on only a small discriminative subset of a larger rollout group can nearly match training on the full group at a fraction of the update cost. This makes it practical to spend more compute on generating informative training samples rather than on larger optimization batches, including through test-time search during the rollout stage. Few-step *flow maps* (Kim et al., 2024; Boffi et al., 2024; Sabour et al., 2025b; Geng et al., 2026) are particularly well suited to this setting, as they enable generation in only a few steps and have already been used for reward-guided search (Sabour et al., 2025a; Huang et al., 2026a; Holderrieth et al., 2026). Recent work has also shown that these few-step models can be directly post-trained with RL (Potaptchik et al., 2026; Lee et al., 2026; Huang et al., 2026b; Mammadov et al., 2026; Li et al., 2026b). We

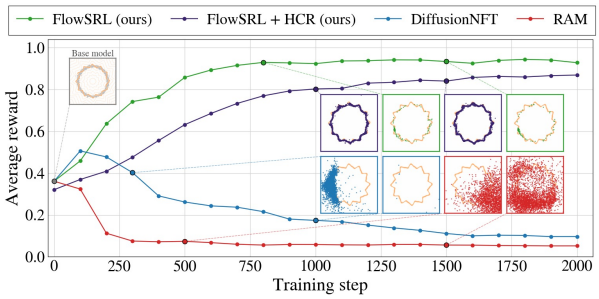


Figure 3: **Reward-guided velocity matching is unstable in 2D.** Training curves on a ring distribution whose reward is concentrated along a thin zigzag curve (orange). DiffusionNFT and RAM both collapse in this setting, while FlowSRL is stable and achieves high rewards.

therefore combine FlowSRL with flow maps and use reward-guided search to generate candidates that standard rollouts rarely reach. On hard rewards, search discovers high-reward samples where the same RL algorithm with standard rollouts gets stuck in a local minimum.

Contributions. *(i)* We show that prior reward-guided velocity matching methods become unstable under prolonged training and introduce *deviation clipping*, a per-sample trust region that stabilizes training with fewer auxiliary tricks. *(ii)* We introduce a learned flow-matching baseline for the update, improving stability, convergence speed, and small-batch training. *(iii)* We offer two complementary techniques to mitigate diversity collapse, one passive and one active. *(iv)* We combine FlowSRL with flow maps and show that rollout-time search helps solve hard rewards that standard sampling fails to reach.

2 Background

Flow matching. Flow matching (Lipman et al., 2022; Liu et al., 2022; Albergo et al., 2025) learns a velocity $\mathbf{v}_\theta(\mathbf{x}_t, t)$ on the linear path $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$ between data $\mathbf{x}_0 \sim p_{\text{data}}$ and noise $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $t \in [0,1]$, by minimizing $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\mathbf{x}_0,\mathbf{x}_1,t}\left[w(t)\left\|\mathbf{v}_\theta(\mathbf{x}_t,t) - (\mathbf{x}_1 - \mathbf{x}_0)\right\|_2^2\right]$, where $w(t)$ is a weighting function. Samples are generated by integrating the probability flow ODE (PF-ODE) $\mathrm{d}\mathbf{x}_t/\mathrm{d}t = \mathbf{v}_\theta(\mathbf{x}_t, t)$ from $t = 1$ to $t = 0$. Diffusion models (Ho et al., 2020; Song et al., 2020) learn the same velocity under a different parameterization, and we work in the flow matching framework from here on.

Online RL for flow models. Given G rollouts $\mathbf{x}_0^{1:G}$ per prompt $\mathbf{c}$ and rewards $r^{1:G}$, DiffusionNFT (Zheng et al., 2026) maps group-normalized rewards to optimality probabilities $r \in [0,1]$, re-noises each rollout along the forward process to $\mathbf{x}_t$ with target $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_0$, and minimizes $\mathcal{L}_{\text{NFT}}(\theta) = \mathbb{E}\Big[r\left\|\mathbf{v}_\theta^+ - \mathbf{v}\right\|_2^2 + (1-r)\left\|\mathbf{v}_\theta^- - \mathbf{v}\right\|_2^2\Big]$, where $\mathbf{v}_\theta^\pm = (1 \mp \beta)\,\mathbf{v}^{\text{old}} \pm \beta\,\mathbf{v}_\theta$ and $\mathbf{v}^{\text{old}}$ is the velocity of the rollout policy, an EMA of θ. Reinforce Adjoint Matching (RAM) (Bergmeister et al., 2026) arrives at a regression objective of the same kind (see Section 3.1) from stochastic optimal control. With the pretrained velocity $\mathbf{v}^{\text{ref}}$ and the normalized reward r of the sample, it regresses $\mathbf{v}_\theta$ onto $\text{sg}(\mathbf{v}^{\text{ref}} + r\,(\mathbf{v} - \mathbf{v}_\theta))$. Both methods avoid the reverse-process likelihoods of policy-gradient methods such as Flow-GRPO (Liu et al., 2025) altogether.

Training tricks. In practice, both methods incorporate additional training modifications beyond the objectives shown here. DiffusionNFT divides each sample's squared error by the stop-gradient of its own mean absolute error, a DMD-style adaptive weighting (Yin et al., 2024), and both methods draw rollouts from an EMA of the policy and use effective batches of around $10,000$ per update. In Section 3, we show what each of these does and replace them by a single trust region.

3 FlowSRL: Simple, Stable Online RL for Flow Models

Reward-guided velocity matching methods such as DiffusionNFT (Zheng et al., 2026) and RAM (Bergmeister et al., 2026) train quickly and without likelihoods, but can be unstable and quickly collapse their diversity. We first unify them under a single regression framework and reproduce both failures in 2D (Section 3.1), then stabilize it with deviation clipping and a learned flow matching baseline (Section 3.2), and finally address diversity collapse with one passive and one active mechanism (Section 3.3). The training algorithm is given in Section B.

3.1 Prior Reward-Guided Velocity Matching Methods Are Unstable

We begin by showing that DiffusionNFT (Zheng et al., 2026) and RAM (Bergmeister et al., 2026), despite being derived differently, are very similar and can be seen as special cases that regress the flow model to a reward-guided target. Recall from Section 2 that DiffusionNFT minimizes

$$\mathcal{L}_{\mathrm{NFT}}(\theta) = \mathbb{E}\left[r\,\|\mathbf{v}_\theta^+ - \mathbf{v}\|_2^2 + (1-r)\,\|\mathbf{v}_\theta^- - \mathbf{v}\|_2^2\right], \qquad \mathbf{v}_\theta^\pm = (1 \mp \beta)\mathbf{v}^{\mathrm{old}} \pm \beta\mathbf{v}_\theta, \tag{1}$$

where $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_0$ is the flow matching target. Expanding the two branches shows that its gradient is proportional to that of a plain regression[1],

$$\nabla_\theta \mathcal{L}_{\mathrm{NFT}}(\theta) \propto \nabla_\theta \mathbb{E}\left[\|\mathbf{v}_\theta - \mathrm{sg}\left(\mathbf{v}_{\mathrm{NFT}}^{\mathrm{target}}\right)\|_2^2\right], \quad \mathbf{v}_{\mathrm{NFT}}^{\mathrm{target}} = \mathbf{v}^{\mathrm{old}} + A(\mathbf{v} - \mathbf{v}^{\mathrm{old}}),\ A = \tfrac{2r-1}{\beta}. \tag{2}$$

RAM has nearly the same form. Writing its normalized reward as A, its objective is

$$\mathcal{L}_{\mathrm{RAM}}(\theta) = \mathbb{E}\left[\|\mathbf{v}_\theta - \mathrm{sg}(\mathbf{v}_{\mathrm{RAM}}^{\mathrm{target}})\|_2^2\right], \qquad \mathbf{v}_{\mathrm{RAM}}^{\mathrm{target}} = \mathbf{v}^{\mathrm{ref}} + A\left(\mathbf{v} - \mathbf{v}_\theta\right). \tag{3}$$

Therefore, both methods can be viewed as constructing a reward-guided target velocity

$$\mathbf{v}^{\mathrm{target}} = \mathbf{v}^{\mathrm{anchor}} + A(\mathbf{v} - \mathbf{v}^{\mathrm{baseline}}), \tag{4}$$

and regressing $\mathbf{v}_\theta$ toward it. DiffusionNFT uses the rollout velocity $\mathbf{v}^{\mathrm{old}}$ for both the anchor and the baseline, while RAM uses the pretrained velocity $\mathbf{v}^{\mathrm{ref}}$ as the anchor and the current velocity $\mathbf{v}_\theta$ as the baseline. When $\mathbf{v}^{\mathrm{anchor}} = \mathbf{v}^{\mathrm{baseline}}$, the target takes the same affine form as classifier-free guidance (CFG) (Ho & Salimans, 2022): positive A moves the target from the anchor toward or beyond the flow-matching target velocity $\mathbf{v}$, while negative A moves it away.

We next examine how these objectives behave on the 2D ring problem of Figure 3. Despite their success at image scale, both become unstable under prolonged training, with reward improving at first and then collapsing. Their sample distributions collapse visibly as well, concentrating in a small part of the high-reward region rather than covering it, with DiffusionNFT already on one side of the ring by step 300. Reward-guided velocity matching therefore has two distinct problems, training instability and loss of output diversity. We address stability first.

3.2 Stabilizing Training

KL regularization alone is not enough. A standard approach for stabilizing diffusion RL is to regularize the policy toward the pretrained model, commonly through a penalty $\lambda\|\mathbf{v}_\theta - \mathbf{v}^{\mathrm{ref}}\|_2^2$. To understand how far this alone can go, we sweep λ on the 2D problem of Figure 3. As shown in Figure G.4, weak regularization allows the model to drift too far and eventually becomes unstable, while strong regularization prevents this at the cost of significantly lower reward. Crucially, no setting achieves both stable training dynamics and effective reward optimization.

Deviation clipping. Let $\mathbf{v}^{\mathrm{target}}$ be a reward-guided target of the form in Equation (4) at $(\mathbf{x}_t, t)$, and let $\mathbf{v}^{\mathrm{ref}}$ denote the frozen pretrained velocity at the same point. We define *deviation clipping* by projecting the target into a ball of radius ρ around $\mathbf{v}^{\mathrm{ref}}$:

$$\mathbf{v}_{\mathrm{FlowSRL}}^{\mathrm{target}} = \mathbf{v}^{\mathrm{ref}} + \mathrm{clip}_\rho(\mathbf{v}^{\mathrm{target}} - \mathbf{v}^{\mathrm{ref}}), \qquad \mathrm{clip}_\rho(\Delta\mathbf{v}) = \frac{\Delta\mathbf{v}}{\max\{1, \|\Delta\mathbf{v}\|/\rho\}}, \tag{5}$$

where $\|\cdot\|$ denotes the RMS norm. Every regression target therefore stays within a distance ρ of the pretrained velocity, regardless of the sample's advantage, a hard trust region in velocity space that bounds large deviations directly instead of penalizing them. Unlike TRPO (Schulman et al., 2015), which constrains the KL divergence between policy distributions, our constraint acts on the velocity field and requires no likelihoods.

Because every target lies in this ball while the reference penalty pulls the policy inward, the policy itself can never leave a ball of its own. The theorem below makes this precise.

Theorem 1 (Invariant trust region, proof in Section C.1)**.** *Initialize the policy at the pretrained velocity, $\mathbf{v}_0 = \mathbf{v}^{\mathrm{ref}}$, and train it on deviation-clipped targets under the reference penalty of weight λ, so that each update moves the velocity at every input toward the mean of its clipped target. Then, under mild regularity conditions, for every training time $s \geq 0$,*

$$\sup_{\mathbf{x},t} \|\mathbf{v}_s(\mathbf{x},t) - \mathbf{v}^{\mathrm{ref}}(\mathbf{x},t)\| \;\leq\; \tfrac{\rho}{1+\lambda}\left(1 - e^{-(1+\lambda)s}\right) \;\leq\; \tfrac{\rho}{1+\lambda}.$$

[1] To highlight the reward-guided target interpretation, this derivation omits DiffusionNFT's adaptive loss weighting described in Section 2.

The bound holds throughout training rather than only at convergence, however large an advantage becomes, however inaccurate the learned baseline of Section 3.2 is, and whether or not the policy is its own flow matching field. It also shows that clipping does not make the reference penalty redundant, since clipping alone confines the policy to radius ρ and the penalty tightens that to $\rho/(1+\lambda)$. Under additional spatial regularity, this uniform bound on the velocity deviation also bounds the divergence between the policy's output distribution and the pretrained model's in both directions, at a rate linear in ρ for small radii (Section C.2).As shown in Figure G.4, this is enough to stabilize training without limiting reward optimization. With deviation clipping active, the extra stabilization tricks of prior methods, such as adaptive loss reweighting, soft updates of the rollout policy, and careful KL weight tuning, are no longer necessary. We found a constant radius $\rho = 0.7$ works well across most experiments.

A learned flow matching baseline. While deviation clipping stabilizes training, we find that the velocity update's baseline also has a large effect on training speed and stability at image scale. To understand why, we distinguish the velocity used to generate rollouts from the velocity learned by flow matching on those rollouts. The latter is the conditional mean $\mathbf{v}^{\mathrm{FM}} = \mathbb{E}[\mathbf{v} \mid \mathbf{x}_t, t]$. It need not equal the sampling velocity $\mathbf{v}_\theta$, since different velocity fields can generate the same endpoint distribution through different intermediate trajectories.

Using $\mathbf{v}^{\mathrm{FM}}$ as the baseline isolates the change that reward tilting induces in the flow matching field. Using the policy velocity instead adds a term proportional to $\mathbf{v}^{\mathrm{FM}} - \mathbf{v}_\theta$, so the update direction depends on the difference between two sampling fields when that difference does not change the desired output distribution (Section C.3).

We therefore learn $\mathbf{v}^{\mathrm{FM}}$ with a second adapter ψ, trained by standard flow matching on the same re-noised rollouts:

$$\mathcal{L}_{\mathrm{FMB}}(\psi) = \mathbb{E}_{\mathbf{x}_0 \sim \pi_\theta,\, \mathbf{x}_1, t}\left[\|\mathbf{v}_\psi(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|_2^2\right]. \tag{6}$$

The optimal minimizer for this problem is $\mathbf{v}^{\mathrm{FM}}$, and we approximate it with $\mathbf{v}_\psi$ and use as the update baseline. The adapter is a LoRA of the same rank as the policy, initialized from the pretrained model and updated once after each RL step on a subset of the same training samples. It is never used for sampling.

The full algorithm. Putting everything together, FlowSRL uses the reward-guided target $\mathbf{v}^{\mathrm{target}} = \mathbf{v}_\theta + A(\mathbf{v} - \mathbf{v}^{FM})$, applies deviation clipping $\mathbf{v}^{\mathrm{target}}_{\mathrm{FlowSRL}} = \mathbf{v}^{\mathrm{ref}} + \mathrm{clip}_\rho\big(\mathbf{v}^{\mathrm{target}} - \mathbf{v}^{\mathrm{ref}}\big)$ from Equation (5), and minimizes

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathrm{sg}\big(\mathbf{v}^{\mathrm{target}}_{\mathrm{FlowSRL}}\big)\|_2^2\right] + \lambda \mathbb{E}\left[\|\mathbf{v}_\theta - \mathbf{v}^{\mathrm{ref}}\|_2^2\right]. \tag{7}$$

Rollouts are sampled from the current policy, and we set $A = \mathrm{clip}(\kappa\tilde{r}, -1, 1)$, where $\tilde{r}$ is the group-normalized reward and $\kappa = 0.2$, following DiffusionNFT. The full training procedure is summarized in Algorithm 1. The FM baseline requires an additional optimization pass and introduces roughly a 10-15% overhead, however it massively increases convergence speed and stability making it an essential component of FlowSRL.

3.3 Mitigating Extreme Diversity Collapse

With training stable, we turn to diversity. Approaches to countering the mode collapse of Section 3.1 fall into two broad families. A *passive* approach preserves as much of the base model's diversity as possible, and ideally exposes a parameter that controls the diversity vs. reward trade-off at inference. An *active* approach instead incentivizes the policy to explore beyond what the base model already produces. Passive approaches are simpler but cannot move beyond the base model's distribution, whereas active ones can, at the cost of being harder to control.

Passive mechanisms. The obvious knob is the KL weight λ, together with the clip radius ρ, which plays a similar role. Both bound how far the policy can move from the reference, but neither discourages it from concentrating among the high-reward modes it can still reach, and neither is sufficient on its own (Section 4.2). A second passive mechanism follows from the structure of the generative process. Flow models generate coarse to fine, so a sample's mode is largely decided in the high-noise part of the trajectory and only refined in the low-noise part. Diversity can therefore be preserved by splitting the trajectory at a *refinement horizon* T, letting the frozen base model sample from $t = 1$ to T and the reward-tuned model refine from T to 0. Following Sorokin et al. (2025),

Table 1: **SD3.5M results**: training rewards on held-out prompts and image-quality metrics on DrawBench. Within each task, **bold**: best; underline: second best. †Numbers taken from Bergmeister et al. (2026).

Model	# Steps	Training Reward			Image Quality Metrics				
		GenEval	OCR	PickScore	Aesthetic	DeQA	ImgRwd	HPSv2	PickScore
SD3.5M		0.25	0.12	19.73	5.15	3.30	-0.55	0.26	20.51
+ CFG = 4.5		0.65	0.59	21.68	5.35	4.05	0.84	0.29	22.33
Compositional Image Generation									
Flow-GRPO†	> 5k	0.95			**5.25**	4.01	**1.03**	0.27	22.37
AWM†	300	0.83			5.14	3.75	0.67	0.24	22.04
DiffusionNFT	400	0.91			4.99	3.75	0.52	0.27	21.64
RAM	25	0.72			5.22	**4.28**	0.45	**0.28**	21.87
FlowSRL	400	**0.96**			5.21	3.94	0.37	0.27	**22.62**
Visual Text Rendering									
Flow-GRPO†	1.2k		0.92		**5.32**	**4.06**	**0.95**	**0.28**	**22.44**
AWM†	200		0.97		5.01	2.83	-0.85	0.18	20.56
DiffusionNFT	125		**0.99**		4.59	2.33	-1.02	0.25	20.39
RAM	50		0.97		4.45	3.36	-1.67	0.25	19.33
FlowSRL	400		**0.99**		5.01	3.53	-0.74	0.26	20.53
Human Preference Alignment									
Flow-GRPO†	> 3k			23.31	5.92	**4.22**	1.28	**0.32**	23.53
AWM†	1k			23.39	6.31	4.10	1.27	0.31	23.76
DiffusionNFT	400			23.31	6.24	4.05	1.28	0.29	23.62
RAM	400			23.47	6.07	4.21	1.30	0.29	23.72
FlowSRL	400			**23.94**	**6.38**	4.18	**1.31**	0.29	**24.05**

we call this *CombGen*, short for combined generation. It applies at inference to any reward-tuned checkpoint and trades reward against diversity as T is varied, but it is a mismatch, since the checkpoint it hands over to is asked to complete trajectories it was never trained to complete. We therefore train a model to refine over a horizon it is told about, which we call horizon-conditioned refinement, adapting Sorokin et al. (2025), who train a model to complete forward-noised real images under a differentiable reward, to the group-based RL setting of Section 3.2. Each group is built from a single base sample $\mathbf{x}_0^b$, re-noised D times to the horizon as $\mathbf{x}_T^i = (1-T)\,\mathbf{x}_0^b + T\,\boldsymbol{\epsilon}_i$ and denoised by the policy from T to 0. Advantages are computed within the group, and the update of Equation (7) is applied only at $t < T$. We draw a new T for every group and condition the policy on it through a zero-initialized embedding, so that one model can learn to switch at every horizon. Section A discusses how this differs from prior work. Because every anchor is a sample from the reference model, the distribution of states the policy trains on never moves, so the policy learns only how to refine a sample and never where to place mass, which is the compounding loop that collapses the methods of Section 3.1. Since horizon-conditioned refinement is trained across horizons and conditioned on them, T can be chosen at inference to select a point on the induced reward–diversity trade-off curve without retraining.

Active mechanisms. Both passive mechanisms are bounded by the base model, since neither the KL anchor nor the refinement horizon can place mass where the base model has none. An active mechanism instead changes the objective, so that the policy is rewarded for exploring. Diversity is a property of the distribution rather than of any individual sample, while the task reward evaluates samples independently. Motivated by Maximum-Entropy RL (Eysenbach & Levine, 2021), we reward samples according to how surprising they are relative to the other samples in their group, estimating that surprisal by kernel density estimation (KDE) as the most straightforward version of the idea. Concretely, given a group $\mathbf{x}_0^{1:G}$ and a feature extractor ϕ, we estimate each sample's surprisal under a KDE of the group $r_{\text{div}}^i = -\log\left(\frac{1}{G}\sum_{j=1}^{G} K_h(\phi(\mathbf{x}_0^i) - \phi(\mathbf{x}_0^j))\right)$, where K_h is a Gaussian kernel with bandwidth h. We group-normalize this surprisal independently of the task reward and combine the standardized values as $\tilde{r} = \tilde{r}_{\text{task}} + \alpha\,\tilde{r}_{\text{div}}$, where α controls the strength of the bonus, and separate normalization makes it scale-free of both the task reward and the data dimensionality. Before standardization, the group average of r_{div}^i is a KDE estimate of the entropy of $\phi(\mathbf{x}_0)$, so the bonus is a group-normalized approximation to maximum-entropy RL in feature space, raising the relative reward of samples in underrepresented regions and lowering it in crowded ones. The choice of ϕ depends on the task, which we discuss in Section 4.2. For details, see Section D. The two families are complementary rather than interchangeable, since the passive mechanisms retain the diversity the base model already has while the entropy bonus reaches regions it does not cover.

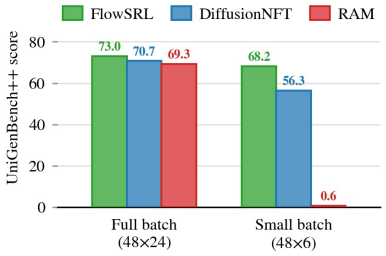


Figure 4: **UniGenBench++ overall score after** 500 **training steps**, evaluated without guidance. All methods roll out 48×24 samples per step; the small-batch bars train on only the 3 highest- and 3 lowest-reward samples of each group. The three methods are within 3.7 points of each other in the full batch setting, but only FlowSRL is competitive in the small batch one. For more details, see Table G.1. Training curves are visualized in Figure G.2.

4 Experiments

We post-train Stable Diffusion 3.5 Medium (Esser et al., 2024), a popular open-source 2B parameter pretrained flow model, on multiple standard text-to-image benchmarks, and compare against recent baselines including Flow-GRPO (Liu et al., 2025), DiffusionNFT (Zheng et al., 2026), and RAM (Bergmeister et al., 2026). We then test our two diversity mechanisms on both 2D toy data and text-to-image generation, and finally train on hard out-of-distribution rewards, where we replace standard rollouts with test-time search. For more details, see Section F.

4.1 Standard Text-to-Image Experiments

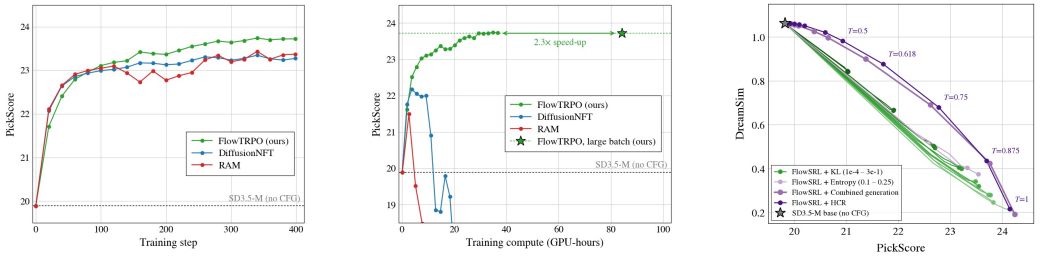
Following the evaluation protocol of prior work, we train LoRA adapters on three standard rewards, GenEval for compositional generation, OCR for visual text rendering, and PickScore for human preference. Figure 5a shows the PickScore training curves of FlowSRL, DiffusionNFT, and RAM at the original full batch of 48 prompts with 24 images each, re-noised 8 to 9 times depending on the method to compute the loss; Figure G.3 gives all three rewards and Section F the training parameters in full. Both prior methods are susceptible to crashing during prolonged training, with RAM crashing on both GenEval and OCR within 100 steps and DiffusionNFT on OCR within 200, whereas FlowSRL trains smoothly on all three rewards for the full 400 steps. For a fair comparison, all methods checkpoint every 25 steps and are evaluated at their final checkpoint, or at their best checkpoint before the crash if they crashed. Table 1 summarizes the results. Every method except Flow-GRPO converges quickly to a high reward, but FlowSRL still comes out slightly ahead, which we attribute to its stable training dynamics. Figure G.1 ablates the update baseline, comparing the learned flow-matching velocity $\mathbf{v}^{\mathrm{FM}}$ with the policy velocity $\mathbf{v}_\theta$ itself. Without the flow-matching baseline, training can become unstable and the reward does not climb as high.

On 16 NVIDIA H100 GPUs, each training step spends roughly 10 seconds sampling from the policy and 40 seconds updating the model. Inspired by Sol-RL (Li et al., 2026a), we therefore keep the rollout stage unchanged but update the model only on the 3 highest- and 3 lowest-reward samples of each group. This cuts the update time by $4\times$, and the step time by closer to $2\times$ once the small overhead of training the flow-matching baseline $\mathbf{v}_{\mathrm{FM}}$ is included. Figure 5b shows the training curves in this setting. DiffusionNFT and RAM both become even less stable and crash on all three benchmarks, whereas FlowSRL keeps training smoothly and matches its own full-batch results at almost half the compute.

We also evaluate FlowSRL, DiffusionNFT, and RAM on UniGenBench++ (Wang et al., 2025), using a combination of a VLM judge that measures text alignment and HPSv2 (Wu et al., 2023) that measures perceptual quality as the reward. As RAM crashed in the first 25 steps, we opt to evaluate it's first saved ckpts, and for the others we evaluate the final saved checkpoint. Figure 4 summarizes the results, and the full training curves are shown in Figure G.2. At the full batch FlowSRL again reaches the highest final reward. At the small batch, RAM crashes almost immediately, while DiffusionNFT climbs at first but starts to diverge slowly from around step 175.

4.2 Diversity Experiments

We start on the 2D ring from Figure 1, where the reward-tilted distribution can be sampled exactly by importance sampling. This distribution is the optimal solution to the KL-regularized RL objective, and we use it as our reference. We evaluate the three techniques of Section 3.3 in this setting, namely sweeping the KL weight λ and the clip radius ρ, training a horizon-conditioned refinement model,



(a) Full batch (48×24) (b) Small batch (48×6) (c) Reward versus diversity

Figure 5: **Text-to-image results on SD3.5-M with PickScore reward.** (a) At the full batch, all three methods train stably and FlowSRL ends highest. (b) At the small batch, DiffusionNFT and RAM collapse, whereas FlowSRL reaches its own full-batch reward (star) $2.3\times$ faster. GenEval and OCR versions can be seen in Figure G.3. (c) PickScore against DreamSim diversity, sweeping the KL weight λ (green), the entropy weight α (light purple), and the horizon T of CombGen from the FlowSRL checkpoint (purple) and of horizon-conditioned refinement (dark purple). Star: SD3.5-M without guidance.

and adding the entropy bonus to the objective, whose feature map ϕ is the identity in 2D. Figure 6 plots the results as reward versus diversity, where diversity is the angular entropy of the samples around the ring, and Figure D.1 shows samples from each technique. Every point is a separate training run; see Section D for details and Figure D.2 for the full sweep samples.

Sweeping ρ and λ alone leaves a large gap to the reference, and every run above a reward of 0.8 collapses onto a few segments of the zigzag. The entropy bonus lifts the frontier onto the reference, and in places slightly past it. In contrast, horizon-conditioned refinement is trained once and evaluated at any horizon T at inference, so it appears as a curve rather than a single point, and it holds almost all of the base model's diversity while still reaching a high reward. The samples show where the difference comes from. Under horizon-conditioned refinement the policy finds the high-reward regions within the ring but never moves beyond it, whereas the entropy bonus reaches some of the highest points of the zigzag, which lie outside the ring, at the cost of more outliers which lower its average reward.

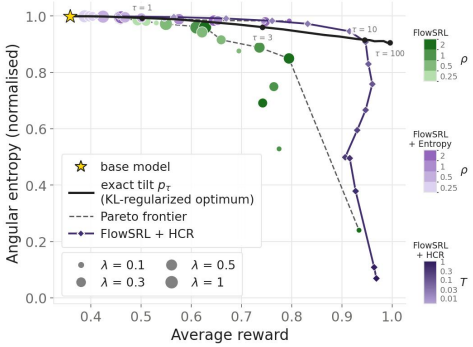


Figure 6: **Both mechanisms hold the reward-diversity trade-off that the dials give up.** Reward versus diversity on the 2D ring of Figure 3, sweeping the clip radius ρ (color) and KL weight λ (size) without (green) and with (purple) the entropy bonus, and the horizon T of horizon-conditioned refinement (dark purple), against the exact reward tilt (black). Star: the base model.

Next, we scale this study to text-to-image generation with the PickScore reward. We keep the default FlowSRL parameters of Table 1 and sweep the KL weight λ, both with and without the entropy bonus, whose feature map ϕ here extracts intermediate features from the base model at noise level $t = 0.5$. We also train a horizon-conditioned refinement model, drawing the horizon during training from a logit-normal distribution, $\text{logit}(T) \sim \mathcal{N}(\mu, \sigma^2)$. Following prior work (McAllister et al., 2026), diversity is measured as the mean pairwise distance between DreamSim features among samples from the same prompt; Section D.1 gives the full setup. The improvements here are less dramatic, with the entropy bonus lifting the Pareto frontier only slightly above the KL sweep alone (Figure 5c). horizon-conditioned refinement also scales to images, improving on CombGen from the final FlowSRL checkpoint across the sweep, though it adds hyperparameters that have to be tuned, namely the distribution the horizon is drawn from during training.

4.3 Hard-reward Experiments

Online RL can only improve the model where its rollouts place samples. When the high-reward region is outside the base model's distribution, such that no sample in a group reaches it, every advantage is zero and training never starts. Replacing the standard rollout by a reward-guided search can help, since search can find high-reward samples the policy would not have drawn on its own. We

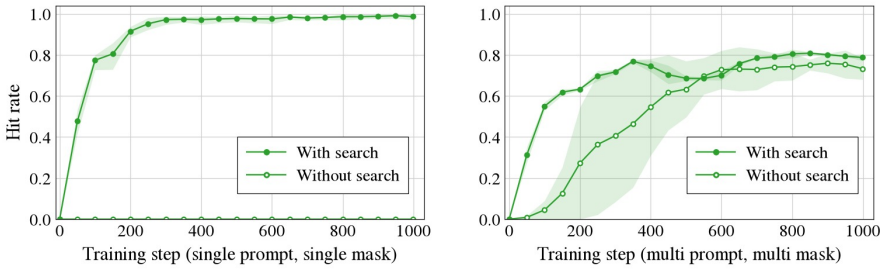


Figure 8: **Search in the rollout on hard layout rewards.** Hit rate against training step with and without search, over two seeds. When the reward is discovery bound (left), plain rollouts never find a positive sample and never start learning. When it is not (right), search converges faster to a similar hit rate.

use flow map guided search (Sabour et al., 2025a; Huang et al., 2026a), which uses few-step flow map (Boffi et al., 2024; Sabour et al., 2025b; Geng et al., 2026) generators and has been shown to solve hard rewards of this kind. We start from a flow map distilled from SD3.5-M following Sabour et al. (2025b) and optimize the policy and the flow map together, fine-tuning the flow map on the updating policy to ensure its jumps remain consistent with the velocity being updated. For details, see Section E. The reward we use asks for the scene to be confined to a masked region of an otherwise empty canvas, which is far outside what the base model produces.

Figure 8 reports the fraction of successful samples over training, which we call the *hit rate*. A sample counts as a success if the average brightness outside the mask is below 0.05 and a VLM judge verifies that the described scene can be seen inside it. On a single prompt with a single mask, standard rollouts darken the whole image until every sample is pure black, and the hit rate never rises above 0 across 1000 steps. With search in the rollout, training instead reaches a hit rate of 0.9 within 200 steps; Figure 7 shows qualitative samples. On paper, expanding the task to multiple prompts and multiple masks of different sizes should make it harder. Counterintuitively, it makes it easier, and even standard rollouts escape their local optimum and start learning. We believe this acts as a curriculum, where the easier instances, such as larger masks or prompts describing dimmer scenes, provide the signal needed to escape, and since the mask is described as a suffix appended to each prompt, what is learned on them transfers to the rest. Search still helps, reaching by step 100 the hit rate that standard rollouts only match by step 400, roughly $4\times$ sooner, and ending slightly higher. Figure G.5 shows samples from this setting.

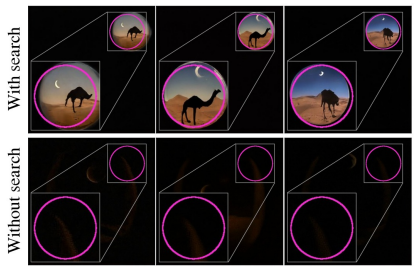


Figure 7: **The effect of search on hard rewards.** Online RL without search converges to a local optima by making all images full black. Search during the rollout stage can fix this. The multi-prompt setting is shown in Figure G.5.

5 Conclusions

We have presented FlowSRL, an online RL algorithm for flow models that trains the simple regression form of the DiffusionNFT objective as written, replacing the training tricks of prior methods by a single velocity-space trust region and anchoring the update to a learned flow-matching baseline. On standard text-to-image benchmarks, FlowSRL matches or outperforms prior methods while converging faster and remaining stable at much smaller batch sizes. We further showed that a kernel-density term in the reward, an entropy bonus applicable to any RL algorithm, counters the diversity collapse that plagues online RL, and that combining FlowSRL with flow map training enables search in the rollout stage, which solves hard rewards on which plain rollouts never find a positive sample.

AI USE STATEMENT

The authors used LLM-based tools for coding assistance and to suggest edits to the writing, including the theoretical results. The authors reviewed and verified all AI-assisted material and take full responsibility for the content, results, and claims of the manuscript.

ETHICS STATEMENT

Our experiments involved no human subjects. However, reward optimization may exploit biases or flaws in the reward model or judge, raising scores without improving quality and potentially reducing diversity. More capable generators could also be misused to create deceptive or harmful content, motivating independent evaluation and safeguards before deployment.

REPRODUCIBILITY STATEMENT

We provide pseudocode for our training algorithm in Section B. Experimental and implementation details are given in Section F. We will also release code for the reported experiments.

REFERENCES

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Anonymous. Scaling reinforcement learning for diffusion models via velocity matching. 2026.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 51304–51323, 2024.

Roberto Barceló, Cristóbal Alcázar, and Felipe Tobar. Avoiding mode collapse in diffusion models fine-tuned with reinforcement learning. *arXiv preprint arXiv:2410.08315*, 2024.

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling RL post-training of diffusion and flow-matching models. *arXiv preprint arXiv:2605.10759*, 2026.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38: 33346–33382, 2026.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

Daewon Chae, June Suk Choi, Jinkyu Kim, and Kimin Lee. DiffExp: Efficient exploration in reward fine-tuning for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15696–15703, 2025.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Jaemoo Choi, Yuchen Zhu, Wei Guo, Petr Molodyk, Bo Yuan, Jinbin Bai, Yi Xin, Molei Tao, and Yongxin Chen. Rethinking the design space of reinforcement learning for diffusion models: On the importance of likelihood estimation beyond loss design. *arXiv preprint arXiv:2602.04663*, 2026.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

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

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. *arXiv preprint arXiv:2103.06257*, 2021.

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*, 2023.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 2026.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps. *arXiv preprint arXiv:2602.05993*, 2026.

Jerry Y Huang, Justin Lin, Sheel Shah, Kartik Nair, and Nicholas M Boffi. How to guide your flow: Few-step alignment via flow map reward guidance. *arXiv preprint arXiv:2604.27147*, 2026a.

Yushi Huang, Xiangxin Zhou, Jun Zhang, Liefeng Bo, and Tianyu Pang. MeanFlowNFT: Bringing forward-process RL to average-velocity generators. *arXiv preprint arXiv:2607.15273*, 2026b.

Rohit Jena, Ali Taghibakhshi, Sahil Jain, Gerald Shen, Nima Tajbakhsh, and Arash Vahdat. Elucidating optimal reward-diversity tradeoffs in text-to-image diffusion models. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 232–242. IEEE, 2025.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations*, 2024.

Jaewoo Lee, Hyeongyu Kang, Dohyun Kim, Kyuil Sim, Woocheol Shin, Minsu Kim, Taeyoung Yun, Jeongjae Lee, Sanghyeok Choi, Tabitha Edith Lee, et al. Aligning few-step generative models by amortizing sample-based variational inference. *arXiv preprint arXiv:2605.26552*, 2026.

Junzhe Li, Yutao Cui, Tao Huang, Weijie Kong, Yiming Cheng, Chuxuan Zeng, Yinping Ma, Chun Fan, Miles Yang, Zhao Zhong, et al. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025.

Yitong Li, Junsong Chen, Shuchen Xue, Pengcuo Zeren, Siyuan Fu, Dinghao Yang, Yangyang Tang, Junjie Bai, Ping Luo, Song Han, and Enze Xie. Fp4 explore, bf16 train: Diffusion reinforcement learning via efficient rollout scaling. *arXiv preprint arXiv:2604.06916*, 2026a.

Zhiqi Li, Wen Zhang, and Bo Zhu. Flow-map grpo: Reinforcement learning for few-step flow-map generators via anchored stochastic composition. *arXiv preprint arXiv:2607.00535*, 2026b.

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, 2025. *URL https://arxiv. org/abs/2505.05470*, 2:5, 2025.

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

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

Abbas Mammadov, So Takao, Bohan Chen, Ricardo Baptista, Morteza Mardani, Yee Whye Teh, and Julius Berner. Variational flow maps: Make some noise for one-step conditional generation. *arXiv preprint arXiv:2603.07276*, 2026.

David McAllister, Miika Aittala, Tero Karras, Janne Hellsten, Angjoo Kanazawa, Timo Aila, and Samuli Laine. Finite difference flow optimization for RL post-training of text-to-image models. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Peter Potaptchik, Cheuk-Kit Lee, and Michael S Albergo. Tilt matching for scalable sampling and fine-tuning. *arXiv preprint arXiv:2512.21829*, 2025.

Peter Potaptchik, Samuel Howard, and Michael Samuel Albergo. Reward-aligning few-step flow models with integrated regularizers. In *ICML 2026 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2026.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

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 *International conference on machine learning*, pp. 8748–8763, 2021.

Amirmojtaba Sabour, Michael S. Albergo, Carles Domingo-Enrich, Nicholas M. Boffi, Sanja Fidler, Karsten Kreis, and Eric Vanden-Eijnden. Test-time scaling of diffusions with flow maps. *arXiv preprint arXiv:2511.22688*, 2025a.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. *arXiv preprint arXiv:2506.14603*, 2025b.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Dmitrii Sorokin, Maksim Nakhodnov, Andrey Kuznetsov, and Aibek Alanov. Imagerefl: Balancing quality and diversity in human-aligned diffusion models. *arXiv preprint arXiv:2505.22569*, 2025.

Maojiang Su, Jerry Yao-Chieh Hu, Sophia Pi, and Han Liu. On flow matching KL divergence. *arXiv preprint arXiv:2511.05480*, 2025.

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 *Conference on Computer Vision and Pattern Recognition*, 2024.

Yibin Wang, Zhimin Li, Yuhang Zang, Jiazi Bu, Yujie Zhou, Yi Xin, Junjun He, Chunyu Wang, Qinglin Lu, Cheng Jin, et al. Unigenbench++: A unified semantic evaluation benchmark for text-to-image generation. *arXiv preprint arXiv:2510.18701*, 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.

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 *Advances in Neural Information Processing Systems*, 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.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23117–23127. IEEE, 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. In *International Conference on Learning Representations*, 2026.

Appendix

A Related Work **15**

B Algorithms **15**

C Proofs and Derivations **16**
C.1 Theorem: Deviation clipping confines the policy to a ball around the reference . 16
C.2 Theorem: The trust region bounds the output distribution in both directions . . . 17
C.3 Derivation: The flow-matching baseline isolates the reward tilt 19

D Diversity Mechanisms **20**
D.1 Implementation details 20
D.2 The exact reward tilt as a reference 21
D.3 Group structure in horizon-conditioned refinement 21
D.4 Feature spaces for the entropy bonus 21

E Flow Maps and Search in the Rollout **22**

F Experimental Details **23**
F.1 Hyperparameters 23

G Additional Results and Ablations **25**
G.1 Text-to-image 25
G.2 2D toy problems 28
G.3 Hard-reward layouts 29

A Related Work

We briefly review prior work on reinforcement learning for generative models and on few-step generation.

RL for diffusion and flow models. Reward backpropagation (Xu et al., 2023; Prabhudesai et al., 2023; Clark et al., 2023) differentiates through the sampling chain and is limited to differentiable rewards. Reverse-process policy gradients (Black et al., 2024; Fan et al., 2023; Liu et al., 2025; Xue et al., 2025b; Li et al., 2025) discretize the reverse SDE so that step-wise likelihoods become tractable and apply PPO (Schulman et al., 2017) or GRPO (Shao et al., 2024); preference methods (Wallace et al., 2024) extend DPO via likelihood approximations. DiffusionNFT (Zheng et al., 2026) and RAM (Bergmeister et al., 2026) avoid likelihoods altogether by regressing reward-weighted targets on the forward process. Concurrent work by Anonymous (2026) focuses on scaling such objectives to video diffusion models. We keep the likelihood-free, regression-based objectives but stabilize and simplify them.

Stabilizing online RL. Trust-region methods limit policy updates through a KL constraint (Schulman et al., 2015) or clipped likelihood ratios (Schulman et al., 2017). Recent diffusion RL approaches use velocity-space reference penalties as KL regularization (Xue et al., 2025a; Choi et al., 2026), EMA rollout policies (Zheng et al., 2026; Bergmeister et al., 2026), and adaptive loss weighting inspired by DMD (Yin et al., 2024). Our deviation clipping keeps reward-guided velocity targets close to the pretrained velocity without requiring likelihoods, while additional KL regularization can tighten the bound on the learned field. In our experiments, clipping removes the need for EMA rollout updates, adaptive loss weighting, and careful KL-weight tuning.

Diversity in reward-tuned generators. Splitting sampling between the pretrained and the reward-tuned model is an established way to recover diversity at inference. ImageReFL (Sorokin et al., 2025) switches from the base model to a reward-tuned one partway through sampling and sweeps the switch point as a quality-diversity knob, while AIG (Jena et al., 2025) replaces the hard switch with a time-annealed blend of the two velocity fields. On the training side, HRF (Barceló et al., 2024) runs policy gradient from several forward-noised copies of a single clean sample with the loss restricted to the low-noise steps, also motivated by mode collapse, but samples from the tuned model alone at inference. Relative to Sorokin et al. (2025), the anchors of horizon-conditioned refinement are samples of the base model rather than real images, the update is group-relative and therefore does not require the reward to be differentiable, and the policy is trained over a distribution of refinement horizons and conditioned on which one it is given, rather than on a single fixed horizon. Unlike Barceló et al. (2024), the roll-in the policy is trained on is the one it is deployed with.

Flow maps and RL on few-step models. Consistency models (Song et al., 2023; Lu & Song, 2024) and flow maps (Kim et al., 2024; Boffi et al., 2024; Sabour et al., 2025b; Geng et al., 2026; Frans et al., 2024) compress sampling into a few evaluations. Recently, various methods have been proposed to finetuning flow maps on rewards (Li et al., 2026b; Huang et al., 2026b; Potaptchik et al., 2026; Mammadov et al., 2026). Similar to Potaptchik et al. (2026), we apply FlowSRL to the velocity of the flow map and use an additional flow map loss to learn larger jumps along the trajectories induces by this velocity. Beyond that, we also leverage the inexpensive clean-data look-aheads to facilitate reward-guided search during rollouts.

Reward guidance and test-time search. Diffusion sampling can be steered at inference time through search (Ma et al., 2025) or gradient guidance (Chung et al., 2022; Yu et al., 2023; Bansal et al., 2024). The latter typically uses differentiable objectives evaluated on predicted clean samples. Flow-map look-aheads support few-step reward guidance (Huang et al., 2026a) and reward-tilted search or importance-weighted sampling (Sabour et al., 2025a). Search can also shape policy learning: expert iteration trains a policy from tree-search results (Anthony et al., 2017), while Sol-RL selects reward-contrastive diffusion rollouts for policy optimization (Li et al., 2026a). Relatedly, Chae et al. (2025) diversify rollouts during reward fine-tuning by varying classifier-free guidance and prompt-phrase weights. We use guidance during the rollout to propose candidates, then rank guided and unguided rollouts by the full, potentially non-differentiable reward.

B Algorithms

Algorithm 1 FlowSRL.

1: **Input:** pretrained model θ, reward r_{task}, clip radius ρ, KL penalty λ, (optional) diversity weight α.
2: **Init:** initialize policy θ and flow matching baseline ψ from the pretrained model.
3: **for** each iteration **do**
4: Sample G rollouts for each of N prompts using the current policy θ.
5: Score with r_{task} and group-normalize to obtain $\tilde{r}$; optionally set $\tilde{r} \leftarrow \tilde{r}_{\text{task}} + \alpha \tilde{r}_{\text{div}}$ (Section 3.3); set $A = \text{clip}(0.2\tilde{r}, -1, 1)$.
6: Re-noise each rollout at K times $t \sim \mathcal{U}(0,1)$ with noises $\mathbf{x}_1$, and take one step on Equation (7).
7: Update ψ for one step on Equation (6) using a subset of the same samples.
8: **end for**

C Proofs and Derivations

This section contains proofs for the theoretical claims made in Section 3. Section C.1 proves the invariant trust region of Theorem 1, Section C.2 converts that trust region into a two-sided bound on the KL divergence between output distributions, Section C.3 shows why the learned flow-matching baseline isolates the reward tilt.

C.1 Theorem: Deviation clipping confines the policy to a ball around the reference

Throughout this section we fix a prompt and suppress it from the notation. We define $t \in [0,1]$ to be the *diffusion time* where $t = 0$ corresponds to the clean data distribution and $t = 1$ corresponds to the Gaussian prior. For this part, we'll assume that our updates are ideal and won't concern ourselves with the training dynamics associated with finite step gradient descent. Instead, we'll assume the functionals continuously change along the ideal path. We'll use $s \geq 0$ to denote the *training time*, which is a continuous analogue to the optimizer step count. Therefore, $\mathbf{v}_s(\mathbf{x}, t)$ will represent the model at training time s being evaluated at diffusion time t. For a field $\mathbf{f}$ we write $\|\mathbf{f}\|_\infty = \sup_{(\mathbf{x},t)} \|\mathbf{f}(\mathbf{x},t)\|$, with $\|\cdot\|$ denoting the RMS norm.

Theorem 2 (Restated from Theorem 1)**.** *We initialize the policy at the reference, so that $\mathbf{v}_0 = \mathbf{v}^{\text{ref}}$. At each training input $(\mathbf{x}, t)$, we form the raw target $Y_s = \mathbf{v}_s + A(\mathbf{v} - \mathbf{v}_\psi)$ of Equation (4) and clip it to $C_s = \mathbf{v}^{\text{ref}} + \text{clip}_\rho(Y_s - \mathbf{v}^{\text{ref}})$ as in Equation (5). We write $M_s = \mathbb{E}[C_s \mid \mathbf{x}, t]$ for the mean of that clipped target over the sample and group randomness. The ideal update then moves the field at every input along the negative functional gradient of 0.5 times the objective of Equation (7),*

$$\partial_s \mathbf{v}_s = M_s - \mathbf{v}_s - \lambda\big(\mathbf{v}_s - \mathbf{v}^{\text{ref}}\big). \tag{8}$$

Let $\kappa = 1 + \lambda$ and $R = \rho/(1+\lambda)$. Then, under mild regularity conditions, for every $s \geq 0$ we can show that

$$\big\|\mathbf{v}_s - \mathbf{v}^{\text{ref}}\big\|_\infty \leq R\big(1 - e^{-\kappa s}\big) \leq R.$$

In other words, the velocity field $\mathbf{v}_s$ is within a distance R of the reference model.

Proof. We'll fix an arbitrary input $(\mathbf{x}, t)$ and prove the bound there, after which the bound on the supremum follows immediately. To keep the notation light, we abbreviate $\delta_s = \mathbf{v}_s - \mathbf{v}^{\text{ref}}$ and $m_s = M_s - \mathbf{v}^{\text{ref}}$, both evaluated at that input.

We start by bounding m_s, which is the term driving the update. Clipping places every realization of C_s within ρ of $\mathbf{v}^{\text{ref}}$, and the RMS norm is convex. Conditional Jensen's inequality therefore gives

$$\|m_s\| = \big\|\mathbb{E}[C_s - \mathbf{v}^{\text{ref}} \mid \mathbf{x}, t]\big\| \leq \mathbb{E}\big[\|C_s - \mathbf{v}^{\text{ref}}\| \mid \mathbf{x}, t\big] \leq \rho. \tag{9}$$

We can now solve for the deviation itself. Since $\mathbf{v}^{\text{ref}}$ does not change with s, rewriting Equation (8) in terms of δ_s and m_s turns it into the forced linear equation $\partial_s \delta_s = m_s - \kappa \delta_s$. Its variation-of-constants solution is

$$\delta_s = e^{-\kappa s}\delta_0 + \int_0^s e^{-\kappa(s-u)} m_u \, \mathrm{d}u = \int_0^s e^{-\kappa(s-u)} m_u \, \mathrm{d}u,$$

where the second equality uses $\delta_0 = 0$, which is how we initialized the policy. Taking norms on both sides, we get

$$\|\delta_s\| = \left\| \int_0^s e^{-\kappa(s-u)} m_u \, \mathrm{d}u \right\| \leq \int_0^s e^{-\kappa(s-u)} \, \|m_u\| \, \mathrm{d}u \leq \rho \int_0^s e^{-\kappa(s-u)} \, \mathrm{d}u = \frac{\rho}{\kappa}\big(1 - e^{-\kappa s}\big),$$

where we pulled the positive scalar $e^{-\kappa(s-u)}$ out of the norm and then applied Equation (9). Since $R = \rho/\kappa$ by definition, this is exactly the bound we wanted. Taking the supremum over $(\mathbf{x}, t)$ then completes the proof. □

There are two things worth highlighting about this bound. First, it holds at every training time, rather than only at convergence. Second, it doesn't care about the size of the advantage, the accuracy of $\mathbf{v}_\psi$, or any relation between the policy and its own flow matching field. The clipping keeps things within a fixed radius.

C.2 Theorem: The trust region bounds the output distribution in both directions

Theorem 1 tells us that the policy's velocity field stays close to the reference's velocity field. What we actually care about, though, is whether the *samples* stay close, meaning the distribution of outputs we get after integrating a velocity field all the way from noise to data. This section shows that under certain regularity assumptions on the two field, an invariant bound on their differences leads to an upper bound on the KL divergence between their output distributions.

We'll need a bit of notation. Let D be the dimension of a sample, and let $\gamma = \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$ be the Gaussian prior that sampling starts from. We write p_s for the distribution of samples we get by integrating the policy's velocity $\mathbf{v}_s$ from $t = 1$ down to $t = 0$ starting from γ, and p^{ref} for the same thing with $\mathbf{v}^{\mathrm{ref}}$. These are the two distributions we want to compare.

We'll also need two constants that describe how smooth the velocity fields are. The first, L, bounds how quickly a velocity field can change as we move through space, so that $\|\nabla_{\mathbf{x}} \mathbf{v}\|_{\mathrm{op}} \leq L$ at every input. Intuitively, L controls how much the flow can stretch or squeeze a small neighbourhood of points. The second, H, bounds how quickly the divergence of a velocity field can change, so that $\|\nabla_{\mathbf{x}} \operatorname{div}_{\mathbf{x}} \mathbf{v}\|_2 \leq H$ at every input. We assume both bounds hold for both fields, at every diffusion time and every training time, and that the fields are smooth enough for the derivatives below to exist.

Theorem 3 (Two-sided KL bound)**.** *Suppose the two velocity fields never differ by more than ε, so that $\|\mathbf{v}_s(\mathbf{x}, t) - \mathbf{v}^{\mathrm{ref}}(\mathbf{x}, t)\|_2 \leq \varepsilon$ for every $\mathbf{x}$ and t. Then the KL divergence between the two output distributions is small in both directions,*

$$\max\big\{D_{\mathrm{KL}}(p_s \,\|\, p^{\mathrm{ref}}),\ D_{\mathrm{KL}}(p^{\mathrm{ref}} \,\|\, p_s)\big\} \leq e^L\big(2\sqrt{D} + H\big)\varepsilon + \tfrac{1}{2} e^{3L} \varepsilon^2. \tag{10}$$

In other words, a velocity field within ε of the reference produces a distribution within $O(\varepsilon)$ of the reference's.

Proof. For this proof we follow the continuity-equation argument of Albergo et al. (2025, Lemma 21), which Su et al. (2025) also use, and keep track of the velocity gap ε explicitly throughout.

The idea is the following. As we integrate from noise to data, the two distributions start out identical, since both start at γ, and then gradually drift apart. We'll write down the rate at which their KL divergence grows, show that this rate is proportional to the velocity gap ε, and then integrate the rate over the whole trajectory. Most of the work goes into bounding that rate, which requires controlling the *scores* of the two distributions, meaning the gradients of their log-densities.

We start by flipping the direction of time so that sampling runs forward. Set $\tau = 1 - t$, so that $\tau = 0$ is the noise end and $\tau = 1$ is the data end. In this new clock the two velocity fields are $b(\mathbf{x}, \tau) = -\mathbf{v}_s(\mathbf{x}, 1-\tau)$ and $c(\mathbf{x}, \tau) = -\mathbf{v}^{\mathrm{ref}}(\mathbf{x}, 1-\tau)$, where the minus sign comes from reversing time. We write $F_\tau(\mathbf{z})$ for the point that a noise sample $\mathbf{z}$ has reached at time τ when carried by b, and $G_\tau(\mathbf{z})$ for the same thing under c. These are the flow maps of the two fields. Pushing the prior through them gives the intermediate distributions $P_\tau = (F_\tau)_{\#}\gamma$ and $Q_\tau = (G_\tau)_{\#}\gamma$. By construction $P_0 = Q_0 = \gamma$ at the noise end, and $P_1 = p_s$ and $Q_1 = p^{\mathrm{ref}}$ at the data end.

Since the velocity fields are L-Lipschitz in space, two nearby points can drift apart, or together, by at most a factor of $e^{L\tau}$ over a time τ. Formally, Grönwall's inequality gives $\|J_{F_\tau}\|_{\mathrm{op}} \le e^{L\tau}$ and $\|J_{F_\tau}^{-1}\|_{\mathrm{op}} \le e^{L\tau}$ for the spatial Jacobian J, and likewise for G_τ. The same Lipschitz bound also guarantees that both flows are well-defined all the way to $\tau = 1$ and are invertible.

Next, we bound the score of each distribution along its own flow. We'll do this for a generic field a, which can be either b or c, with flow map Φ_τ and intermediate distribution r_τ. The instantaneous change-of-variables formula (Chen et al., 2018) tells us how the log-density changes along a trajectory,

$$\log r_\tau(\Phi_\tau(\mathbf{z})) = \log \gamma(\mathbf{z}) - \int_0^\tau \operatorname{div} a(\Phi_u(\mathbf{z}), u)\, \mathrm{d}u.$$

We want the gradient of the left-hand side with respect to the endpoint $\Phi_\tau(\mathbf{z})$, which is the score. Differentiating both sides with respect to $\mathbf{z}$, using $\nabla \log \gamma(\mathbf{z}) = -\mathbf{z}$ for the Gaussian, and then applying the chain rule to convert a gradient in $\mathbf{z}$ into a gradient in $\Phi_\tau(\mathbf{z})$, we get

$$\nabla \log r_\tau(\Phi_\tau(\mathbf{z})) = -J_{\Phi_\tau}(\mathbf{z})^{-\top}\mathbf{z} - \int_0^\tau \left(J_{\Phi_u}(\mathbf{z}) J_{\Phi_\tau}(\mathbf{z})^{-1}\right)^\top \nabla \operatorname{div} a(\Phi_u(\mathbf{z}), u)\, \mathrm{d}u.$$

The matrix $J_{\Phi_u} J_{\Phi_\tau}^{-1}$ is the Jacobian of flowing backward from time τ to time u, so by the stretching bound above its operator norm is at most $e^{L(\tau-u)}$. Taking norms and using $\|\nabla \operatorname{div} a\|_2 \le H$, we obtain

$$\left\|\nabla \log r_\tau(\Phi_\tau(\mathbf{z}))\right\|_2 \le e^{L\tau}\|\mathbf{z}\|_2 + H\int_0^\tau e^{L(\tau-u)}\, \mathrm{d}u \le e^{L\tau}\left(\|\mathbf{z}\|_2 + H\tau\right). \tag{11}$$

So the score at the end of a trajectory is controlled by how far its starting noise was from the origin, plus a term that grows with H and with how long we have flowed.

The rate at which the KL divergence grows involves the difference $\nabla \log P_\tau - \nabla \log Q_\tau$ evaluated at a sample from P_τ, meaning at a point $F_\tau(\mathbf{z})$ with $\mathbf{z} \sim \gamma$. The first score is bounded directly by Equation (11). For the second, we need to know where the point $F_\tau(\mathbf{z})$ came from under the *reference* flow, since Equation (11) is stated in terms of a trajectory's own starting point. Call that starting point $\mathbf{w}_\tau(\mathbf{z}) = G_\tau^{-1}(F_\tau(\mathbf{z}))$.

Since the two velocity fields differ by at most ε, the two trajectories that start from the same $\mathbf{z}$ cannot drift far apart. Grönwall's inequality gives $\|F_\tau(\mathbf{z}) - G_\tau(\mathbf{z})\|_2 \le \varepsilon\tau e^{L\tau}$. Pulling both points back through G_τ^{-1}, which stretches by at most $e^{L\tau}$, we get $\|\mathbf{w}_\tau(\mathbf{z}) - \mathbf{z}\|_2 \le \varepsilon\tau e^{2L\tau}$. In words, the noise point that the reference flow would have needed to start from is within $\varepsilon\tau e^{2L\tau}$ of the one the policy actually started from.

We can now bound the score difference. Applying Equation (11) to the policy at starting point $\mathbf{z}$, and to the reference at starting point $\mathbf{w}_\tau(\mathbf{z})$ with $\|\mathbf{w}_\tau(\mathbf{z})\|_2 \le \|\mathbf{z}\|_2 + \varepsilon\tau e^{2L\tau}$, the triangle inequality bounds the difference of the two scores at $F_\tau(\mathbf{z})$ by $2e^{L\tau}(\|\mathbf{z}\|_2 + H\tau) + \varepsilon\tau e^{3L\tau}$. Taking the expectation over $\mathbf{z} \sim \gamma$ and using $\mathbb{E}\|\mathbf{z}\|_2 \le \sqrt{D}$, we obtain

$$\mathbb{E}_{P_\tau}\left\|\nabla \log P_\tau - \nabla \log Q_\tau\right\|_2 \le 2e^{L\tau}\left(\sqrt{D} + H\tau\right) + \varepsilon\tau e^{3L\tau}. \tag{12}$$

Finally, we write down how the KL divergence evolves and integrate it. Let $K(\tau) = D_{\mathrm{KL}}(P_\tau \| Q_\tau)$. Both intermediate distributions are transported by their velocity fields, so they satisfy the continuity equations $\partial_\tau P_\tau + \operatorname{div}(P_\tau b) = 0$ and $\partial_\tau Q_\tau + \operatorname{div}(Q_\tau c) = 0$. Differentiating K in τ, substituting these, and integrating by parts gives

$$K'(\tau) = \mathbb{E}_{P_\tau}\left[(b-c)^\top\left(\nabla \log P_\tau - \nabla \log Q_\tau\right)\right]. \tag{13}$$

This is the key identity. It says that the KL divergence only grows where the two velocity fields disagree, at a rate given by the velocity gap dotted with the score gap. The integration by parts is justified because both scores grow at most linearly, which follows from Equation (11).

Everything we need is now in place. By the Cauchy–Schwarz inequality, $\|b - c\|_2 \le \varepsilon$, and Equation (12),

$$K'(\tau) \le \varepsilon\, \mathbb{E}_{P_\tau}\left\|\nabla \log P_\tau - \nabla \log Q_\tau\right\|_2 \le 2\sqrt{D}\,\varepsilon\, e^{L\tau} + 2H\varepsilon\,\tau e^{L\tau} + \varepsilon^2\,\tau e^{3L\tau}.$$

Since $K(0) = 0$, integrating from $\tau = 0$ to $\tau = 1$ gives

$$D_{\mathrm{KL}}(p_s \| p^{\mathrm{ref}}) = K(1) \le 2\sqrt{D}\,\varepsilon \int_0^1 e^{L\tau}\mathrm{d}\tau + 2H\varepsilon \int_0^1 \tau e^{L\tau}\mathrm{d}\tau + \varepsilon^2 \int_0^1 \tau e^{3L\tau}\mathrm{d}\tau \le e^{L}\big(2\sqrt{D}+H\big)\varepsilon + \tfrac{1}{2}e^{3L}\varepsilon^2,$$

where in the last step we bounded the integrals using $\int_0^1 e^{L\tau}\mathrm{d}\tau \le e^L$ and $\int_0^1 \tau e^{M\tau}\mathrm{d}\tau \le \frac{1}{2}e^M$. This is the bound for $D_{\mathrm{KL}}(p_s \| p^{\mathrm{ref}})$.

For the other direction, notice that nothing above distinguishes the policy from the reference. Both fields satisfy the same smoothness bounds, and the velocity gap is symmetric. Swapping the roles of b and c therefore gives the same bound on $D_{\mathrm{KL}}(p^{\mathrm{ref}} \| p_s)$, which completes the proof. □

Putting the two theorems together gives the statement used in Section 3.2. Theorem 1 bounds the deviation in the RMS norm by $R = \rho/(1+\lambda)$, and since the Euclidean norm of a D-dimensional vector is $\sqrt{D}$ times its RMS norm, this means $\varepsilon \le \sqrt{D}\,\rho/(1+\lambda)$. Substituting into Equation (10) bounds both KL divergences, at every training time, by

$$e^{L}\big(2D + H\sqrt{D}\big)\frac{\rho}{1+\lambda} + \frac{D}{2}\,e^{3L}\frac{\rho^2}{(1+\lambda)^2},$$

which is linear in the clip radius ρ for small radii.

Two caveats are worth stating. First, the smoothness constants L and H are assumptions about the velocity field that training produces, not consequences of clipping, and the hard projection in Equation (5) is itself not smooth at its boundary. Second, the bound is for exact ODE sampling, so a discretized solver or a distilled few-step flow map would need a separate argument. We also make no claim that the constants are tight.

C.3 Derivation: The flow-matching baseline isolates the reward tilt

Recall from Equation (4) that every method we consider regresses the policy toward a target of the form $\mathbf{v}^{\mathrm{anchor}} + A(\mathbf{v} - \mathbf{v}^{\mathrm{baseline}})$. This section is about the choice of $\mathbf{v}^{\mathrm{baseline}}$. Prior work subtracts the policy's own velocity $\mathbf{v}_\theta$, whereas FlowSRL subtracts the flow matching velocity $\mathbf{v}^{\mathrm{FM}}$ of the rollouts. We derive an exact identity that says $\mathbf{v}^{\mathrm{FM}}$ is the right thing to subtract, and then show what subtracting $\mathbf{v}_\theta$ adds instead.

We'll need a bit of notation. Fix a prompt and a reward r, and let π_θ be the distribution of the policy's rollouts. We write $A(\mathbf{x}_0) = r(\mathbf{x}_0) - \mathbb{E}_{\mathbf{x}_0\sim\pi_\theta}[r(\mathbf{x}_0)]$ for the centered advantage, which in practice we estimate by subtracting the group mean. Recall that the flow matching target of a rollout is $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_0$, and that the flow matching velocity of the rollout distribution is its conditional mean, $\mathbf{v}^{\mathrm{FM}}(\mathbf{x}_t, t) = \mathbb{E}[\mathbf{v} \mid \mathbf{x}_t, t]$. The sampling velocity $\mathbf{v}_\theta$ need not equal $\mathbf{v}^{\mathrm{FM}}$, since many different velocity fields generate the same endpoint distribution while taking different routes through the intermediate times. They coincide when the policy is a freshly pretrained flow matching model, but drift apart as RL training moves the policy.

The goal of reward-guided training is to shift the rollout distribution toward higher reward. The cleanest way to say what that means is the *reward tilt*, which is the family of distributions

$$\pi_\eta(\mathbf{x}_0) \propto \pi_\theta(\mathbf{x}_0)\exp\big(\eta\, r(\mathbf{x}_0)\big),$$

obtained by re-weighting each sample by its reward, with $\eta \ge 0$ controlling how aggressively. At $\eta = 0$ this is just π_θ, and increasing η moves probability mass toward high-reward samples. The question is then which direction the flow matching field has to move in order to follow this family.

The identity. Write $\mathbf{v}_\eta^{\mathrm{FM}}$ for the flow matching velocity of π_η under the same forward process. Under mild regularity conditions, its derivative at $\eta = 0$ is (Potaptchik et al., 2025; Bergmeister et al., 2026)

$$\left.\frac{\partial \mathbf{v}_\eta^{\mathrm{FM}}(\mathbf{x}_t, t)}{\partial \eta}\right|_{\eta=0} = \mathbb{E}_{\mathbf{x}_0\sim\pi_\theta}\left[A\,(\mathbf{v} - \mathbf{v}^{\mathrm{FM}}) \mid \mathbf{x}_t, t\right] =: \mathbf{C}(\mathbf{x}_t, t).$$

In words, the direction that tilts the distribution toward higher reward is $\mathbf{C}$, and $\mathbf{C}$ is exactly what our update computes when the baseline is $\mathbf{v}^{\mathrm{FM}}$.

It helps to see why $\mathbf{C}$ is a covariance. Since $\mathbf{v}^{\mathrm{FM}}$ is deterministic once we condition on $(\mathbf{x}_t, t)$, we can pull it out of the expectation,

$$\mathbf{C}(\mathbf{x}_t, t) = \mathbb{E}[A\,\mathbf{v} \mid \mathbf{x}_t, t] - \mathbb{E}[A \mid \mathbf{x}_t, t]\,\mathbb{E}[\mathbf{v} \mid \mathbf{x}_t, t] = \mathrm{Cov}(A, \mathbf{v} \mid \mathbf{x}_t, t).$$

So $\mathbf{C}$ measures how the flow matching target co-varies with the advantage among the rollouts that pass through $\mathbf{x}_t$. Note that $\mathbb{E}[A \mid \mathbf{x}_t, t]$ is not zero in general, even though A has mean zero overall. A noisy point that tends to lead to good samples has a positive conditional advantage, and one that tends to lead to bad samples has a negative one. Subtracting $\mathbf{v}^{\mathrm{FM}}$ is precisely what removes this conditional mean and leaves the covariance.

What the policy baseline adds. Now suppose we subtract $\mathbf{v}_\theta$ instead. Adding and subtracting $\mathbf{v}^{\mathrm{FM}}$ inside the expectation, and again using that both $\mathbf{v}^{\mathrm{FM}}$ and $\mathbf{v}_\theta$ are deterministic given $(\mathbf{x}_t, t)$, we get

$$\mathbb{E}[A\,(\mathbf{v} - \mathbf{v}_\theta) \mid \mathbf{x}_t, t] = \mathbf{C}(\mathbf{x}_t, t) + \mathbb{E}[A \mid \mathbf{x}_t, t]\,(\mathbf{v}^{\mathrm{FM}} - \mathbf{v}_\theta).$$

The first term is the direction we want. The second is a push along $\mathbf{v}^{\mathrm{FM}} - \mathbf{v}_\theta$, scaled by the conditional advantage. That direction is the gap between two velocity fields that generate the same distribution, so moving along it does not change what the policy produces and has nothing to do with the reward. At initialization the gap is zero and the two baselines agree. As soon as RL moves the policy away from its own flow matching field, however, the policy baseline starts adding this spurious term to every update. Subtracting $\mathbf{v}^{\mathrm{FM}}$ removes it, which is the sense in which the flow matching baseline isolates the reward tilt.

D Diversity Mechanisms

D.1 Implementation details

2D ring. The base model is a flow-matching model trained on a ring embedded in 32 dimensional space, where the first two coordinates come from a ring and the rest of the 30 dimensions are sampled from a Gaussian distribution. The reward is the cycloid of Figure 3, which we evaluate exactly and cares only about the first two dimensions. We report the average reward of a batch of samples against their diversity, measured as the entropy of their angular distribution around the ring in the first two dimensions, and normalized so that the base model sits at 1. Figure 6 sweeps the clip radius over $\rho \in \{0.25, 0.5, 1, 2\}$ and the KL weight over $\lambda \in \{0.1, 0.3, 0.5, 1\}$, both without and with the entropy bonus, so each of those two arms is a 4×4 grid in which every cell is a separate run. The horizon-conditioned refinement curve is instead a single model trained over a distribution of horizons and evaluated at horizons spaced logarithmically over $T \in [0.01, 1]$, and Figure D.3 shows its samples along that curve.

All 2D runs share the following settings. Training rollouts use 20 Euler steps, while evaluation, snapshots and figures use 200. Every training step draws 4096 samples, and each sample is re-noised at $K = 8$ fresh $(t, \boldsymbol{\epsilon})$ pairs to compute the loss. We optimize with AdamW, using $\beta_1 = 0.9$ and no weight decay. The settings that differ between methods are listed in Table D.1, and every configuration is run with 5 seeds. For the entropy bonus we use a fixed KDE bandwidth of $h = 0.05$ in 2D.

Text-to-image. The policy is a LoRA adapter on SD3.5-M trained on HPDv2 prompts with the PickScore reward. We measure diversity as the mean pairwise DreamSim distance between samples generated from the same prompt, using a fixed set of noises so that any change in diversity comes from the policy rather than from the noises drawn, and we evaluate every arm without guidance. Figure 5c sweeps the KL weight from 10^{-4} to 3×10^{-1}, the entropy weight over $\alpha \in [0.1, 0.25]$, and the refinement horizon T for both CombGen and horizon-conditioned refinement. CombGen is an inference-time procedure that applies to any reward-tuned checkpoint, and we apply it to the FlowSRL checkpoint of Table 1, whereas the horizon-conditioned refinement arm is a separately trained policy that is conditioned on the horizon it is asked to refine over. Both arms are sampled the same way, so the comparison between them isolates the effect of training for the horizon. The entropy bonus uses the self-features of Section D.4 rather than an off-the-shelf encoder. The diversity is measured using 64 prompts from the test set, using 4 generated images per prompt. The models were trained for 400 steps. Three seeds were average over to obtain the final results. The horizon T was sampled from a logit-normal distribution $logit(T) \sim \mathcal{N}(1.5, 1)$. Other parameters were the same as the ones in Table 1.

Table D.1: **Per-method settings for the 2D runs.** Settings shared by all methods are given in Section D.1. †Figure 3 shows the first 2000 steps.

	FlowSRL	DiffusionNFT	RAM	FlowSRL + HCR
Groups × group size	128×32	128×32	128×32	512×8
Training steps	$5000^{\dagger}$	5000	5000	4000
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	10^{-3}
AdamW β_2	0.95	0.999	0.99	0.95
Gradient clipping	1.0	1.0	none	1.0
Clip radius ρ	0.631	–	–	0.631
KL weight λ	10^{-4}	10^{-4}	10^{-4}	10^{-2}
EMA sampler decay	–	$0 \to 0.5$	$0 \to 0.9$	–
RAM coefficient	–	–	1000	–

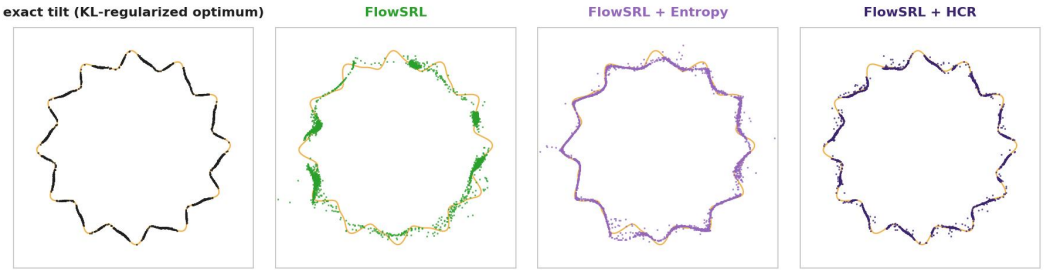


Figure D.1: **Samples on the 2D ring.** Representative examples from the diversity experiments in Section 4.2. The full sweep behind Figure 6 is shown sample by sample in Figure D.2.

D.2 The exact reward tilt as a reference

In the low-dimensional setting the reward-tilted distribution $\pi_\tau(\mathbf{x}_0) \propto \pi_{\text{ref}}(\mathbf{x}_0)\exp(\tau\, r(\mathbf{x}_0))$ can be sampled exactly by importance sampling against the pretrained model, which gives the exact optimum of the KL-regularized RL objective at every temperature τ and therefore a reference curve in the reward-diversity plane. Figure D.2 shows the samples behind that curve and behind the sweeps of Figure 6. Specifically, we draw 4×10^5 proposals from the pretrained model with the 200-step evaluation sampler and self-normalize the weights $\exp(\tau\, r(\mathbf{x}_0))$, which keeps the effective sample size above 2.1×10^4 at every temperature up to $\tau = 100$. Each point on the curve and each tilt panel uses 2×10^4 samples resampled with these weights

D.3 Group structure in horizon-conditioned refinement

Members of a group share the coarse structure of their anchor, so an advantage never compares one mode against another, and a small refinement horizon T bounds how far a refinement can travel. Figure D.3 sweeps the horizon of a single model trained on log-uniform T across the 2D ring, and shows the two limits it interpolates between, the base model at $T = 0$ and the collapsed policy at $T = 1$.

D.4 Feature spaces for the entropy bonus

For text-to-image models we use features from the frozen reference model itself, re-noising each sample to $t = 1/2$ with noise shared across the group and average pooling the activations of the pretrained transformer's middle block onto a 4×4 spatial grid. This coarse representation captures compositional differences between samples, which are among the first aspects of diversity to collapse, and costs one forward pass through a model we already hold in memory. Because the entropy bonus encourages samples to move toward underrepresented regions, it also requires a slightly larger KL regularization weight to keep training stable.

We initially used standard feature extractors such as CLIP (Radford et al., 2021) and DINOv2 (Oquab et al., 2023), but found that this makes the diversity objective inherit the biases of the extractor. For

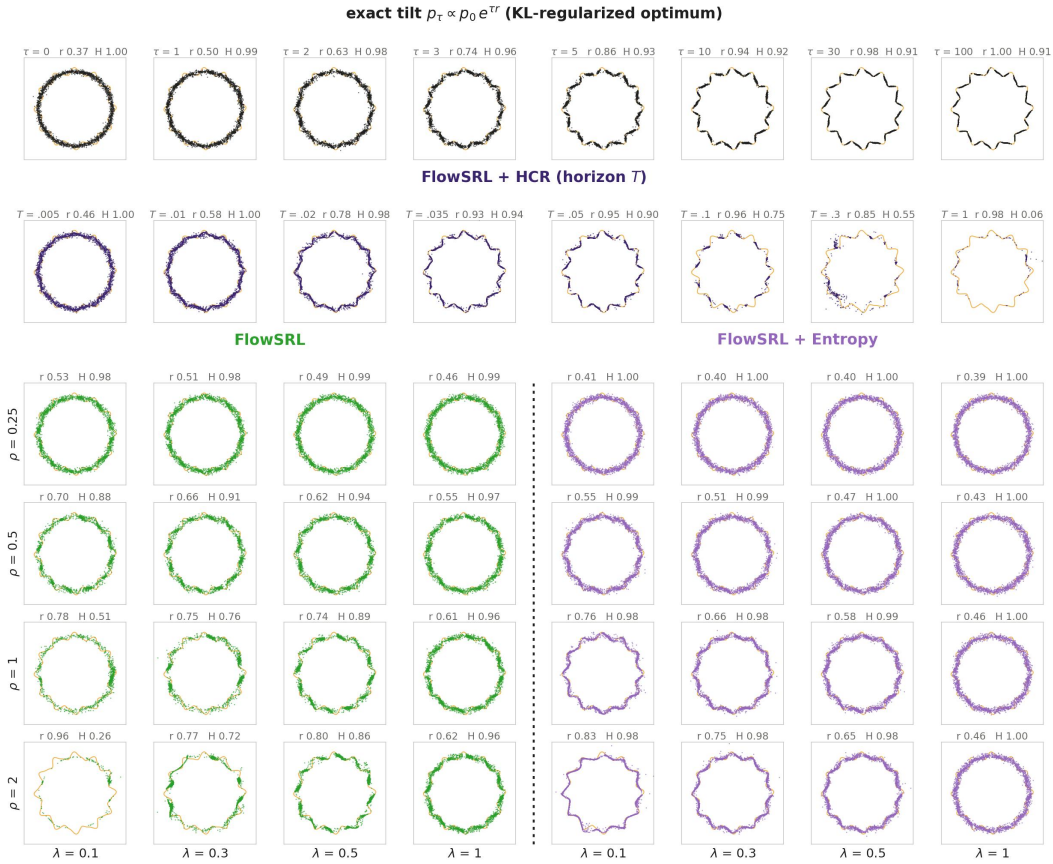


Figure D.2: **Samples behind the frontier of Figure 6.** Top: the exact tilt $p_\tau \propto p_0 e^{\tau r}$ at the temperatures that trace the reference curve. Middle: horizon-conditioned refinement at the refinement horizons that trace its curve. Bottom: FlowSRL without (left) and with (right) the entropy bonus over the full sweep of clip radius ρ (rows) and KL weight λ (columns). The average reward r and the normalized angular entropy H are given above each panel. Without either mechanism, entropy falls away as soon as the reward passes 0.8, and the highest-reward cell keeps only 26% of it on a handful of lobes. The entropy bonus holds every cell at 0.98 or above, at a lower reward than the bare sweep reaches at matched ρ and λ, so what it buys is the trade-off rather than the reward, which is what places it on the tilt in Figure 6 instead of below it. Sweeping the horizon instead holds 0.90 at a reward of 0.95, and gives up coverage only once the horizon grows past 0.05.

example, DINOv2 is largely insensitive to style changes, which leads to style collapse. Relying on a separate third-party encoder also introduces additional complexity that we would prefer to avoid. The self-features of Section 3.3 match or outperform CLIP features in our experiments.

E Flow Maps and Search in the Rollout

This section collects the technical details behind the hard-reward experiments of Section 4.3. The flow map is a host rather than a contribution. It makes rollouts cheap enough to search over, and its look-ahead makes the reward available at every jump.

Flow maps. Flow maps (Kim et al., 2024; Boffi et al., 2024; Sabour et al., 2025b; Geng et al., 2026) generalize flow-based and consistency models (Song et al., 2023; Lu & Song, 2024) by training a single network $\mathbf{f}_\theta(\mathbf{x}_t, t, s)$ to map $\mathbf{x}_t$ directly to $\mathbf{x}_s$ along the PF-ODE. Following prior work, we use the simple parameterization

$$\mathbf{f}_\theta(\mathbf{x}_t, t, s) = \mathbf{x}_t + (s - t)\, \mathbf{F}_\theta(\mathbf{x}_t, t, s), \tag{14}$$

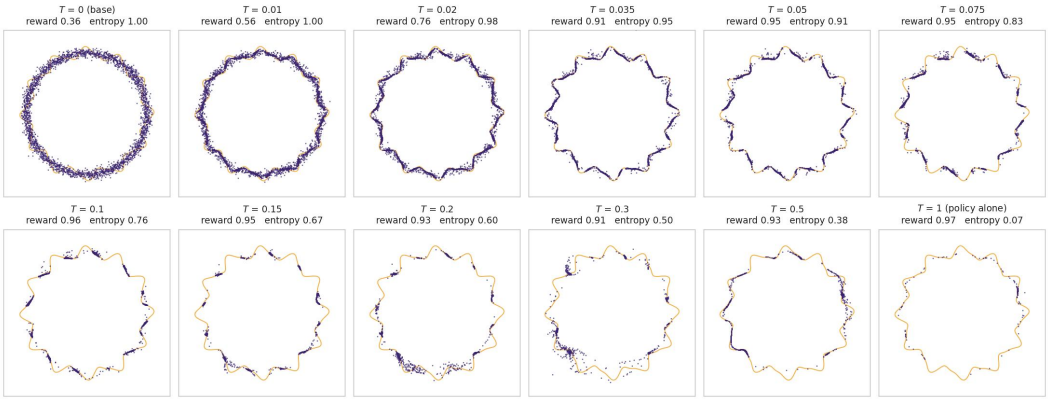


Figure D.3: **Sweeping the refinement horizon of a single horizon-conditioned refinement model on the 2D ring.** Samples (purple) against the reward ridge (orange) for horizons from $T = 0$, where the base model generates alone, to $T = 1$, where the policy does. Reward rises to 0.95 by $T = 0.05$ while 91% of the base model's angular entropy is retained, and every further increase of the horizon buys almost no reward and costs entropy, ending at 7% on the few lobes the policy alone keeps.

so that $\mathbf{F}_\theta$ is the average velocity along the PF-ODE over $[t, s]$, the diagonal $\mathbf{F}_\theta(\mathbf{x}_t, t, t)$ is the instantaneous velocity $\mathbf{v}_\theta(\mathbf{x}_t, t)$, and $\mathbf{f}_\theta(\mathbf{x}_t, t, 0)$ predicts a clean image in one step. We use a 4-step map distilled from SD3.5-M following Sabour et al. (2025b).

Applying FlowSRL to a flow map. The diagonal of a flow map is itself a velocity model (Boffi et al., 2026), so Algorithm 1 applies to it unchanged.

Lemma 4 (Boffi et al., 2026)**.** *Let $\mathbf{f}_\theta$ be a flow map of the PF-ODE $\mathrm{d}\mathbf{x}_t/\mathrm{d}t = \mathbf{b}_t(\mathbf{x}_t)$ in the form of Equation (14). Then $\mathbf{F}_\theta(\mathbf{x}_t, t, t) = \mathbf{b}_t(\mathbf{x}_t)$ for all $\mathbf{x}_t$ and t.*

To see this, Equation (14) gives $(\mathbf{f}_\theta(\mathbf{x}_t, t, s) - \mathbf{x}_t)/(s - t) = \mathbf{F}_\theta(\mathbf{x}_t, t, s)$ for $s \neq t$, and as $s \to t$ along the PF-ODE the left side tends to $\partial_s \mathbf{f}_\theta(\mathbf{x}_t, t, s)|_{s=t} = \mathbf{b}_t(\mathbf{x}_t)$. We therefore update at the diagonal and keep the off-diagonal jumps consistent with the updated velocity through a standard AYF consistency step (Sabour et al., 2025b) on the same rollouts,

$$\mathcal{L}_{\mathrm{cons}}(\theta) = \mathbb{E}_{\mathbf{x}_t, t, s}\left[\left\|\mathbf{F}_\theta(\mathbf{x}_t, t, s) - \mathrm{sg}\big(\mathbf{F}_\theta(\mathbf{x}_t, t, s)\big) + \mathbf{g}\right\|_2^2\right], \tag{15}$$

where $\mathbf{g}$ is the AYF tangent . This term is a sampler guard rather than a reward mechanism, leaving the reward unchanged to within ± 0.003, but it cannot be dropped on hard rewards.

Search in the rollout. Hard rewards pose a discovery problem. If no sample in a group is positive, every advantage is zero and learning never starts. At the jump from t to s, the guided sampler moves along the gradient of the differentiable part r_{diff} of the reward evaluated at the look-ahead $\mathbf{f}_\phi(\mathbf{x}_t, t, 0)$,

$$\mathbf{x}_s^{\mathrm{guided}} = \mathbf{f}_\phi(\mathbf{x}_t, t, s) + \eta\,(t - s)\,\frac{\nabla_{\mathbf{x}_t} r_{\mathrm{diff}}\big(\mathbf{f}_\phi(\mathbf{x}_t, t, 0)\big)}{\max\big(\|\nabla_{\mathbf{x}_t} r_{\mathrm{diff}}\|,\, \gamma\big)}, \qquad \phi \in \{\theta^{\mathrm{base}}, \theta\}, \tag{16}$$

where η is the guidance strength and γ a soft gradient floor that keeps a vanishing gradient from being normalized into a random kick. Guidance only proposes. Plain samples and proposals are then scored with the full, non-differentiable reward and the better half enters the training group, so selection rather than the gradient is what carries the non-differentiable part of the reward. Pseudocode is given in Section B.

F Experimental Details

F.1 Hyperparameters

Table F.1: **Hyperparameters per experiment group.** The 2D column lists FlowSRL; the baselines are given in Table D.1. Rollout steps are for training; all evaluation in 2D uses 200 steps.

	2D	T2I	Layout search
Rollout steps	20	10	4
Groups × group size	128×32	48×24	48×24
Fully online / refresh	yes	yes	yes
Clip radius ρ	0.631	0.7	0.7
NFT soft-clip scale	0.2	0.2	0.2
KL weight	10^{-4}	10^{-4}	10^{-2}
KDE weight α / bandwidth / features	– / 0.05 / identity	–/1.0/self-features ($t = 0.5$)	–
fm-baseline LR	3×10^{-4}	3×10^{-4}	3×10^{-4}
Candidates / guide source / η / floor	–	–	TBD
Advantage baseline (plain / demo)	group	group	group
LR / optimizer / LoRA rank	3×10^{-4} / AdamW / –	3×10^{-4} / AdamW / 32	3×10^{-4} / AdamW / 32
Steps / seeds	5000 / 5	400 / 3	1000 / 2

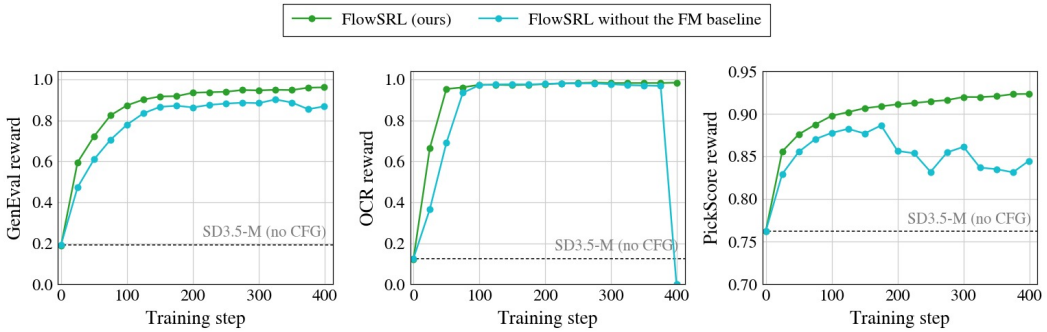


Figure G.1: **Ablating the update baseline.** FlowSRL (green) against the same algorithm with the policy velocity $\mathbf{v}_\theta$ substituted for the learned flow-matching velocity $\mathbf{v}^{\mathrm{FM}}$ as the update baseline (blue), at the full batch. Without the flow-matching baseline the reward rises more slowly and ends lower on all three rewards, drifting back down on PickScore after step 175 and collapsing on OCR near step 400. Dashed line: SD3.5-M without guidance.

Table G.1: **UniGenBench++ results** after 500 training steps, evaluated without guidance and with CFG $= 2.0$. All methods use the same reward and roll out 48×24 samples per step; the small-batch rows train on only 6 of the 24 samples in each group. **Bold**: best per column among methods at the same guidance scale and batch size.

Model	Style	World	Attr.	Action	Relat.	Compd.	Gramm.	Logic	Layout	Text	Overall
Full batch (48×24)											
DiffusionNFT	90.7	84.0	81.1	72.6	**81.7**	**81.4**	**67.1**	**52.1**	86.4	9.5	70.7
+ CFG = 2.0	**95.8**	**92.4**	83.9	75.9	**81.8**	**80.5**	**67.7**	50.7	86.2	21.8	73.7
RAM	87.0	85.9	80.3	74.4	79.2	79.5	66.7	48.4	83.6	8.3	69.3
+ CFG = 2.0	92.2	91.1	85.2	**76.4**	78.8	78.2	65.1	53.2	84.7	19.8	72.5
FlowSRL	**93.3**	**91.0**	**84.8**	**77.4**	80.7	79.0	66.4	50.5	**87.9**	**18.7**	**73.0**
+ CFG = 2.0	93.4	92.2	**86.3**	76.0	80.8	78.5	67.5	**53.7**	**87.1**	**24.4**	**74.0**
Small batch: trained on the 3 highest- and 3 lowest-reward samples of each group (48×6)											
DiffusionNFT	71.4	63.9	65.5	65.8	60.9	52.3	**65.2**	41.1	71.1	5.5	56.3
+ CFG = 2.0	77.8	78.6	75.3	68.5	69.8	58.9	**64.2**	45.2	76.1	10.3	62.5
RAM	0.0	0.0	0.0	0.0	0.0	0.0	5.9	0.0	0.0	0.0	0.6
+ CFG = 2.0	0.2	0.0	0.0	0.0	0.0	0.0	6.3	0.0	0.0	0.0	0.6
FlowSRL	**87.8**	**89.9**	**78.5**	**70.1**	**77.8**	**73.1**	62.7	**42.9**	**84.3**	**14.9**	**68.2**
+ CFG = 2.0	**92.7**	**92.2**	**83.1**	**73.5**	**79.3**	**77.7**	62.6	**47.5**	**87.5**	**23.3**	**71.9**

G Additional Results and Ablations

G.1 Text-to-image

Table G.1 breaks the UniGenBench++ score of Figure 4 down by category, at both batch sizes and with and without guidance, and Figure G.2 gives the corresponding training curves. Figure G.3 extends the PickScore panels of Figure 5 to GenEval and OCR. Figure G.1 ablates the update baseline of Section 3.2, comparing FlowSRL against the same algorithm with the policy velocity $\mathbf{v}_\theta$ in place of the learned flow-matching velocity $\mathbf{v}^{\mathrm{FM}}$.

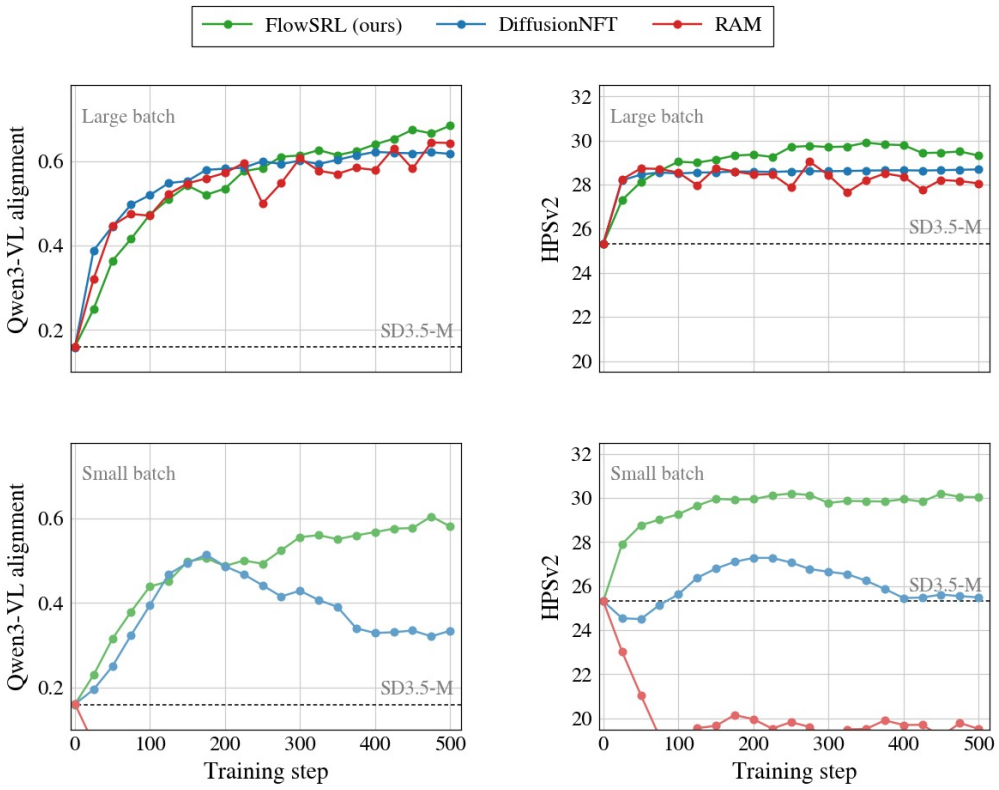
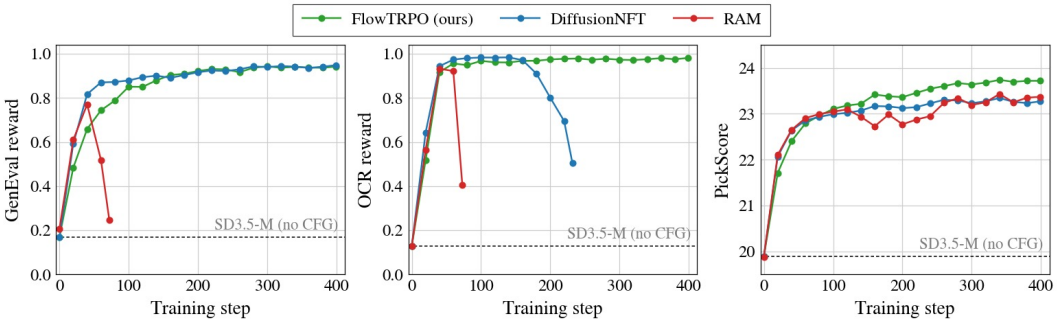
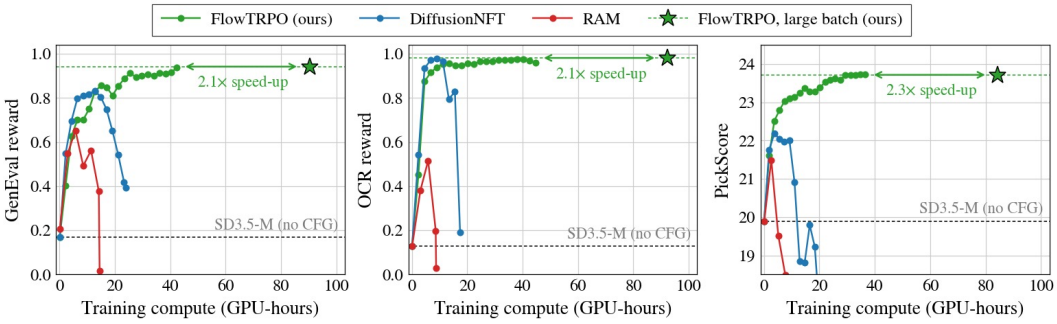


Figure G.2: **Training curves for the UniGenBench++ runs of Figure 4 and Table G.1.** Qwen3-VL alignment (left) and HPSv2 (right) against training step, at the full batch (top) and the small batch (bottom). Dashed line: SD3.5-M without guidance. At the full batch all three methods track each other, with FlowSRL pulling ahead late on both metrics. At the small batch only FlowSRL keeps improving, while DiffusionNFT peaks around step 175 and falls back to the base model on HPSv2, and RAM collapses below it within 100 steps.



(a) Full batch (48×24), reward against training step.



(b) Small batch (48×6), reward against training compute.

Figure G.3: **Text-to-image training curves on all three rewards**, extending the PickScore panels of Figure 5. (a) At the full batch, DiffusionNFT and RAM peak and then collapse on GenEval and OCR, while FlowSRL holds for all 400 steps and ends highest on PickScore. (b) Training on only the 3 highest- and 3 lowest-reward samples of each group, both prior methods collapse on every reward, whereas FlowSRL reaches the reward of its own full-batch run (star) in less than half the compute. Dashed line: SD3.5-M without guidance.

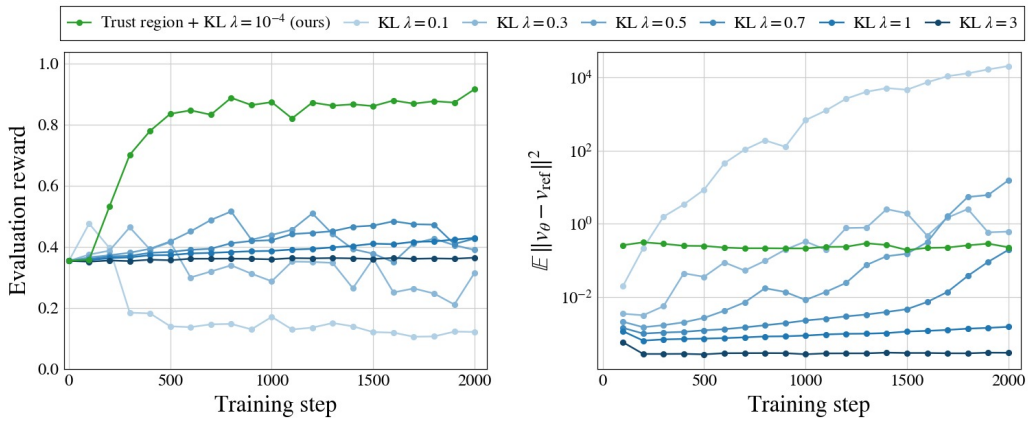


Figure G.4: **KL regularization is either too weak or too strong.** Reward (left) and mean squared distance to the reference (right) across KL weights on the problem of Figure 3. Small weights are unstable and large weights sacrifice reward. FlowSRL achieves both stability and high reward.

G.2 2D TOY PROBLEMS

On 2D problems, every failure mode is plainly visible. Figure 3 shows that DiffusionNFT and RAM peak and then collapse while FlowSRL trains stably for 2000 steps, and Figure G.4 shows that no KL weight repairs this, since weak regularization still drifts into instability while strong regularization buys stability only by giving up reward. The same pattern holds across learning rates, where FlowSRL is stable at every value we tried on the arc reward and neither prior method is stable at any, and removing deviation clipping reproduces their collapse. Turning to diversity, Figure 6 shows that the clip radius and KL weight alone concentrate the samples on a few lobes of the reward ridge, while the entropy bonus keeps all 12 lobes in every cell of the sweep. Neither ingredient works alone, since without the KL anchor the samples spray off the ring, and the bonus by itself never beats the KL dial. The gap also requires a multi-modal reward, since on the single-lobe arc reward the tilt itself loses entropy and the sweep sits within 0.15 of it.



Figure G.5: **Samples with and without search in the rollout, on multiple prompts and masks**, the counterpart of Figure 7. Fixed prompts and noises after 1000 training steps, with insets magnifying the target region. Here standard rollouts do place the scene, but the surrounding canvas is not fully cleared.

G.3 HARD-REWARD LAYOUTS

Figure G.5 shows samples with and without search in the rollout on the multi-prompt, multi-mask setting, the counterpart of the single-prompt panels in Figure 7.