# LEARNING WHERE AND WHAT TO OPTIMIZE: FROM DIFFUSION DISTILLATION TO REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Teacher-guided distillation provides dense supervision and fast optimization, whereas reward-driven reinforcement learning (RL) directly optimizes scalar rewards at a substantially higher sampling cost. We seek to unify these complementary strengths within a single diffusion post-training framework. A natural alternative is sequential training: distill first, then switch to reward optimization. However, such stage-wise training can suffer from post-switch degradation, late-stage collapse, and sensitivity to the handoff schedule, particularly under limited prompt diversity. We propose a *Unified Reward-adaptive Framework* (URF) that jointly determines *where* to optimize and *what* velocity target to follow for each generated sample. Both decisions are derived from a single reward-adaptive objective and controlled by the same sample-wise reward-preference weight $p \in [0, 1]$. For the velocity target, motivated by the coarse negative-sample update directions, URF uses negative reward feedback to constrain teacher-guided correction rather than directly applying the repulsive NFT update. As the reward increases, $p$ smoothly transitions the update from DiffusionOPD to DiffusionNFT, eliminating the need for an explicit stage-wise switch. To our knowledge, URF is the first diffusion post-training framework to unify teacher-guided distillation and reward-driven RL at both the query and velocity-target levels within a single reward-adaptive objective. Extensive experiments show that URF combines substantially faster convergence than teacher-guided distillation with continued reward-driven improvement, achieving 0.963 on both GenEval and OCR while converging 1.7–7.9$\times$ faster than DiffusionOPD in multi-reward training.

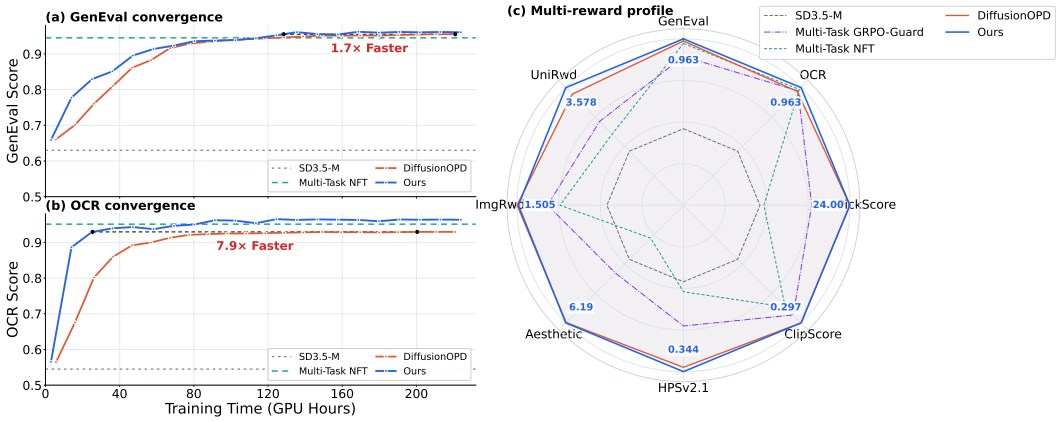



Figure 1: URF achieves faster convergence and higher performance ceilings than the multi-task teacher-distillation and reinforcement-learning baselines on both (a) GenEval and (b) OCR. (c) URF delivers the strongest and most balanced performance across the eight reward objectives.



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.

# 1 INTRODUCTION

Diffusion and flow-based generative models have become a dominant paradigm for high-quality text-to-image generation (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023; Liu et al., 2023; Esser et al., 2024). Their post-training has developed along two complementary branches: teacher-guided distillation and reward-driven reinforcement learning (RL). Distillation supplies dense teacher supervision (Salimans & Ho, 2022). DiffusionOPD performs this supervision directly on states visited by the current student, reducing covariate shift and enabling multi-reward transfer from reward-specialized teachers (Li et al., 2026).

Reward-driven RL methods instead optimize scalar rewards from generated samples. Reverse-process policy-gradient methods typically use PPO/GRPO-style objectives and require costly rollout-based optimization (Schulman et al., 2017; Shao et al., 2024; Black et al., 2024; Liu et al., 2025). DiffusionNFT instead performs online reward optimization through the forward process, requiring only generated endpoints rather than full denoising trajectories (Zheng et al., 2026). Thus, distillation offers efficient dense supervision but is bounded by the teacher, whereas RL enables continued reward-driven improvement at higher sampling cost.

A natural combination is sequential training: distill first, then switch to reward optimization. However, as shown in Figure 2(a), switching from DiffusionOPD to DiffusionNFT at 60 steps causes post-switch degradation, whereas switching at 90 steps leads to late-stage collapse. It also shows that the convergence–performance trade-off is highly sensitive to the handoff schedule. Moreover, Figure 2(b) measures the alignment gap between positive- and negative-sample updates, $\Delta\text{Alignment} = \text{Alignment}_{+} - \text{Alignment}_{-}$, where $\text{Alignment}_{\pm}$ denotes the mean pairwise cosine similarity among the update directions of positive (negative) samples. Its consistently positive and increasing value indicates that negative samples induce less aligned update directions than positive samples, reducing learning efficiency. This directional deficit may become particularly problematic when the available prompt set is small, where the diversity of reward-driven supervision is further limited. Consistent with this interpretation, Figure 2(c) shows that DiffusionNFT becomes increasingly unstable and eventually collapses as the number of training prompts decreases. Overall, these factors make OPD→NFT sequential training inherently fragile.

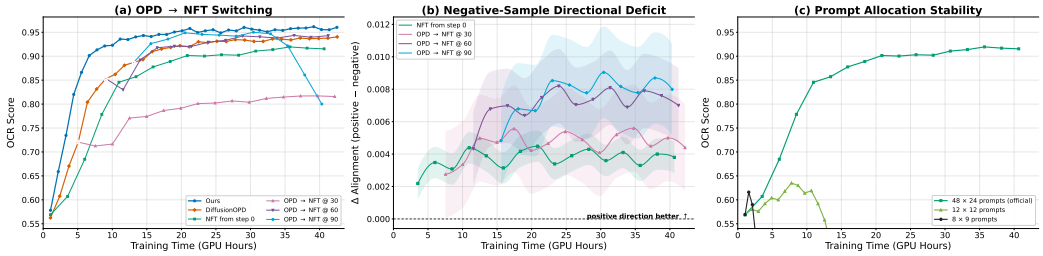



Figure 2: Observed failure modes of OPD→NFT sequential training and reward-only NFT optimization. (a) OCR training dynamics when switching from OPD to NFT at steps 30, 60, and 90. (b) Positive–negative update-alignment gap during NFT training. (c) OCR performance under three prompt allocations (number of prompts × samples per prompt): $48\times24$, $12\times12$, and $8\times9$.

We therefore aim to bridge teacher-guided distillation and reward-driven RL through a continuous reward-adaptive interpolation, while adaptively guiding negative samples with the teacher. Related LLM post-training work has explored distillation–RL integration and online/off-policy hybrids (Agarwal et al., 2024; Xiong et al., 2024; Noukhovitch et al., 2025), but, to our knowledge, no analogous reward-adaptive transition has been studied for diffusion post-training.

We propose a *Unified Reward-adaptive Framework* (URF) that jointly determines *where* to optimize and *what* velocity target to follow for each generated sample. Specifically, we introduce a unified reward-adaptive objective (Eq. (7)) governed by a sample-wise reward-preference weight $p_i$. From this objective, we derive closed-form solutions for both the reward-adaptive query distribution (Eq. (11)) and the reward-adaptive velocity target (Eqs. (17)–(18)). In particular, motivated by the coarse negative-sample update directions in DiffusionNFT (Figure 2(b)), the velocity target does not directly apply the repulsive $\boldsymbol{v}^{\star}_{\text{NFT}-}$ update; instead, negative reward feedback constrains the teacher along the reward-supported direction to form a teacher-guided corrective target. The same $p_i$ con-

trols both the query and velocity solutions, producing a synchronized sample-wise transition from student-visited teacher supervision toward endpoint-derived reward supervision without an explicit stage-wise switch.

We evaluate URF through multi-reward joint training, single-reward head-to-head comparisons, distillation to fewer-step students, and component ablations. In multi-reward joint training, URF is 1.7–7.9$\times$ more training-efficient than DiffusionOPD while achieving strong final performance, with scores of 0.963 on GenEval, 0.963 on OCR, and 24.003 on PickScore.

## 2 BACKGROUND

### 2.1 DIFFUSION AND FLOW MODELS

Let $\boldsymbol{x}_0$ denote a clean image conditioned on prompt $\boldsymbol{c}$. A diffusion forward process constructs

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{1}$$

with the corresponding probability-path velocity $\dot{\boldsymbol{x}}_t = \dot{\alpha}_t \boldsymbol{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}$.

Flow matching and its generic post-training form can then be written as

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}) - \dot{\boldsymbol{x}}_t\|_2^2\right], \qquad \mathcal{L}_{\mathrm{PT}} = \mathbb{E}\left[\|\boldsymbol{v}_\theta(\boldsymbol{z}_t, t, \boldsymbol{c}) - \boldsymbol{v}^\star\|_2^2\right], \tag{2}$$

where $\boldsymbol{z}_t$ denotes the *query* at which the model is optimized and $\boldsymbol{v}^\star$ denotes the corresponding *velocity target*. DiffusionOPD and DiffusionNFT differ in how they construct these two quantities.

### 2.2 DIFFUSIONOPD: ON-POLICY TEACHER SUPERVISION

DiffusionOPD (Li et al., 2026) distills task-specialized teachers along states visited by the current student. Its objective is derived from the reverse KL between the student and teacher transition kernels evaluated along student-generated trajectories. Because the two transition kernels share the same covariance, the KL reduces to mean matching:

$$\mathcal{L}_{\mathrm{OPD}}^{\mathrm{ODE}}(\theta) = \mathbb{E}_{\boldsymbol{x}_{0:N}^{\mathrm{old}} \sim p_{S,\theta}}\left[\sum_{j=0}^{N-1} \frac{1}{2}\left\|\mu_S(\boldsymbol{z}_{t_j}^{\mathrm{OPD}}; \theta) - \mu_T(\boldsymbol{z}_{t_j}^{\mathrm{OPD}})\right\|_2^2\right], \qquad \boldsymbol{z}_{t_j}^{\mathrm{OPD}} = \boldsymbol{x}_{t_j}^{\mathrm{old}}, \tag{3}$$

where $\mu_S$ and $\mu_T$ denote the student and teacher one-step transitions at the same on-policy state.

In the velocity view used throughout this paper, DiffusionOPD is therefore characterized by

$$\boxed{\boldsymbol{z}_t^{\mathrm{OPD}} = \boldsymbol{x}_t^{\mathrm{old}}, \qquad \boldsymbol{v}_{\mathrm{OPD}}^\star = \boldsymbol{v}_T\left(\boldsymbol{x}_t^{\mathrm{old}}, t, \boldsymbol{c}\right).} \tag{4}$$

### 2.3 DIFFUSIONNFT: FORWARD-PROCESS REWARD SUPERVISION

DiffusionNFT (Zheng et al., 2026) instead constructs supervision from the reward of a generated endpoint. The old policy generates $\boldsymbol{x}_0$ with reward $R(\boldsymbol{x}_0, \boldsymbol{c})$ and optimality weight $r \in [0, 1]$, and forward-noises it as $\boldsymbol{z}_t^{\mathrm{NFT}} = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}$, with forward-process velocity $\boldsymbol{u}_t = \dot{\alpha}_t \boldsymbol{x}_0 + \dot{\sigma}_t \boldsymbol{\epsilon}$. DiffusionNFT defines positive and negative branches around the old policy as $\boldsymbol{v}_\theta^+ = (1-\beta)\boldsymbol{v}^{\mathrm{old}} + \beta \boldsymbol{v}_\theta$ and $\boldsymbol{v}_\theta^- = (1+\beta)\boldsymbol{v}^{\mathrm{old}} - \beta \boldsymbol{v}_\theta$, and optimizes

$$\mathcal{L}_{\mathrm{NFT}} = \mathbb{E}\left[r\left\|\boldsymbol{v}_\theta^+ - \boldsymbol{u}_t\right\|_2^2 + (1-r)\left\|\boldsymbol{v}_\theta^- - \boldsymbol{u}_t\right\|_2^2\right]. \tag{5}$$

Equating each branch with $\boldsymbol{u}_t$ gives the corresponding trainable-velocity targets $\boldsymbol{v}_{\mathrm{NFT}\pm}^\star = \boldsymbol{v}^{\mathrm{old}} \pm (\boldsymbol{u}_t - \boldsymbol{v}^{\mathrm{old}})/\beta$. Thus, DiffusionNFT can be summarized as

$$\boxed{\boldsymbol{z}_t^{\mathrm{NFT}} = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{v}_{\mathrm{NFT}}^\star = r\,\boldsymbol{v}_{\mathrm{NFT}+}^\star + (1-r)\boldsymbol{v}_{\mathrm{NFT}-}^\star = \boldsymbol{v}^{\mathrm{old}} + \frac{2r-1}{\beta}\left(\boldsymbol{u}_t - \boldsymbol{v}^{\mathrm{old}}\right).} \tag{6}$$

Unlike teacher distillation, this update is driven directly by the observed reward and is not restricted to a fixed teacher. DiffusionOPD and DiffusionNFT therefore provide complementary teacher- and reward-derived supervision, motivating our joint adaptation of both the query and velocity target.

# 3 METHOD

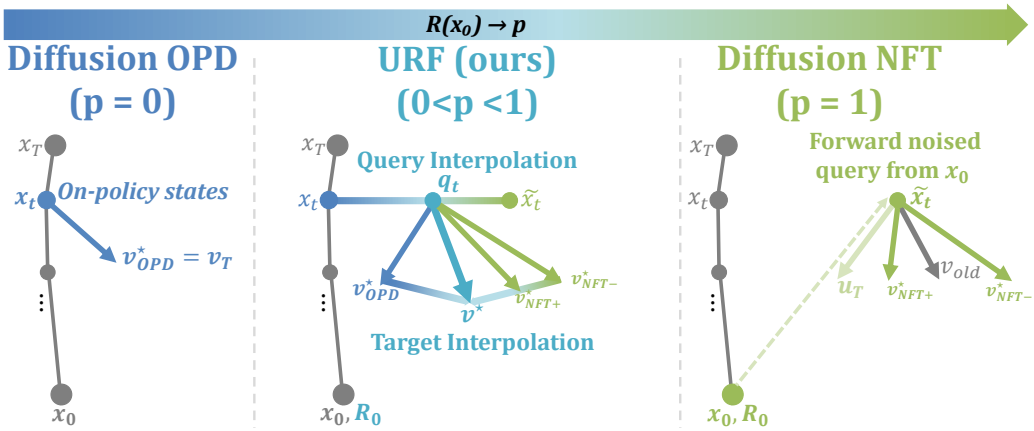



Figure 3: URF interpolates DiffusionOPD and DiffusionNFT at both the query and velocity-target levels through a sample-wise reward-preference weight $p_i$. Low $p_i$ favors on-policy teacher supervision, while high $p_i$ favors endpoint-derived forward-process supervision, with $p_i = 0$ recovering the OPD limit and $p_i = 1$ reaching the NFT-side limit.

Our URF bridges teacher-guided distillation and reward-driven RL through a sample-wise *reward-preference weight* $p_i$ (Eq. (8); Figure 3), indicating how strongly reward-derived supervision should be preferred over teacher-guided correction for sample $i$. Through the unified reward-adaptive objective (Eq. (7)), the same $p_i$ determines *where* to optimize through $q_t^\star$ (Eq. (11)) and *what* to optimize through $\boldsymbol{v}^\star$ (Eq. (17)), for which we derive closed-form solutions.

## 3.1 PROBLEM SETUP AND UNIFIED REWARD-ADAPTIVE OBJECTIVE

At an online iteration, the data-collection policy $\boldsymbol{v}^{\text{old}}$ generates a clean endpoint $\boldsymbol{x}_{0,i}$ for prompt $\boldsymbol{c}_i$, and we observe its reward $R_{0,i} = R(\boldsymbol{x}_{0,i}, \boldsymbol{c}_i)$. A frozen task teacher provides $\boldsymbol{v}_T$. We seek an optimal adaptive query distribution $q_t^\star$ and an adaptive velocity target $\boldsymbol{v}^\star$ for each generated sample.

For a generic decision variable $y$, let $\mathcal{S}_T(y)$ and $\mathcal{S}_R(y)$ denote teacher-derived and reward-derived evidence scores, respectively. We use a unified reward-adaptive objective throughout the method,

$$\boxed{y^\star(p_i) = \arg\max_y \ \mathcal{J}_{p_i}(y), \qquad \mathcal{J}_{p_i}(y) = (1-p_i)\mathcal{S}_T(y) + p_i\mathcal{S}_R(y)} \tag{7}$$

Here, the sample-wise reward-preference weight $p_i \in [0,1]$ controls the relative contribution of teacher-derived and reward-derived evidence. Our intuition is that a low-reward endpoint provides weak positive evidence for reward-driven optimization; moreover, its negative-sample update can provide only coarse directional supervision, as observed in Figure 2(b). We therefore rely more on teacher-guided correction in this regime. As the endpoint reward increases, the generated endpoint becomes increasingly informative as a positive target, motivating a gradual transition toward reward-derived supervision. Accordingly, we determine $p_i$ directly from the observed endpoint reward using a monotonic calibration:

$$p_i = \text{clip}\left(\frac{\widetilde{R}_{0,i} - R_L}{R_H - R_L}, 0, 1\right), \tag{8}$$

where $\widetilde{R}_{0,i}$ denotes the task-normalized endpoint reward and $R_L$ and $R_H$ are lower and upper reward anchors. The anchors are defined by the corresponding rewards from the initial student and the frozen teacher, respectively. Thus, $p_i$ measures relative reward progress from the initial-policy anchor: small $p_i$ favors teacher supervision, whereas large $p_i$ favors reward-derived endpoint supervision. A joint policy-level interpretation of the coupled query and final velocity blocks is provided in Appendix A.4, where the velocity target is the exact conditional minimizer and the query differs from Eq. (11) only by a finite-step disagreement tilt.

### 3.2 WHERE TO OPTIMIZE: REWARD-ADAPTIVE QUERYING

At diffusion time $t$, let $q_{\mathrm{old},t}$ denote the state distribution induced by the current student rollout, and let $q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0) = \mathcal{N}(\alpha_t \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I})$ denote the forward-noised distribution of the generated endpoint. The two supervision sources are naturally associated with different query distributions: teacher-guided on-policy correction operates on student-visited states, whereas reward-derived endpoint supervision is constructed from forward-noised states of the generated endpoint. We therefore instantiate Eq. (7) as

$$\mathcal{S}_T(q) = -D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{old},t}\right), \qquad \mathcal{S}_R(q) = -D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{fwd},t}\right). \tag{9}$$

Maximizing Eq. (7) is equivalent to minimizing the weighted KL objective

$$q_t^\star = \arg\min_q \left[(1-p_i) D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{old},t}\right) + p_i D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{fwd},t}\right)\right]. \tag{10}$$

Solving this variational problem gives

$$q_t^\star(\boldsymbol{z}) = \frac{1}{Z_t} q_{\mathrm{old},t}(\boldsymbol{z})^{1-p_i} q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0)^{p_i}, \tag{11}$$

where $Z_t$ is the normalizing constant.

Motivated by this population-level barycenter, we use the following OU-type kernel for sample-wise querying. It coincides with Eq. (11) in the shared-covariance Gaussian case and, for arbitrary on-policy marginals, provides distribution-free contraction toward $q_{\mathrm{fwd},t}$ (Appendix A.1).

$$\boxed{\boldsymbol{z}_t^q = \boldsymbol{\kappa}_t + (1-p_i)\left(\boldsymbol{z}_t^{\mathrm{old}} - \boldsymbol{\kappa}_t\right) + \sqrt{2p_i - p_i^2}\,\sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\kappa}_t = \alpha_t \boldsymbol{x}_0,} \tag{12}$$

with fresh $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Thus, the final query continuously transitions from the student-visited state at $p_i = 0$ to a fresh forward-noised state of the generated endpoint at $p_i = 1$.

### 3.3 WHAT TO OPTIMIZE: REWARD-ADAPTIVE VELOCITY TARGET

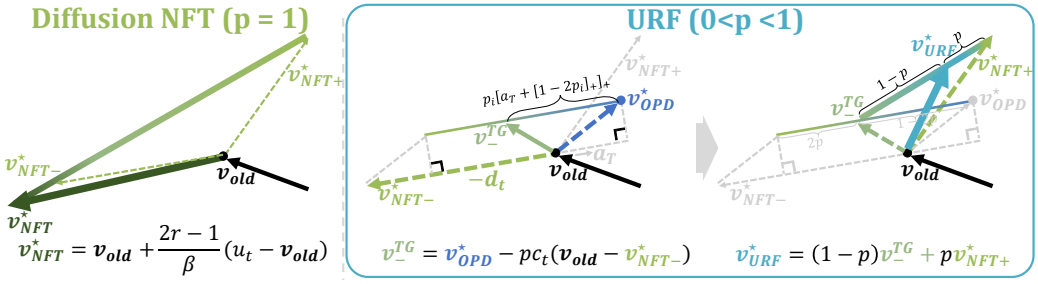



Figure 4: Velocity construction for negative-sample updates in DiffusionNFT and URF. DiffusionNFT handles a negative sample by simply pushing the velocity in the opposite direction. URF instead combines the teacher velocity $\boldsymbol{v}^\star_{\mathrm{OPD}}$ with the reward-preference weight $p_i$ to determine an adaptive correction direction, avoiding a purely repulsive update and exploiting teacher guidance.

We next apply the reward-adaptive objective in Eq. (7) to determine the velocity target at the adaptive query $\boldsymbol{z}_t^q$. Motivated by the coarse directional supervision of negative samples in Figure 2(b), we use negative reward feedback only to constrain the teacher along the reward-supported direction:

$$\boxed{a_i^\star = \arg\min_a \left[\frac{1-p_i}{2}(a - a_T)^2 + \frac{p_i}{2}\left[a + [1-2p_i]_+\right]_+^2\right].} \tag{13}$$

Here,

$$\boldsymbol{d}_t = \boldsymbol{v}^{\mathrm{old}} - \boldsymbol{v}^\star_{\mathrm{NFT}-} = \boldsymbol{v}^\star_{\mathrm{NFT}+} - \boldsymbol{v}^{\mathrm{old}}, \qquad a_T = \frac{\langle \boldsymbol{v}_T - \boldsymbol{v}^{\mathrm{old}}, \boldsymbol{d}_t \rangle}{\|\boldsymbol{d}_t\|_2^2 + \varepsilon_v}, \tag{14}$$

where $\boldsymbol{d}_t$ is the reward-supported direction and $a_T$ measures the teacher displacement along this direction. The first term in Eq. (13) keeps the correction close to the teacher-derived target, while the second term penalizes components that violate the negative-reward constraint (Figure 4).

Solving Eq. (13) gives

$$c_t = \left[a_T + [1 - 2p_i]_+\right]_+, \qquad \boxed{\boldsymbol{v}_-^{\mathrm{TG}} = \boldsymbol{v}_T - p_i c_t \boldsymbol{d}_t.} \tag{15}$$

We then define the reward- and teacher-derived velocity evidence as quadratic scores centered at their corresponding targets,

$$\mathcal{S}_R^v(\boldsymbol{v}) = -\left\|\boldsymbol{v} - \boldsymbol{v}^\star_{\mathrm{NFT}+}\right\|_2^2, \qquad \mathcal{S}_T^v(\boldsymbol{v}) = -\left\|\boldsymbol{v} - \boldsymbol{v}_-^{\mathrm{TG}}\right\|_2^2. \tag{16}$$

Substituting these scores into Eq. (7) yields the closed-form adaptive velocity target

$$\boxed{\boldsymbol{v}^\star = p_i\, \boldsymbol{v}^\star_{\mathrm{NFT}+} + (1 - p_i)\boldsymbol{v}_-^{\mathrm{TG}}.} \tag{17}$$

Equivalently,

$$\boxed{\boldsymbol{v}^\star = p_i\, \boldsymbol{v}^\star_{\mathrm{NFT}+} + (1 - p_i)\left[\boldsymbol{v}_T - p_i\, c_t\left(\boldsymbol{v}^{\mathrm{old}} - \boldsymbol{v}^\star_{\mathrm{NFT}-}\right)\right].} \tag{18}$$

For $p_i < 1/2$, the TG-negative correction bounds the additional attraction toward the NFT-positive target beyond the teacher-supported component by $O(p_i^2)$; see Appendix A.3. The detailed derivation of the teacher-guided corrective target is provided in Appendix A.2. Training remains ordinary flow-matching regression,

$$\mathcal{L}_{\mathrm{ours}} = \mathbb{E}\left[w(t)\left\|\boldsymbol{v}_\theta(\boldsymbol{z}_t^q, t, \boldsymbol{c}) - \boldsymbol{v}^\star\right\|_2^2\right], \tag{19}$$

where $w(t)$ is a constant weight.

When $p_i \to 0$, the query approaches the student-visited state and $\boldsymbol{v}^\star \to \boldsymbol{v}_T$, recovering DiffusionOPD. When $p_i \to 1$, the reward saturates the upper anchor, so the query approaches the forward-noised endpoint, no negative-reward branch is retained, and $\boldsymbol{v}^\star \to \boldsymbol{v}^\star_{\mathrm{NFT}+}$, recovering DiffusionNFT's fully positive limit ($r = 1$).

### 3.4 Optimization algorithm

Algorithm 1 summarizes the overall training procedure. At each round, the current data-collection policy generates samples and their rewards, from which we compute the sample-wise reward-preference weight $p_i$. The same $p_i$ is then used to construct the adaptive query and velocity target, and the student is optimized by velocity regression at the resulting query locations. The data-collection policy is updated by EMA after each round.

For clarity, Algorithm 1 presents the single-timestep form; in the full-step experiments, we evaluate and uniformly average the nine selected non-terminal transitions of each rollout, as detailed in Appendix B.1.

For multi-reward optimization, multiple reward functions and their corresponding teachers are incorporated within the same update step. For few-step distillation, $\boldsymbol{v}_T$ is taken as the final velocity produced by a multi-step teacher rollout.

## 4 Experiments

We evaluate multi-reward joint training, single-reward head-to-head optimization, distillation to fewer-step students, and component ablations.

### 4.1 Experimental setup

We use Stable Diffusion 3.5 Medium (SD3.5-M) (Esser et al., 2024) at $512 \times 512$ resolution and largely follow the training and evaluation settings of Flow-GRPO (Liu et al., 2025), DiffusionNFT (Zheng et al., 2026), and DiffusionOPD (Li et al., 2026).

**Algorithm 1:** Unified Reward-adaptive Framework (URF)

**Input :** Prompt set $\mathcal{C}$; data-collection policy $\boldsymbol{v}^{\text{old}}$; trainable policy $\boldsymbol{v}_\theta$; teacher $\boldsymbol{v}_T$; reward function $R$.
**Output:** Optimized policy $\boldsymbol{v}_\theta$.
Initialize $\boldsymbol{v}^{\text{old}}$ and $\boldsymbol{v}_\theta$ from the same reference model, and data buffer $\mathcal{D} \leftarrow \emptyset$;
**for** *each training round* **do**
  **for** *each sampled prompt* $\boldsymbol{c}_i$ **do**
    Roll out $\boldsymbol{v}^{\text{old}}$ to obtain endpoint $\boldsymbol{x}_{0,i}$; `// Data collection`
    Evaluate $R_{0,i} = R(\boldsymbol{x}_{0,i}, \boldsymbol{c}_i)$ and compute $p_i$ by Eq. (8);
    Sample diffusion time $t_i$ and record the student-visited state $\boldsymbol{z}_{t_i}^{\text{old}}$;
    Store $(\boldsymbol{c}_i, \boldsymbol{x}_{0,i}, R_{0,i}, p_i, t_i, \boldsymbol{z}_{t_i}^{\text{old}})$ in $\mathcal{D}$;
  Initialize the round loss $\mathcal{L}_{\text{total}} \leftarrow 0$;
  **for** *each mini-batch* $\mathcal{B} \subset \mathcal{D}$ **do**
    Sample fresh $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    Form the adaptive query

$$\boldsymbol{z}_{t_i}^{q} = \boldsymbol{\kappa}_{t_i} + (1 - p_i)\left(\boldsymbol{z}_{t_i}^{\text{old}} - \boldsymbol{\kappa}_{t_i}\right) + \sqrt{2p_i - p_i^2}\,\sigma_{t_i}\boldsymbol{\epsilon}_i, \qquad \boldsymbol{\kappa}_{t_i} = \alpha_{t_i}\boldsymbol{x}_{0,i} \tag{12}$$

    Evaluate $\boldsymbol{v}^{\text{old}}$, $\boldsymbol{v}_T$, and $\boldsymbol{v}^{\star}_{\text{NFT}+}$ at $(\boldsymbol{z}_{t_i}^{q}, t_i, \boldsymbol{c}_i)$, and compute $c_{t_i}$ by Eq. (15);
    Form the adaptive velocity target

$$\boldsymbol{v}_i^{\star} = p_i\,\boldsymbol{v}^{\star}_{\text{NFT}+} + (1 - p_i)\left[\boldsymbol{v}_T - p_i c_{t_i}\left(\boldsymbol{v}^{\star}_{\text{NFT}+} - \boldsymbol{v}^{\text{old}}\right)\right] \tag{18}$$

    Accumulate

$$\mathcal{L}_{\text{total}} \mathrel{+}= \sum_{i \in \mathcal{B}} w(t_i)\left\|\boldsymbol{v}_\theta(\boldsymbol{z}_{t_i}^{q}, t_i, \boldsymbol{c}_i) - \boldsymbol{v}_i^{\star}\right\|_2^2$$

  Update $\theta$ by performing one backward pass on $\mathcal{L}_{\text{total}}$ and taking an optimizer step;
  Update the data-collection policy $\boldsymbol{v}^{\text{old}} \leftarrow \text{EMA}(\boldsymbol{v}^{\text{old}}, \boldsymbol{v}_\theta)$, and clear $\mathcal{D} \leftarrow \emptyset$;

**Reward Models.** We evaluate two rule-based rewards—GenEval (Ghosh et al., 2023) and OCR following Flow-GRPO (Liu et al., 2025)—and six model-based rewards: PickScore (Kirstain et al., 2023), CLIPScore (Hessel et al., 2021), HPSv2.1 (Wu et al., 2023), Aesthetic Score (Schuhmann, 2022), ImageReward (Xu et al., 2023), and UnifiedReward (Wang et al., 2025).

**Prompt Datasets.** For GenEval and OCR, we use the corresponding training and evaluation prompt sets from Flow-GRPO (Liu et al., 2025). For the remaining rewards, we train on Pick-a-Pic (Kirstain et al., 2023) and evaluate on DrawBench (Saharia et al., 2022). All methods use the same official prompt splits for fair comparison.

**Training and Evaluation.** We fine-tune the model with LoRA (Hu et al., 2022) ($\alpha = 64$, rank $r = 32$). We use a single rollout per prompt with 72 prompts per round (72 samples in total), the same sample budget as the $8\times9$ allocation under which DiffusionNFT collapses (Figure 2(c)). We use the three task-specific teachers released with DiffusionOPD (Li et al., 2026): the GenEval Teacher, OCR Teacher, and Aes Teacher. The Aes Teacher is trained on an equally weighted mixture of PickScore, CLIPScore, and HPSv2.1 (Li et al., 2026) and thus serves as the PickScore teacher. We use 30 rollout sampling steps for visualization, and full-step evaluations use a 40-step first-order ODE sampler.

### 4.2 Multi-reward joint training

We jointly optimize GenEval, OCR, and PickScore using their corresponding task-specific teachers. Each round contains 24 prompts per task (72 in total), with each sample independently constructing its own $p_i$, adaptive query, and velocity target; the three task losses are equally weighted. Full implementation details are provided in Appendix B.2.

Table 1: Multi-reward evaluation on SD3.5-M. Average is the mean of per-metric min–max normalized scores (SD3.5-M w/o CFG = 0). **Bold** and underline denote the best and second-best values in each column.

| Model | Rule-Based | | | Model-Based | | | | | Average |
|---|---|---|---|---|---|---|---|---|---|
| | GenEval | OCR | PickScore | ClipScore | HPSv2.1 | Aesthetic | ImgRwd | UniRwd | |
| SD3.5-M (w/o CFG) | 0.204 | 0.110 | 20.611 | 0.238 | 0.199 | 5.123 | -0.599 | 2.028 | 0.000 |
| + CFG | 0.630 | 0.545 | 22.422 | 0.281 | 0.281 | 5.426 | 0.863 | 3.110 | 0.570 |
| GenEval Teacher | 0.958 | 0.510 | 22.111 | 0.286 | 0.266 | 5.240 | 0.533 | 3.080 | 0.562 |
| OCR Teacher | 0.654 | 0.927 | 22.231 | 0.288 | 0.272 | 5.281 | 0.879 | 3.141 | 0.617 |
| Aes Teacher | 0.474 | 0.598 | **24.004** | 0.296 | **0.345** | **6.214** | 1.501 | 3.576 | 0.863 |
| Multi-Task GRPO-Guard | 0.898 | 0.942 | 23.331 | 0.295 | 0.312 | 5.592 | 1.293 | 3.329 | 0.825 |
| Multi-Task NFT | 0.946 | 0.951 | 22.490 | 0.293 | 0.288 | 5.307 | 1.203 | 3.236 | 0.732 |
| DiffusionOPD | 0.955 | 0.938 | 24.000 | **0.297** | 0.341 | 6.186 | **1.514** | 3.530 | 0.984 |
| **Ours** | **0.963** | **0.963** | 24.003 | **0.297** | 0.344 | 6.190 | 1.505 | **3.578** | **0.996** |

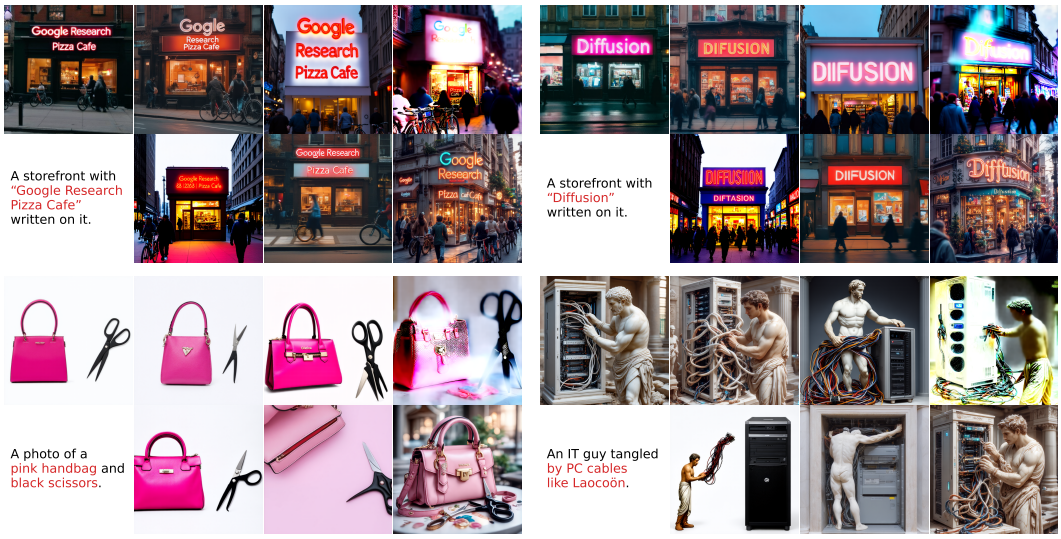



Figure 5: Qualitative comparisons with multi-task baselines and single-task teachers. Top row, from left to right: URF (ours), DiffusionOPD, Multi-Task NFT, and DiffusionNFT MultiReward. Bottom row: input prompt, GenEval Teacher, OCR Teacher, and Aes Teacher.

Table 1 summarizes the multi-reward results. Specialized teachers perform strongly only on their target rewards, while Multi-Task GRPO-Guard (Wang et al., 2026), Multi-Task NFT (Zheng et al., 2026), and DiffusionOPD (Li et al., 2026) remain below them on most reward objectives.

URF achieves either the best or second-best result across all eight metrics and the highest normalized average score of 0.996. It also reaches higher reward levels while converging approximately 1.7–7.9× faster (Figure 1(a)–(b)). Qualitative comparisons are shown in Figure 5.

### 4.3 Single-Reward Optimization and Few-Step Distillation

We first separately optimize GenEval, PickScore and OCR score and compare DiffusionNFT, DiffusionOPD, and our method. As shown in Figure 6(a–c), URF achieves faster convergence and higher final rewards, with a $1.8\times$–$10.4\times$ speedup over DiffusionOPD.

We further evaluate whether the learned reward-optimized behavior can be transferred to a fewer-step student. For OCR, we use the URF checkpoint at optimizer step 495 as a frozen multi-step teacher and the 4-step SenseFlow SD3.5-M model (Ge et al., 2026) as the student. DiffusionOPD and URF distill from this teacher, and all methods use the same OCR reward. As shown in Figure 6(d), URF converges faster and reaches a higher OCR reward during 4-step student optimization.

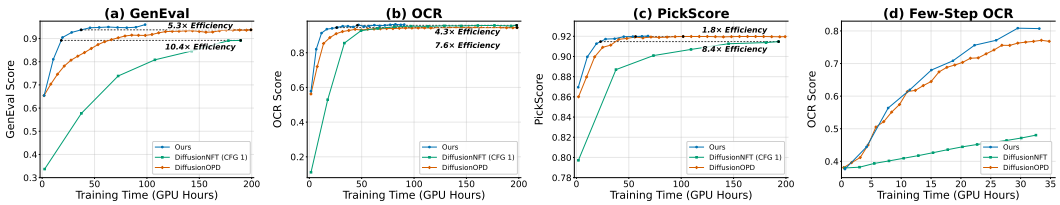



Figure 6: Single-reward optimization and few-step distillation. (a–c) Training curves of URF, DiffusionOPD, and DiffusionNFT on GenEval, PickScore, and OCR, respectively; ×Efficiency denotes wall-clock speedup to reach DiffusionOPD's peak reward. (d) OCR optimization of a 4-step SenseFlow student using URF, DiffusionOPD, and DiffusionNFT.

### 4.4 Ablation Studies

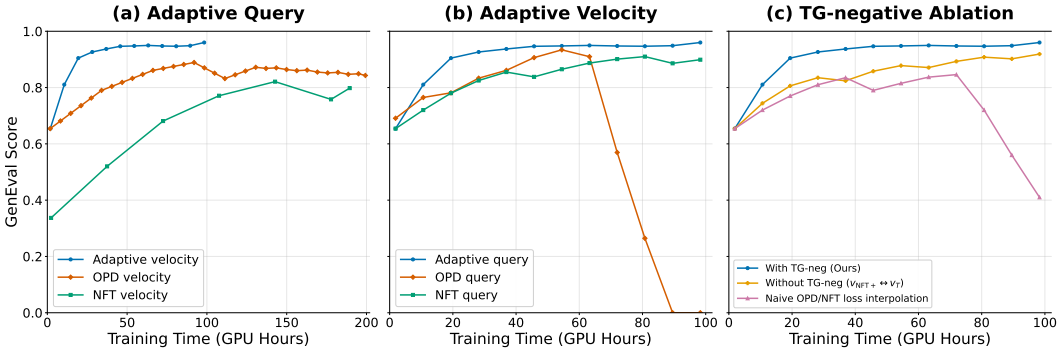



Figure 7: Ablation on GenEval optimization. (a) We compare OPD, NFT, and adaptive velocity losses under the same adaptive query. (b) We compare OPD, NFT, and adaptive queries under the same adaptive velocity loss. (c) We evaluate the effect of TG-negative guidance when jointly adapting the query and loss, comparing the full target against plain OPD/NFT loss interpolation and a variant without the TG-negative correction ($\boldsymbol{v}^\star = p_i \boldsymbol{v}^\star_{\mathrm{NFT}+} + (1-p_i)\boldsymbol{v}_T$).

We isolate each component by fixing it to its OPD/NFT limit or removing the correction term (Figure 7). Under a shared adaptive query (a) or velocity target (b), the adaptive counterpart consistently outperforms its OPD and NFT limits, and the OPD query in (b) leads to training collapse. In (c), we compare the full URF with two variants: removing the TG-negative correction ($c_t = 0$), i.e., $\boldsymbol{v}^\star = p_i \boldsymbol{v}^\star_{\mathrm{NFT}+} + (1-p_i)\boldsymbol{v}_T$, and directly interpolating the OPD and NFT losses. The full URF performs best, whereas direct loss interpolation collapses.

## 5 Conclusion

We observed that a naive OPD→NFT transition is difficult to optimize reliably: manual switching is sensitive to the handoff schedule, can cause post-switch degradation or late-stage collapse, and becomes particularly fragile under limited prompt diversity (Figure 2). Motivated by the negative-sample directional deficit in Figure 2(b), we introduced URF, which retains teacher-guided correction when reward supervision is coarse while progressively transitioning toward reward-driven optimization as endpoint rewards improve. A single reward-adaptive objective jointly determines *where* to optimize and *what* velocity target to follow, eliminating explicit stage-wise switching. Across multi-reward, single-reward, and few-step settings, URF consistently achieves faster convergence and strong final performance, demonstrating that teacher-guided distillation and reward-driven RL can be unified within a single sample-adaptive diffusion post-training framework. Future work may further improve this framework by learning $p_i$ directly, for example with a predictor that estimates which supervision signal is more likely to yield greater reward improvement.

### AI USE STATEMENT

Generative AI tools were used sparingly to assist with language polishing, LaTeX formatting, and presentation of author-written material. We did not use AI-generated outputs as experimental results or evaluation evidence. All AI-assisted text and formatting were reviewed and revised by the authors, who take full responsibility for the final content of the paper.

### ETHICS STATEMENT

This work studies post-training methods for text-to-image generative models using publicly available models, prompt datasets, and reward models, and does not involve the collection of new human-subject data or personally identifiable information. As with existing text-to-image generative models, the resulting models may inherit biases from pretrained models, training data, or reward models and may produce undesirable or misleading content. Our method is not intended to eliminate these broader risks, and deployment should therefore follow appropriate model-safety and content-use safeguards.

### REPRODUCIBILITY STATEMENT

We provide the experimental setup, prompt datasets, reward models, training and evaluation protocol, and model configuration in Section 4.1, including the use of SD3.5-M at $512 \times 512$ resolution, LoRA with rank 32 and scaling factor $\alpha = 64$, 72 rollout samples per round, and a 40-step first-order ODE sampler for full-step evaluation. Complete derivations of the proposed query and velocity constructions are provided in Appendix A. We additionally submit anonymized source code, including training and evaluation scripts, as supplementary material to facilitate reproduction of our results.

## REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations*, 2024. URL https://mlanthology.org/iclr/2024/agarwal2024iclr-onpolicy/.

Dominique Bakry, Ivan Gentil, and Michel Ledoux. *Analysis and Geometry of Markov Diffusion Operators*, volume 348 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2014. doi: 10.1007/978-3-319-00227-9.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=pBN8CgrDhE.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12606–12633. PMLR, 2024. URL https://proceedings.mlr.press/v235/esser24a.html.

Xingtong Ge, Xin Zhang, Tongda Xu, Yi Zhang, Xinjie Zhang, Yan Wang, and Jun Zhang. Senseflow: Scaling distribution matching for flow-based text-to-image distillation. In *International Conference on Learning Representations*, 2026.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems*, volume 36, pp. 52132–52152, 2023. doi: 10.52202/075280-2270.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.595. URL https://aclanthology.org/2021.emnlp-main.595/.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 36652–36663, 2023. doi: 10.52202/075280-1594.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. Diffusionopd: A unified perspective of on-policy distillation in diffusion models. *SIGGRAPH Asia 2026 Conference Papers*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1362.

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

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Asynchronous RLHF: Faster and More Efficient Off-Policy RL for Language Models. In *International Conference on Learning Representations*, 2025. URL https://mlanthology.org/iclr/2025/noukhovitch2025iclr-asynchronous/.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Raphael Gontijo-Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, pp. 36479–36494, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

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

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. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Jing Wang, Jiajun Liang, Jie Liu, Henglin Liu, Gongye Liu, Jun Zheng, Wanyuan Pang, Ao Ma, Zhenyu Xie, Xintao Wang, Meng Wang, Pengfei Wan, and Xiaodan Liang. Grpo-guard: Mitigating implicit over-optimization in flow matching via regulated clipping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5988–5998, June 2026.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation. *arXiv preprint arXiv:2503.05236*, 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.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for RLHF under KL-constraint. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54715–54754. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xiong24a.html.

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*, volume 36, pp. 15903–15935, 2023. doi: 10.52202/075280-0700.

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.

# A Derivations

## A.1 Reward-adaptive query derivation

We first derive the variational query distribution in Eq. (11). For a fixed sample $i$ and diffusion time $t$, Eq. (7) uses

$$\mathcal{S}_T(q) = -D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{old},t}\right), \qquad \mathcal{S}_R(q) = -D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{fwd},t}\right). \tag{20}$$

Hence maximizing the unified objective is equivalent to

$$q_t^\star = \arg\min_q \left[(1-p_i) D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{old},t}\right) + p_i D_{\mathrm{KL}}\left(q \,\|\, q_{\mathrm{fwd},t}\right)\right]. \tag{21}$$

Expanding the KL terms gives

$$\begin{aligned}\mathcal{F}(q) = (1-p_i) \int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{q_{\mathrm{old},t}(\boldsymbol{z})}\, d\boldsymbol{z} \\ + p_i \int q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0)}\, d\boldsymbol{z}.\end{aligned} \tag{22}$$

Introducing a Lagrange multiplier for $\int q(\boldsymbol{z}) d\boldsymbol{z} = 1$, the stationary condition is

$$\frac{\delta}{\delta q}\left[\mathcal{F}(q) + \lambda\left(\int q(\boldsymbol{z}) d\boldsymbol{z} - 1\right)\right] = 0, \tag{23}$$

which yields

$$\log q_t^\star(\boldsymbol{z}) = (1-p_i) \log q_{\mathrm{old},t}(\boldsymbol{z}) + p_i \log q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0) + \mathrm{const}. \tag{24}$$

Exponentiating and normalizing gives

$$q_t^\star(\boldsymbol{z}) = \frac{1}{Z_t} q_{\mathrm{old},t}(\boldsymbol{z})^{1-p_i} q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0)^{p_i}, \tag{25}$$

which is Eq. (11). This population-level variational form does not require Gaussian marginals, but assumes compatible source measures for which the weighted KL objective is finite. The sample-wise construction below remains well defined even for singular realized states, including point masses induced by deterministic ODE rollouts.

**Variance-preserving sample-wise realization.** The full state marginal induced by a nonlinear student rollout need not be Gaussian. We therefore do not impose a Gaussian assumption on $q_{\mathrm{old},t}$ in the implementation. Instead, conditioned on the realized on-policy state $\boldsymbol{z}_t^{\mathrm{old}}$, we use a Gaussian Markov kernel with an Ornstein–Uhlenbeck-type contraction toward the forward-process center

$$\boldsymbol{\kappa}_t = \alpha_t \boldsymbol{x}_0. \tag{26}$$

Let

$$\rho_i = 1 - p_i. \tag{27}$$

We define

$$K_{p_i,t}\left(\boldsymbol{z} \mid \boldsymbol{z}_t^{\mathrm{old}}, \boldsymbol{x}_0\right) = \mathcal{N}\left(\boldsymbol{\kappa}_t + \rho_i\left(\boldsymbol{z}_t^{\mathrm{old}} - \boldsymbol{\kappa}_t\right),\ (1-\rho_i^2)\sigma_t^2 \boldsymbol{I}\right). \tag{28}$$

Sampling from this kernel gives

$$\begin{aligned}\boldsymbol{z}_t^q &= \boldsymbol{\kappa}_t + (1-p_i)\left(\boldsymbol{z}_t^{\mathrm{old}} - \boldsymbol{\kappa}_t\right) \\ &\quad + \sqrt{1-(1-p_i)^2}\,\sigma_t \boldsymbol{\epsilon} \\ &= \boldsymbol{\kappa}_t + (1-p_i)\left(\boldsymbol{z}_t^{\mathrm{old}} - \boldsymbol{\kappa}_t\right) + \sqrt{2p_i - p_i^2}\,\sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),\end{aligned} \tag{29}$$

recovering Eq. (12).

**Distribution-free contraction properties.** The sample-wise construction above is defined conditionally on an arbitrary realized on-policy state and therefore does not require $q_{\mathrm{old},t}$ to be Gaussian. In particular, conditioned on a realized deterministic ODE rollout, $\boldsymbol{z}_t^{\mathrm{old}}$ may be viewed as a point mass, for which the construction remains well defined.

Let

$$\rho_i = 1 - p_i, \qquad \nu_t = q_{\mathrm{fwd},t} = \mathcal{N}(\boldsymbol{\kappa}_t, \sigma_t^2 \boldsymbol{I}), \tag{30}$$

and let $K_{p_i,t}$ denote the Markov kernel in Eq. (28).

**Endpoint exactness and invariance.** At the two endpoints,

$$p_i = 0 \quad \Longrightarrow \quad \boldsymbol{z}_t^q = \boldsymbol{z}_t^{\mathrm{old}}, \tag{31}$$

whereas

$$p_i = 1 \quad \Longrightarrow \quad \boldsymbol{z}_t^q = \boldsymbol{\kappa}_t + \sigma_t \boldsymbol{\epsilon} \sim \nu_t. \tag{32}$$

Moreover, $\nu_t$ is invariant under the kernel for every $p_i$:

$$\nu_t K_{p_i,t} = \nu_t. \tag{33}$$

Indeed, if $\boldsymbol{z} - \boldsymbol{\kappa}_t \sim \mathcal{N}(\boldsymbol{0}, \sigma_t^2 \boldsymbol{I})$, then

$$\rho_i(\boldsymbol{z} - \boldsymbol{\kappa}_t) + \sqrt{1-\rho_i^2}\,\sigma_t \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_t^2 \boldsymbol{I}). \tag{34}$$

**Wasserstein contraction.** For any distribution $q$ with finite second moment,

$$W_2\left(qK_{p_i,t}, \nu_t\right) \le (1-p_i) W_2\left(q, \nu_t\right). \tag{35}$$

To see this, let $(\boldsymbol{X}, \boldsymbol{Y})$ be an optimal coupling of $q$ and $\nu_t$, and use the same independent $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for both kernel transitions:

$$\boldsymbol{X}' = \boldsymbol{\kappa}_t + \rho_i(\boldsymbol{X} - \boldsymbol{\kappa}_t) + \sqrt{1-\rho_i^2}\,\sigma_t \boldsymbol{\epsilon}, \tag{36}$$

$$\boldsymbol{Y}' = \boldsymbol{\kappa}_t + \rho_i(\boldsymbol{Y} - \boldsymbol{\kappa}_t) + \sqrt{1-\rho_i^2}\,\sigma_t \boldsymbol{\epsilon}. \tag{37}$$

By Eq. (33), $\boldsymbol{Y}' \sim \nu_t$, while

$$\boldsymbol{X}' - \boldsymbol{Y}' = \rho_i(\boldsymbol{X} - \boldsymbol{Y}). \tag{38}$$

Therefore,

$$W_2^2(qK_{p_i,t}, \nu_t) \le \rho_i^2 W_2^2(q, \nu_t), \tag{39}$$

which proves Eq. (35). This statement applies directly to point-mass inputs and therefore also covers realized deterministic ODE states.

**Relative-entropy contraction.** When $D_{\mathrm{KL}}(q\|\nu_t) < \infty$, the same Gaussian OU kernel satisfies the standard entropy contraction

$$D_{\mathrm{KL}}\left(qK_{p_i,t}\|\nu_t\right) \le (1-p_i)^2 D_{\mathrm{KL}}\left(q\|\nu_t\right), \tag{40}$$

which follows from exponential entropy decay of the Gaussian Ornstein–Uhlenbeck semigroup via the Gaussian logarithmic Sobolev inequality (Bakry et al., 2014). We invoke Eq. (40) only when the initial KL is finite; the Wasserstein statement above is the relevant guarantee for singular inputs such as point masses.

**Shared-covariance Gaussian tightness.** The Gaussian case is therefore not an assumption required by the implementation, but a tight consistency case. If

$$q_{\mathrm{old},t} = \mathcal{N}\left(\boldsymbol{\mu}_{\mathrm{old},t}, \sigma_t^2 \boldsymbol{I}\right), \tag{41}$$

then applying the kernel gives

$$q_{\mathrm{old},t}K_{p_i,t} = \mathcal{N}\left((1-p_i)\boldsymbol{\mu}_{\mathrm{old},t} + p_i\boldsymbol{\kappa}_t, \sigma_t^2\boldsymbol{I}\right). \tag{42}$$

For two source Gaussians with this shared covariance, Eq. (42) is exactly the geometric KL barycenter in Eq. (11). Moreover, both Eqs. (35) and (40) hold with equality. Thus, the shared-covariance Gaussian case serves as a tightness check, while the implemented query construction itself remains valid for arbitrary on-policy state distributions.

### A.2 Reward-adaptive velocity derivation

We next derive the teacher-guided corrective target $\boldsymbol{v}_-^{\mathrm{TG}}$ and the final adaptive velocity target $\boldsymbol{v}^\star$.

For a fixed adaptive query $\boldsymbol{z}_t^q$, define the reward-supported direction

$$\boldsymbol{d}_t = \boldsymbol{v}^{\mathrm{old}} - \boldsymbol{v}^\star_{\mathrm{NFT}-} = \boldsymbol{v}^\star_{\mathrm{NFT}+} - \boldsymbol{v}^{\mathrm{old}}. \tag{43}$$

The teacher displacement along this direction is measured by

$$a_T = \frac{\left\langle \boldsymbol{v}_T - \boldsymbol{v}^{\mathrm{old}}, \boldsymbol{d}_t \right\rangle}{\|\boldsymbol{d}_t\|_2^2 + \varepsilon_v}, \tag{44}$$

as in Eq. (14). Here $\varepsilon_v$ is a small numerical stabilizer and is distinct from the reward-normalization constant $\varepsilon_R$ used below.

We restrict the reward-dependent teacher correction to the one-dimensional affine family

$$\boldsymbol{v}(a) = \boldsymbol{v}_T + (a - a_T)\boldsymbol{d}_t. \tag{45}$$

Thus, varying $a$ changes only the component of the teacher target along $\boldsymbol{d}_t$, while preserving its orthogonal component.

For low-reward samples, negative reward evidence provides a one-sided directional constraint rather than a complete velocity target. We encode this using the reward-dependent margin

$$m_i = \left[1 - 2p_i\right]_+, \tag{46}$$

with the soft constraint

$$a \le -m_i. \tag{47}$$

Violation of this constraint is penalized by $[a + m_i]_+^2$. Combining teacher proximity and the negative-reward constraint gives

$$a_i^\star = \arg\min_a \left[\frac{1-p_i}{2}(a - a_T)^2 + \frac{p_i}{2}\left[a + m_i\right]_+^2\right], \tag{48}$$

which is Eq. (13).

This one-dimensional convex problem has a closed-form solution. If $a_T \le -m_i$, the teacher already satisfies the constraint and the hinge term is inactive:

$$a_i^\star = a_T. \tag{49}$$

Otherwise, for $a_T > -m_i$, the hinge is active and the first-order condition is

$$(1-p_i)(a-a_T)+p_i(a+m_i)=0. \tag{50}$$

Therefore,

$$a_i^\star=(1-p_i)a_T-p_im_i=a_T-p_i(a_T+m_i). \tag{51}$$

Both cases can be written as

$$a_i^\star=a_T-p_i\left[a_T+m_i\right]_+. \tag{52}$$

Defining

$$c_t=\left[a_T+\left[1-2p_i\right]_+\right]_+, \tag{53}$$

we obtain

$$a_i^\star=a_T-p_ic_t. \tag{54}$$

Substituting this solution into Eq. (45) gives

$$\begin{aligned}\boldsymbol{v}_-^{\mathrm{TG}}&=\boldsymbol{v}_T+(a_i^\star-a_T)\boldsymbol{d}_t\\&=\boldsymbol{v}_T-p_ic_t\boldsymbol{d}_t,\end{aligned} \tag{55}$$

which is Eq. (15). The correction therefore modifies only the teacher component along the reward-supported direction.

We finally derive the adaptive velocity target from the unified objective. The teacher- and reward-derived velocity scores are

$$\mathcal{S}_T^v(\boldsymbol{v})=-\left\|\boldsymbol{v}-\boldsymbol{v}_-^{\mathrm{TG}}\right\|_2^2,\qquad \mathcal{S}_R^v(\boldsymbol{v})=-\left\|\boldsymbol{v}-\boldsymbol{v}_{\mathrm{NFT}+}^\star\right\|_2^2. \tag{56}$$

Maximizing their reward-adaptive combination is equivalent to

$$\boldsymbol{v}^\star=\arg\min_{\boldsymbol{v}}\left[(1-p_i)\left\|\boldsymbol{v}-\boldsymbol{v}_-^{\mathrm{TG}}\right\|_2^2+p_i\left\|\boldsymbol{v}-\boldsymbol{v}_{\mathrm{NFT}+}^\star\right\|_2^2\right]. \tag{57}$$

Taking the derivative with respect to $\boldsymbol{v}$ gives

$$(1-p_i)(\boldsymbol{v}-\boldsymbol{v}_-^{\mathrm{TG}})+p_i(\boldsymbol{v}-\boldsymbol{v}_{\mathrm{NFT}+}^\star)=0, \tag{58}$$

and therefore

$$\boxed{\boldsymbol{v}^\star=(1-p_i)\boldsymbol{v}_-^{\mathrm{TG}}+p_i\boldsymbol{v}_{\mathrm{NFT}+}^\star.} \tag{59}$$

This recovers Eq. (17). Substituting Eq. (15) gives

$$\boldsymbol{v}^\star=p_i\boldsymbol{v}_{\mathrm{NFT}+}^\star+(1-p_i)\left[\boldsymbol{v}_T-p_ic_t\left(\boldsymbol{v}^{\mathrm{old}}-\boldsymbol{v}_{\mathrm{NFT}-}^\star\right)\right], \tag{60}$$

which is Eq. (18).

At $p_i=0$, the correction vanishes, $\boldsymbol{v}_-^{\mathrm{TG}}=\boldsymbol{v}_T$, and $\boldsymbol{v}^\star=\boldsymbol{v}_T$, recovering DiffusionOPD. At $p_i=1$, the teacher branch receives zero weight and

$$\boldsymbol{v}^\star=\boldsymbol{v}_{\mathrm{NFT}+}^\star, \tag{61}$$

which coincides with the fully positive ($r=1$) branch of DiffusionNFT. This is intentional: $p_i=1$ denotes reward saturation at the upper anchor, so the sample provides positive rather than negative reward evidence. Together with the query limits above, the same reward-preference weight controls the transition of both the query and velocity target.

### A.3 Second-Order Attraction in the Low-Preference Regime

We characterize the role of the TG-negative correction when the reward-preference weight is small. Because $p_i$ is calibrated between the initial-policy reward anchor $R_L$ and the teacher reward anchor $R_H$, $p_i$ represents relative reward progress rather than absolute sample quality. In particular, $p_i>0$ indicates improvement over the lower initial-policy anchor whenever $R_H>R_L$.

For clarity, we analyze the idealized $\varepsilon_v=0$ form of Eq. (14); the positive $\varepsilon_v$ used in implementation serves only as a numerical stabilizer. Define the normalized component of the final update along the reward-supported direction $\boldsymbol{d}_t$ as

$$s_i=\frac{\left\langle\boldsymbol{v}^\star-\boldsymbol{v}^{\mathrm{old}},\boldsymbol{d}_t\right\rangle}{\|\boldsymbol{d}_t\|_2^2}. \tag{62}$$

Using

$$\boldsymbol{v}^{\star}_{\mathrm{NFT}+} - \boldsymbol{v}^{\mathrm{old}} = \boldsymbol{d}_t, \qquad \frac{\langle \boldsymbol{v}_T - \boldsymbol{v}^{\mathrm{old}}, \boldsymbol{d}_t \rangle}{\|\boldsymbol{d}_t\|_2^2} = a_T, \tag{63}$$

and Eq. (18), we obtain

$$s_i = p_i + (1-p_i)a_T - p_i(1-p_i)c_t. \tag{64}$$

Consider the low-preference regime $p_i < 1/2$, for which

$$m_i = 1 - 2p_i. \tag{65}$$

If the hinge is active, $a_T > -(1-2p_i)$, then

$$c_t = a_T + 1 - 2p_i. \tag{66}$$

Substituting into Eq. (64) gives

$$\begin{aligned} s_i &= p_i + (1-p_i)a_T - p_i(1-p_i)\left(a_T + 1 - 2p_i\right) \\ &= (1-p_i)^2 a_T + p_i^2(3-2p_i). \end{aligned} \tag{67}$$

If the hinge is inactive, then

$$c_t = 0, \qquad a_T \le -(1-2p_i), \tag{68}$$

and

$$s_i = p_i + (1-p_i)a_T. \tag{69}$$

Using the inactive-hinge condition,

$$s_i \le (1-p_i)^2 a_T + p_i^2(3-2p_i). \tag{70}$$

Therefore, in both cases,

$$\boxed{s_i \le (1-p_i)^2 a_T + p_i^2(3-2p_i) \le (1-p_i)^2 a_T + 3p_i^2.} \tag{71}$$

The first term is the component supported by the teacher along $\boldsymbol{d}_t$. The remaining attraction toward the NFT-positive target is at most $O(p_i^2)$. By contrast, directly interpolating the teacher and NFT-positive targets without the TG-negative correction would give

$$s_i^{\mathrm{naive}} = p_i + (1-p_i)a_T, \tag{72}$$

which contains a first-order $O(p_i)$ attraction toward the generated endpoint.

Thus, the margin $[1-2p_i]_+$ in Eq. (13) is not merely a hard rejection of low-reward samples. Instead, it suppresses the first-order reward attraction in the low-preference regime while preserving the component already supported by the teacher. This allows samples that improve upon the initial-policy reference to contribute reward information without allowing weak reward evidence to dominate the update direction.

### A.4 Joint Query–Transition KL Interpretation

We provide a joint policy-level interpretation of the adaptive query and final velocity target. This interpretation is conditioned on the teacher-guided corrective reference $\boldsymbol{v}_{-}^{\mathrm{TG}}$ defined in Eq. (15); namely, it unifies the query and final velocity blocks once the constrained teacher reference has been constructed.

For a fixed sample $i$ and diffusion time $t$, suppressing the conditioning variables for brevity, define two joint reference distributions

$$P_T(\boldsymbol{z}, \boldsymbol{x}') = q_{\mathrm{old},t}(\boldsymbol{z}) K_T(\boldsymbol{x}' \mid \boldsymbol{z}), \qquad P_R(\boldsymbol{z}, \boldsymbol{x}') = q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0) K_R(\boldsymbol{x}' \mid \boldsymbol{z}). \tag{73}$$

We use infinitesimal Gaussian transition references with shared covariance,

$$K_T(\boldsymbol{x}' \mid \boldsymbol{z}) = \mathcal{N}\big(\boldsymbol{z} + \Delta t\, \boldsymbol{v}_{-}^{\mathrm{TG}}(\boldsymbol{z}), \gamma_t^2 \Delta t\, \boldsymbol{I}\big), \tag{74}$$

$$K_R(\boldsymbol{x}' \mid \boldsymbol{z}) = \mathcal{N}\big(\boldsymbol{z} + \Delta t\, \boldsymbol{v}^{\star}_{\mathrm{NFT}+}(\boldsymbol{z}), \gamma_t^2 \Delta t\, \boldsymbol{I}\big). \tag{75}$$

Importantly, the teacher-side kernel is centered at the constrained teacher reference $\boldsymbol{v}_-^{\mathrm{TG}}$, rather than the unconstrained teacher velocity $\boldsymbol{v}_T$.

For a candidate joint distribution

$$Q(\boldsymbol{z}, \boldsymbol{x}') = q(\boldsymbol{z})\pi(\boldsymbol{x}' \mid \boldsymbol{z}), \tag{76}$$

consider the single weighted KL objective

$$Q^\star = \arg\min_Q \left[(1-p_i)D_{\mathrm{KL}}(Q\|P_T) + p_i D_{\mathrm{KL}}(Q\|P_R)\right]. \tag{77}$$

By the KL chain rule,

$$\begin{aligned}\mathcal{J}_{\mathrm{joint}}(Q) = (1-p_i)D_{\mathrm{KL}}\left(q\|q_{\mathrm{old},t}\right) + p_i D_{\mathrm{KL}}\left(q\|q_{\mathrm{fwd},t}\right) \\ + \mathbb{E}_{q(\boldsymbol{z})}\left[(1-p_i)D_{\mathrm{KL}}\left(\pi(\cdot \mid \boldsymbol{z})\|K_T(\cdot \mid \boldsymbol{z})\right)\right. \\ \left. + p_i D_{\mathrm{KL}}\left(\pi(\cdot \mid \boldsymbol{z})\|K_R(\cdot \mid \boldsymbol{z})\right)\right].\end{aligned} \tag{78}$$

Within the same shared-covariance Gaussian transition family, parameterize $\pi$ by velocity $\boldsymbol{v}$,

$$\pi_{\boldsymbol{v}}(\boldsymbol{x}' \mid \boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z} + \Delta t\, \boldsymbol{v}(\boldsymbol{z}), \gamma_t^2 \Delta t\, \boldsymbol{I}\right). \tag{79}$$

The conditional KL terms then reduce exactly to

$$\begin{aligned}\mathcal{J}_{\mathrm{joint}}(q, \boldsymbol{v}) = (1-p_i)D_{\mathrm{KL}}\left(q\|q_{\mathrm{old},t}\right) + p_i D_{\mathrm{KL}}\left(q\|q_{\mathrm{fwd},t}\right) \\ + \lambda_t \mathbb{E}_q\left[(1-p_i)\left\|\boldsymbol{v} - \boldsymbol{v}_-^{\mathrm{TG}}\right\|_2^2 + p_i\left\|\boldsymbol{v} - \boldsymbol{v}_{\mathrm{NFT}+}^\star\right\|_2^2\right],\end{aligned} \tag{80}$$

where

$$\lambda_t = \frac{\Delta t}{2\gamma_t^2}. \tag{81}$$

For every query state $\boldsymbol{z}$, minimizing the conditional block gives

$$\boldsymbol{v}^\star(\boldsymbol{z}) = (1-p_i)\boldsymbol{v}_-^{\mathrm{TG}}(\boldsymbol{z}) + p_i \boldsymbol{v}_{\mathrm{NFT}+}^\star(\boldsymbol{z}), \tag{82}$$

which is exactly Eq. (17).

Substituting this optimum back into Eq. (80) yields

$$\begin{aligned}\mathcal{J}_{\mathrm{joint}}(q, \boldsymbol{v}^\star) = (1-p_i)D_{\mathrm{KL}}\left(q\|q_{\mathrm{old},t}\right) + p_i D_{\mathrm{KL}}\left(q\|q_{\mathrm{fwd},t}\right) \\ + \lambda_t p_i (1-p_i)\mathbb{E}_q\left[\left\|\boldsymbol{v}_-^{\mathrm{TG}} - \boldsymbol{v}_{\mathrm{NFT}+}^\star\right\|_2^2\right].\end{aligned} \tag{83}$$

Therefore, the finite-step joint optimum over the query distribution is

$$q_{t,\mathrm{joint}}^\star(\boldsymbol{z}) \propto q_{\mathrm{old},t}(\boldsymbol{z})^{1-p_i} q_{\mathrm{fwd},t}(\boldsymbol{z} \mid \boldsymbol{x}_0)^{p_i} \exp\left[-\lambda_t p_i (1-p_i)\left\|\boldsymbol{v}_-^{\mathrm{TG}}(\boldsymbol{z}) - \boldsymbol{v}_{\mathrm{NFT}+}^\star(\boldsymbol{z})\right\|_2^2\right]. \tag{84}$$

For every $\lambda_t > 0$, the conditional velocity block is minimized exactly by Eq. (17). After substituting this optimum, the joint query solution is Eq. (84): the geometric barycenter in Eq. (11) multiplied by a disagreement-dependent exponential tilt. The log-density tilt is linear in $\lambda_t$ and therefore becomes negligible as $\lambda_t$ decreases. Thus, the joint objective recovers the adaptive velocity target exactly, while Eq. (11) corresponds to its zero-tilt query component.

At finite step size, the additional exponential factor downweights query states at which the constrained teacher and reward-derived transition references disagree strongly. URF does not explicitly use this finite-step tilt; incorporating it is a possible extension of the present framework.

These relationships distinguish two levels of the construction. First, the exact finite-step joint query optimum contains the disagreement tilt in Eq. (84), whereas Eq. (11) is its zero-tilt population barycenter. Second, Eq. (12) is a sample-wise OU surrogate for this population barycenter: it is exact in the shared-covariance Gaussian case and otherwise is justified by the endpoint, invariance, and contraction properties established in Appendix A.1.

# B Detailed Experimental Setup and Reproducibility

This section specifies the protocol used for the multi-reward, single-reward, few-step, and ablation experiments. We report optimizer updates rather than dataloader epochs as the unit of training because the compared methods may collect different numbers of images before one parameter update. We refer to one optimizer update as one *round*. Evaluation does not increment the round counter or modify the model.

## B.1 Common Experimental Protocol

### B.1.1 Backbone, parameterization, and numerical precision

All experiments use Stable Diffusion 3.5 Medium (SD3.5-M) as the common backbone. We optimize only LoRA parameters attached to the diffusion transformer, with LoRA rank 32 and scaling parameter 64. The text encoders, VAE, reward models, and teacher models are frozen. Master trainable weights and optimizer states are kept in FP32, while model execution uses FP16 mixed precision. Training and evaluation images have resolution $512 \times 512$. Unless explicitly stated otherwise, experiments run on a NVIDIA A100-SXM4 GPU with 80 GB of memory.

The tested software stack is Python 3.11, CUDA 12.1, PyTorch 2.4.1, Torchvision 0.19.1, Diffusers 0.36.0, Transformers 4.57.1, Accelerate 1.12.0, PEFT 0.17.1, and W&B 0.21.1. Exact versions of remaining dependencies are provided in `requirements-tested.txt`. The released asset downloader pins the SD3.5-M checkpoint, the DiffusionOPD teachers, and reward models to fixed revisions, and the release preflight validates the corresponding checkpoint hashes.

### B.1.2 Reward calibration

For each prompt $i$, we cache two task-specific reward anchors before optimization. The lower anchor $R_L$ is obtained from the initial policy, and the upper anchor $R_H$ is obtained from the corresponding frozen task teacher. For the current generated endpoint, let $\widetilde{R}_{0,i}$ denote its task-normalized reward. In implementation, we compute

$$p_i = \mathrm{clip}\left(\frac{\widetilde{R}_{0,i} - R_L}{R_H - R_L + \varepsilon_R}, 0, 1\right), \qquad \varepsilon_R = 10^{-6}. \tag{85}$$

Thus, the calibration measures sample-wise progress relative to the reward range between the initial policy and the corresponding teacher rather than using manually selected absolute reward thresholds. The anchors are fixed during optimization and are not selected using evaluation-set performance.

When

$$R_H \le R_L + \varepsilon_R, \tag{86}$$

the reward band is non-positive or numerically unreliable; in this case we set $p_i = 0$, retaining teacher-guided supervision.

The same $p_i$ is used for both the adaptive query and adaptive velocity target for every task. No separate query coefficient is introduced.

### B.1.3 Optimizer and regularization

We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay $10^{-4}$, Adam numerical constant $\epsilon_{\mathrm{Adam}} = 10^{-8}$, and global gradient-norm clipping at 1.0. Where used by the experimental configuration, the reference-model regularizer has coefficient

$$\mathcal{L}_{\mathrm{ref}} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{D}\left\|\boldsymbol{v}_\theta\big(\boldsymbol{z}^q_{t_i}, t_i, \boldsymbol{c}_i\big) - \boldsymbol{v}_{\mathrm{ref}}\big(\boldsymbol{z}^q_{t_i}, t_i, \boldsymbol{c}_i\big)\right\|_2^2, \qquad \mathcal{L} = \mathcal{L}_{\mathrm{task}} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{ref}}, \quad \lambda_{\mathrm{KL}} = 10^{-4}. \tag{87}$$

The reference parameters are frozen.

The reported model uses an exponential moving average of the LoRA parameters. The EMA uses the released training configuration with nominal decay 0.9 and a 10-round warm-up. The data-collection policy is updated separately as specified by Algorithm 1.

### B.1.4 TRAINING TRAJECTORY AND TIMESTEP WEIGHTING

The full-step experiments collect a deterministic ODE trajectory with 10 denoising steps. The trainable objective uses the nine non-terminal transitions selected by `timestep_range=0.99`. Let

$$\mathcal{J} = \{0, \dots, 8\}. \tag{88}$$

We do not apply additional SNR-dependent or hand-designed timestep weighting. The loss is averaged uniformly over the selected transitions,

$$\mathcal{L} = \sum_{j \in \mathcal{J}} w(t_j) \mathcal{L}_{t_j}, \qquad w(t_j) = \frac{1}{|\mathcal{J}|} = \frac{1}{9}. \tag{89}$$

For the four-step experiment,

$$\mathcal{J} = \{0, 1, 2, 3\}, \qquad w(t_j) = \frac{1}{4}. \tag{90}$$

At every selected transition, the adaptive query is constructed directly from the same reward-preference weight $p_i$:

$$\boldsymbol{z}_t^q = \boldsymbol{\kappa}_t + (1 - p_i)\left(\boldsymbol{z}_t^{\text{old}} - \boldsymbol{\kappa}_t\right) + \sqrt{2p_i - p_i^2}\,\sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\kappa}_t = \alpha_t \boldsymbol{x}_0. \tag{91}$$

No task uses a separate query interpolation coefficient.

The adaptive velocity target is then computed using Eq. (18) at the same query:

$$\boldsymbol{v}_i^\star = p_i \boldsymbol{v}_{\text{NFT}+}^\star + (1 - p_i)\left[\boldsymbol{v}_T - p_i c_t \left(\boldsymbol{v}^{\text{old}} - \boldsymbol{v}_{\text{NFT}-}^\star\right)\right]. \tag{92}$$

The core regression loss is

$$\mathcal{L}_{i,t}^{\text{URF}} = \left\| \boldsymbol{v}_\theta(\boldsymbol{z}_t^q, t, \boldsymbol{c}_i) - \boldsymbol{v}_i^\star \right\|_2^2. \tag{93}$$

When enabled in the released experimental configuration, the same reference-model regularizer is added as

$$\mathcal{L}_{i,t} = \mathcal{L}_{i,t}^{\text{URF}} + \lambda_{\text{KL}} \mathcal{L}_{\text{ref}}. \tag{94}$$

### B.1.5 CONTROLLED RANDOMNESS AND PAIRED COMPARISONS

The primary comparisons use seed 42 and a paired randomness protocol. Competing methods use the same ordered training prompts and the same evaluation prompts whenever their sampling procedure permits direct pairing. Prompt order and task quotas are fixed by the released manifests.

Random generators are keyed by prompt, occurrence index, logical round, and global seed for the URF runs, allowing the current policy, initial-policy anchor, and teacher anchor to use matched prompt-conditioned randomness. Evaluation uses the same prompt files, image seed, sampler, CFG, resolution, and denoising-step count across compared checkpoints.

Exact resume restores the model, optimizer, EMA, old-policy state, sampler position, and random-number-generator states saved by the released implementation. Small numerical differences may still arise across CUDA architectures even with identical prompts, seeds, and checkpoints.

### B.1.6 EVALUATION PROTOCOL

Full-step methods are evaluated every 10 optimizer rounds, including round 0. Evaluation uses 40 deterministic denoising steps, CFG 4.5, $512 \times 512$ resolution, one generated image per prompt, and seed 42. The frozen evaluation sets contain 2,212 GenEval prompts, 1,018 OCR prompts, and 1,000 PickScore/DrawBench prompts.

Evaluation prompts are not used to update model parameters or reward anchors, and checkpoints are not selected using evaluation-set performance.

Table 2: Common optimization and evaluation settings. A round denotes one optimizer update.

| Setting | Resolution | Train steps | $w(t)$ | Eval steps | Eval period | Seed | LoRA $r/\alpha$ | $\lambda_{\mathrm{KL}}$ | $\varepsilon_R$ | GPU |
|---|---|---|---|---|---|---|---|---|---|---|
| Full-step | 512 | 10 (9 optimized) | $1/9$ | 40 | 10 rounds | 42 | 32/64 | $10^{-4}$ | $10^{-6}$ | 1×A100 80GB |
| Few-step | 512 | 4 (4 optimized) | $1/4$ | 4 | 10 rounds[†] | 42 | 32/64 | $10^{-4}$ | $10^{-6}$ | A100 80GB |

[†]The few-step DiffusionNFT control is evaluated every five rounds because one NFT outer collection contains a different number of generated samples.

Table 3: Multi-reward training settings. "Prompts/round" lists the per-task quota and total number of generated samples.

| Method | Prompts/round | Group | LR | $\beta_{G/O/P}$ | $\mathrm{CFG}_{G/O/P}$ | Eval CFG | GPU |
|---|---|---|---|---|---|---|---|
| URF | 24/24/24 (72) | 1 | $10^{-4}$ | $1/0.1/1$ | $1/4.5/4.5$ | 4.5 | A100 80GB |
| DiffusionOPD | 24/24/24 (72) | 1 | $3 \times 10^{-4}$ | n/a | $4.5/4.5/4.5$ | 4.5 | A100 80GB |

## B.2 Multi-Reward Experiments

### B.2.1 Task routing and update construction

The multi-reward experiment contains three task sources: GenEval, OCR, and PickScore. Each optimization round contains 24 distinct prompts from each source, for a total of 72 prompts and 72 generated samples. Group size is one.

A sample is evaluated only by the reward function and frozen teacher corresponding to its task. For every sample, we independently compute its task-specific anchors $(R_L, R_H)$, reward-preference weight $p_i$, adaptive query $\boldsymbol{z}_t^q$, and adaptive velocity target $\boldsymbol{v}_i^\star$. The same $p_i$ controls both the query and velocity target.

We do not average reward-preference weights, queries, or velocity targets across tasks. Instead, each routed sample contributes its own URF regression loss. Because each task contributes the same number of samples, averaging over all 72 samples is equivalent, up to an overall constant, to combining the three task means with equal task weight:

$$\mathcal{L}_{\mathrm{multi}} = \frac{1}{3}\left(\overline{\mathcal{L}}_{\mathrm{GenEval}} + \overline{\mathcal{L}}_{\mathrm{OCR}} + \overline{\mathcal{L}}_{\mathrm{PickScore}}\right). \tag{95}$$

The physical microbatch size is three, and gradients are accumulated until all routed samples in the round have contributed to the update.

The multi-reward learning rate is $10^{-4}$. GenEval uses rollout/teacher CFG $1.0/1.0$ and $\beta = 1.0$. OCR uses rollout/teacher CFG $4.5/4.5$ and $\beta = 0.1$. PickScore uses rollout/teacher CFG $4.5/4.5$ and $\beta = 1.0$. Evaluation uses CFG 4.5 for all three tasks.

## B.3 Single-Reward Experiments

Single-reward experiments use the same SD3.5-M initialization, LoRA parameterization, trajectory discretization, prompt/noise protocol, evaluation seed, and test manifests as the multi-reward experiments. Only one reward, its corresponding teacher, and its prompt source are active.

The GenEval experiment uses 72 distinct singleton prompts per round, learning rate $3 \times 10^{-4}$, $\beta = 1.0$, and CFG 1.0 for both the current policy and teacher.

The OCR experiment uses 72 distinct singleton prompts per round, learning rate $3.75 \times 10^{-5}$, $\beta = 0.1$, and CFG 4.5 for both the current policy and teacher.

The PickScore experiment uses 72 distinct singleton prompts per round, learning rate $3 \times 10^{-4}$, $\beta = 1.0$, current-policy CFG 1.0, and teacher CFG 4.5.

All three tasks are evaluated with CFG 4.5.

The retained trajectories contain 510 GenEval rounds, 490 OCR rounds, and 210 PickScore rounds. Evaluations occur at round 0 and every 10 rounds thereafter. The final retained point is reported rather than selecting an earlier checkpoint according to evaluation performance. Baselines use their

Table 4: Single-reward URF settings. All rows use group size 1, $\varepsilon_R = 10^{-6}$, seed 42, and one A100-SXM4-80GB GPU.

| Reward | Prompts/round | LR | $\beta$ | Train CFG | Teacher CFG | Eval CFG | Total rounds |
|---|---|---|---|---|---|---|---|
| GenEval | 72 | $3 \times 10^{-4}$ | 1.0 | 1.0 | 1.0 | 4.5 | 510 |
| OCR | 72 | $3.75 \times 10^{-5}$ | 0.1 | 4.5 | 4.5 | 4.5 | 490 |
| PickScore | 72 | $3 \times 10^{-4}$ | 1.0 | 1.0 | 4.5 | 4.5 | 210 |

respective objectives and learning rates while matching the evaluation prompt sets, resolution, CFG, seed, and 40-step evaluation sampler.

### B.4 Four-Step Distillation Experiments

#### B.4.1 Student, teacher, and segment target

The few-step experiments initialize the student from the public four-step SenseFlow SD3.5-M checkpoint and use the OCR-specialized reward-optimized SD3.5-M checkpoint at optimizer round 495 as the frozen teacher.

The student follows a four-step Euler schedule with boundaries

$$[\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4] = [1.0, 0.9, 0.75, 0.5, 0.0] \tag{96}$$

and runs with CFG 1.0. The teacher runs with CFG 4.5.

For each student segment $[\sigma_k, \sigma_{k+1}]$, we integrate the frozen teacher over 10, 15, 25, and 50 native microsteps, respectively, and construct the segment velocity

$$\bar{\boldsymbol{v}}_{T,k} = \frac{\boldsymbol{x}_T(\sigma_{k+1}) - \boldsymbol{x}^q(\sigma_k)}{\sigma_{k+1} - \sigma_k}. \tag{97}$$

This construction provides a teacher target spanning the longer temporal interval represented by each four-step student transition. The four segments are weighted uniformly:

$$w(t_k) = \frac{1}{4}. \tag{98}$$

The URF few-step experiment uses 72 images from 72 distinct prompts per round, group size 1, learning rate $3.75 \times 10^{-5}$, $\beta = 0.1$, $\varepsilon_R = 10^{-6}$, and $\lambda_{\mathrm{KL}} = 10^{-4}$.

The OPD adaptation uses the same four coarse student segments and integrated teacher target with the OPD transition-matching objective, learning rate $3 \times 10^{-4}$, and the released OPD EMA cadence.

The DiffusionNFT control uses 48 unique prompts with group size 24, producing 1,152 images per outer collection, learning rate $7.5 \times 10^{-5}$, $\beta = 0.1$, and $\lambda_{\mathrm{KL}} = 10^{-4}$. It uses the native four-segment discrete-time sampler. Because an NFT outer collection contains substantially more images, this control is evaluated every five optimizer rounds, whereas URF and OPD are evaluated every 10 rounds.

Every evaluation uses the same 1,018 OCR prompts, four-step student sampler, CFG 1.0, and seed 42.

The retained trajectories contain 260 rounds for OPD, 40 rounds for DiffusionNFT, and 30 rounds for the URF few-step run. If a run terminates after a numerical failure, the curve ends at the last finite scheduled evaluation; no missing values are extrapolated or imputed.

### B.5 Ablation Protocol

The ablations isolate the adaptive query, velocity target, and teacher-guided negative correction. Unless the ablated component is explicitly changed, all settings follow the single-reward GenEval protocol: 72 distinct singleton prompts per round, learning rate $3 \times 10^{-4}$, $\beta = 1.0$, $\varepsilon_R = 10^{-6}$, nine uniformly weighted train transitions, seed 42, and evaluation every 10 rounds.

All compared variants use the same prompt sequence and evaluation protocol.

Table 5: Four-step OCR settings. All methods use the same SenseFlow student initialization, four student segments, OCR evaluation set, and seed.

| Method | Unique prompts | Group | Images/collection | LR | $\beta$ | Eval period | Retained rounds | Student/teacher CFG | GPU |
|---|---|---|---|---|---|---|---|---|---|
| URF | 72 | 1 | 72 | $3.75 \times 10^{-5}$ | 0.1 | 10 | 30 | 1.0/4.5 | A100 80GB |
| DiffusionOPD | 72 | 1 | 72 | $3 \times 10^{-4}$ | n/a | 10 | 260 | 1.0/4.5 | A100 80GB |
| DiffusionNFT | 48 | 24 | 1,152 | $7.5 \times 10^{-5}$ | 0.1 | 5 | 40 | 1.0/n/a | A100 80GB |

**Query ablation.** We hold the adaptive velocity target fixed and compare: (i) the full adaptive query in Eq. (12); (ii) the OPD query

$$\boldsymbol{z}_t^q = \boldsymbol{z}_t^{\mathrm{old}}; \tag{99}$$

and (iii) the NFT-side forward query

$$\boldsymbol{z}_t^q = \boldsymbol{\kappa}_t + \sigma_t \boldsymbol{\epsilon}. \tag{100}$$

Only the query location is changed; the reward-preference weight used by the velocity target remains the original $p_i$.

**Velocity-target ablation.** We hold the adaptive query fixed and compare: (i) the full adaptive target in Eq. (17); (ii) the teacher/OPD target $\boldsymbol{v}^\star = \boldsymbol{v}_T$; and (iii) the positive NFT target $\boldsymbol{v}^\star = \boldsymbol{v}^\star_{\mathrm{NFT}+}$. All other optimization settings are unchanged.

**TG-negative ablation.** We compare the complete method against (i) a variant without TG-negative correction,

$$\boldsymbol{v}^\star = p_i \boldsymbol{v}^\star_{\mathrm{NFT}+} + (1 - p_i)\boldsymbol{v}_T, \tag{101}$$

and (ii) a direct scalar interpolation of separately evaluated OPD and NFT training losses. The former preserves the common adaptive query but removes the teacher-guided negative correction.

All ablation curves use measured scheduled evaluations from their own run logs. Schematic curves used during figure-layout design are excluded from quantitative results.

### B.6 Checkpointing, Failure Handling, and Exact Resume

Exact-resume checkpoints are written at optimizer boundaries. The stored state contains the LoRA parameters, optimizer and learning-rate scheduler, EMA, old-policy parameters, prompt reward anchors, logical round, sampler position, and random-number-generator states required by the released training code.

A failed update is not counted as a completed round. After preemption, the training process resumes from the next uncommitted optimizer round using the restored sampler and random-number-generator state. This prevents an apparent resume from silently changing the subsequent prompt or noise sequence.

### B.7 Reproduction Checklist

For reproduction, we use the following procedure:

1. Install the pinned dependencies in `requirements-tested.txt`.
2. Run `scripts/download_assets.py` to obtain the fixed SD3.5-M, teacher, and reward-model assets.
3. Run `scripts/preflight.sh` to validate the required checkpoints and experiment configuration.
4. Use seed 42 consistently for the released training and evaluation configurations.
5. Use `configs/train_72.yaml` for the multi-reward experiment, which samples 24 GenEval, 24 OCR, and 24 PickScore prompts per round.
6. Count training progress by optimizer rounds rather than generated images or dataloader epochs.
7. Resume training through the released exact-resume launcher so that the optimizer, EMA, old policy, prompt anchors, sampler state, and random states are restored together.

8. Export the EMA LoRA using `scripts/export_ema_lora.py` and evaluate it against the frozen prompt manifests provided with the release.

The executable configuration is provided in `configs/train_72.yaml`; the source-level method mapping is documented in `docs/METHOD.md`; model and reward provenance is provided in `docs/PROVENANCE.md`; and the released source snapshot is accompanied by `SOURCE_MANIFEST.sha256`.