BEYOND SCORES WITH REWARD-MODEL EMBEDDINGS FOR DIVERSE FLOW RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) improves text-to-image flow models, but scalar rewards assigned independently to individual images overlook visual redundancy within a prompt group. We show that reward-model embeddings provide diversity feedback within each prompt group. We propose using regularized D-optimal design to compute diversity weights from reward-model embeddings and reshape scalar rewards before advantage computation. Our method requires no additional image encoder or encoder forward pass and preserves the underlying sampling and policy-update rules. We evaluate our method on Stable Diffusion 3.5 Medium (SD3.5-M) with Advantage Weighted Matching (AWM) and DiffusionNFT, two RL algorithms for flow models. On DrawBench, our method improves overall quality and within-prompt diversity, including gains on three held-out reward models not used during training. On Pick-a-Pic, our method attains each baseline's highest evaluation reward $1.38\times$ faster for AWM and $4.32\times$ faster for DiffusionNFT, measured in GPU hours.

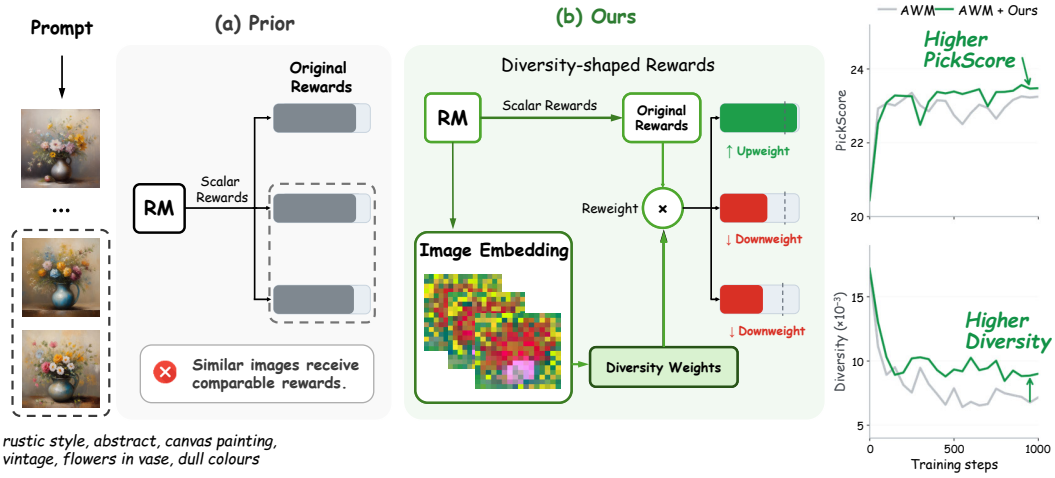


Figure 1: **Reward-model embeddings reveal visual redundancy. (a)** Two of the three images generated for the prompt (dashed box) have a similar vase and composition, yet the reward model (RM) assigns all three comparable scalar rewards. The resulting advantages give the three images comparable weights in the policy update, although the second similar image adds little visual information beyond the first. **(b)** Our method reuses the reward-model embeddings, compares them within the prompt group to obtain diversity weights, and multiplies the original rewards by these weights. The two similar images are downweighted and the distinct image is upweighted, so the distinct image contributes more to the policy update without an additional encoder. Training curves (right) show that our method achieves higher PickScore and within-prompt diversity than the AWM baseline.

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

Flow matching models are widely used for text-to-image and text-to-video generation, as well as image and video editing (Lipman et al., 2023; Esser et al., 2024; Team Wan et al., 2025; Black Forest Labs et al., 2025). Yet pretraining alone does not directly optimize criteria such as aesthetics, prompt faithfulness, or safety. Recent works therefore fine-tune diffusion and flow models with reinforcement learning from human feedback (RLHF), using reward models trained on human preferences (Fan et al., 2023; Liu et al., 2025; Zheng et al., 2026). In a typical text-to-image RL workflow, the model generates a group of images for each prompt, and a reward model assigns each image a scalar score. The RL algorithm then computes advantages from these scores and uses them to optimize the policy.

As shown in Figure 1a, rewards assigned independently to each image assess its quality but do not account for visual redundancy within the prompt group. The reward model can therefore continue to assign high scores to visually similar images. Reinforcing these images over successive updates can concentrate the model on a few recurring visual patterns, leading to diversity collapse (Jena et al., 2025; Tan et al., 2026; Dombrowski et al., 2025). Recent methods use additional visual encoders to derive diversity signals beyond individual quality scores (He et al., 2026; Liu et al., 2026a). Other approaches modify sampling procedures or use specialized objectives that depend on trajectory likelihoods or gradients through denoising transitions (Liu et al., 2026b; Li et al., 2026; Kang et al., 2026). These additions can increase training complexity and computational cost, while dependence on specific sampling or update rules can limit their applicability across RL algorithms.

Our key finding is that *your reward model is secretly a diversity scorer*. Before producing a scalar score, the reward model computes an embedding of each image, and these embeddings can capture visual differences between images with similar rewards. For example, two of the three images shown in Figure 1a have a similar vase, flower arrangement, and background, yet all three receive comparable rewards. Advantages computed from these rewards give the three images comparable weights in the policy update. However, the second similar image adds little visual information beyond the first. As shown in Figure 1b, our method compares the reward-model embeddings within the prompt group and assigns lower weights to the two similar images, so the distinct image contributes more to the policy update.

We propose a reward reshaping method that combines scalar rewards with diversity weights computed from reward-model embeddings. Each image's weight depends on the other images generated for the same prompt. We compute the weights jointly using regularized D-optimal design (Kiefer & Wolfowitz, 1960; Boyd & Vandenberghe, 2004), encouraging coverage of complementary visual features. We then reshape each image's reward with its diversity weight before computing advantages. Because our method changes only the rewards supplied to the RL algorithm, it can be applied to different RL methods for flow models without changing their sampling or policy-update rules. The training curves in Figure 1b show that our method achieves higher PickScore (top right) and maintains higher within-prompt diversity (bottom right) than the AWM baseline in DrawBench evaluations throughout training.

We evaluate our method on SD3.5-M using two RL methods for flow models, AWM and DiffusionNFT. On DrawBench, we achieve overall quality gains of 2.62% for AWM and 3.38% for DiffusionNFT, averaged across six quality metrics. Our method also outperforms GARDO (He et al., 2026) on all six quality metrics with both algorithms. We also improve all three held-out metrics with both algorithms, showing that the gains extend to evaluators not used during training. Our method increases within-prompt diversity by 45.61% and 8.45%, respectively. Our method also improves training efficiency, attaining each baseline's highest Pick-a-Pic reward $1.38\times$ faster for AWM and $4.32\times$ faster for DiffusionNFT.

Our contributions are as follows.

- We find that a reward model provides more than a scalar quality score. Its image embeddings capture visual differences among images from the same prompt, so a single reward evaluation yields both a quality score for each image and the embeddings needed to compute diversity within the prompt group.
- We propose a reward reshaping method that jointly computes diversity weights from reward-model embeddings using regularized D-optimal design. It incorporates these

weights into scalar rewards before advantage computation, without an additional encoder or changes to the sampler or policy-update rule.

- We demonstrate consistent improvements in overall quality, within-prompt diversity, and training efficiency with both AWM and DiffusionNFT on SD3.5-M, outperforming GARDO (He et al., 2026), a recent diversity method for flow RL, on all six quality metrics.

2 PRELIMINARIES

Following Xu et al. (2026), we distinguish two approaches to RL for flow models. Reverse-trajectory RL uses rewards and the likelihoods of sampled denoising transitions to update the model. Forward-matching RL uses rewards to guide flow-matching training on newly noised versions of the generated samples, with training timesteps chosen independently of the sampling trajectory (Ping et al., 2026).

Reverse-trajectory RL. We illustrate reverse-trajectory RL with Flow-GRPO (Liu et al., 2025), which applies a clipped policy loss to stochastic denoising transitions, assigning each step the advantage of its final image. For a fixed prompt $\boldsymbol{c}$, let $\{(\boldsymbol{x}_T^i, \ldots, \boldsymbol{x}_0^i)\}_{i=1}^K$ be K sampled T-step trajectories, where the superscript i indexes images. Flow-GRPO minimizes the following loss with respect to the model parameters θ.

$$\begin{aligned} \mathcal{L}_{\mathrm{GRPO}}(\theta; \boldsymbol{c}) &= \frac{1}{KT} \sum_{i=1}^{K} \sum_{j=1}^{T} \left\{ -\min(\rho_{ij} A_i, \mathrm{clip}(\rho_{ij}, 1-\varepsilon, 1+\varepsilon) A_i) + \beta D_{ij} \right\}, \\ \rho_{ij} &= \frac{p_\theta(\boldsymbol{x}_{j-1}^i \mid \boldsymbol{x}_j^i, \boldsymbol{c})}{p_{\mathrm{old}}(\boldsymbol{x}_{j-1}^i \mid \boldsymbol{x}_j^i, \boldsymbol{c})}. \end{aligned} \tag{1}$$

Here, j indexes denoising steps remaining, so $\boldsymbol{x}_T^i$ is the initial noisy state and $\boldsymbol{x}_0^i$ is the final clean image. For the recorded transition $\boldsymbol{x}_j^i \to \boldsymbol{x}_{j-1}^i$, ρ_{ij} is the likelihood ratio between the current policy p_θ and the fixed rollout policy p_{old} that generated the trajectories. The parameter ε sets the clipping range $[1-\varepsilon, 1+\varepsilon]$ for this ratio. D_{ij} is the Kullback–Leibler (KL) divergence from the current transition distribution to the fixed reference distribution at $(\boldsymbol{x}_j^i, \boldsymbol{c})$. The coefficient $\beta \geq 0$ controls the penalty's strength.

The final image $\boldsymbol{x}_0^i$ receives a scalar reward r_i, which enters the loss through its advantage $A_i = \frac{r_i - \bar{r}_{\boldsymbol{c}}}{\sigma}$, where $\bar{r}_{\boldsymbol{c}} = K^{-1} \sum_{k=1}^{K} r_k$ is the prompt-group mean reward and σ is a normalizing scale.

Forward-matching RL. We illustrate forward-matching RL with DiffusionNFT (Zheng et al., 2026), which uses image-level advantages to weight positive and negative velocity branches. For each image $\boldsymbol{x}_0^i$ in the same prompt group, we draw a fresh noise level $t_i \in [0, 1]$ and Gaussian noise $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We form the noisy input $\boldsymbol{x}_t^i = (1-t_i)\boldsymbol{x}_0^i + t_i \boldsymbol{\epsilon}_i$ with target velocity $\boldsymbol{\epsilon}_i - \boldsymbol{x}_0^i$. Let $\boldsymbol{v}_\theta$ and $\boldsymbol{v}_{\mathrm{old}}$ denote the current and fixed rollout velocity predictions at $(\boldsymbol{x}_t^i, \boldsymbol{c}, t_i)$. With a fixed scale $\lambda > 0$, define the positive and negative branches

$$\boldsymbol{v}_\theta^+ = \boldsymbol{v}_{\mathrm{old}} + \lambda(\boldsymbol{v}_\theta - \boldsymbol{v}_{\mathrm{old}}), \qquad \boldsymbol{v}_\theta^- = \boldsymbol{v}_{\mathrm{old}} - \lambda(\boldsymbol{v}_\theta - \boldsymbol{v}_{\mathrm{old}}).$$

The advantage A_i defined above determines the positive-branch weight $\omega_i = \frac{1}{2}[1 + \mathrm{clip}(A_i / A_{\max}, -1, 1)]$, with clipping threshold $A_{\max} > 0$. For these fixed training samples, the velocity-matching component of DiffusionNFT's loss is

$$\begin{aligned} \mathcal{L}_{\mathrm{NFT}}(\theta; \boldsymbol{c}) = \frac{1}{K} \sum_{i=1}^{K} \Big[& \omega_i \| \boldsymbol{v}_\theta^+(\boldsymbol{x}_t^i, \boldsymbol{c}, t_i) - (\boldsymbol{\epsilon}_i - \boldsymbol{x}_0^i) \|_2^2 \\ & + (1-\omega_i) \| \boldsymbol{v}_\theta^-(\boldsymbol{x}_t^i, \boldsymbol{c}, t_i) - (\boldsymbol{\epsilon}_i - \boldsymbol{x}_0^i) \|_2^2 \Big]. \end{aligned} \tag{2}$$

AWM (Xue et al., 2026) weights the velocity-matching loss by each image's signed advantage in its on-policy update. AdvantageFlow (Kveton et al., 2026) stabilizes an advantage-weighted prediction loss with rollout-policy and reference-policy regularization.

Our diversity method applies to all of the above objectives; we describe it next in Section 3.

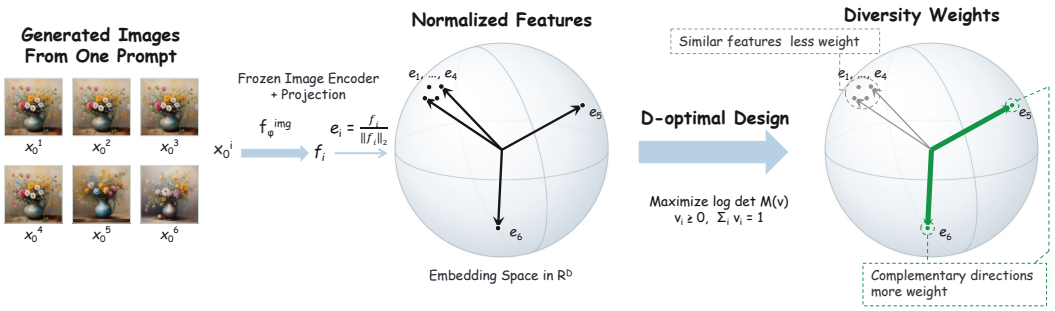


Figure 2: **D-optimal design assigns diversity weights.** We encode the images generated from one prompt with the reward model's frozen image encoder and normalize the features (left). In the embedding space, the first four images point in similar directions, while the last two point in different directions (middle). We then use D-optimal design to assign the weights of all six images jointly (right). The four similar images share weight and each receives less (gray), while the two distinct images receive more (green). The diversity weights thus come from the reward model's existing embeddings without an additional encoder.

3 Flow Reinforcement Learning via Reward Reshaping

Our method uses the reward model's image embeddings to measure visual redundancy within each prompt group and lowers the rewards of redundant images relative to distinct ones. Figure 2 details the allocation step previewed in Figure 1b. We compute diversity weights by regularized D-optimal design and use them to reshape the scalar rewards before advantage computation.

The reward model is secretly a diversity scorer. Before assigning a scalar score, the reward model represents each image as a feature vector, or embedding. The embeddings can capture visual differences between images that receive similar scalar rewards. For each image $\boldsymbol{x}_0^i$ in the prompt group from Section 2, we extract its embedding from the reward model's final projection. These projected embeddings are used for image–text scoring in CLIPScore (Hessel et al., 2021), PickScore (Kirstain et al., 2023), and HPSv2.1 (Wu et al., 2023). We normalize each embedding to unit length so that its magnitude does not affect the diversity weights. We stack the normalized embeddings as rows of E.

$$f_i = f^{\mathrm{img}}(\boldsymbol{x}_0^i), \qquad e_i = \frac{f_i}{\|f_i\|_2}, \qquad E = [e_1, \ldots, e_K]^\top \in \mathbb{R}^{K \times D}. \tag{3}$$

Here, f^{img} is a fixed image encoder, and D is the embedding dimension. We reuse these embeddings from reward evaluation without an additional forward pass.

Diversity weights and reward reshaping. We assign each image a non-negative diversity weight w_i to control its relative emphasis within the prompt group. The aim is to emphasize complementary features while limiting the combined weight on images with similar embeddings, as shown in Figure 2.

We compute the weights jointly using a regularized version of D-optimal design (Boyd & Vandenberghe, 2004; Mukherjee et al., 2024). The design matrix $M(w) \in \mathbb{R}^{D \times D}$ is a weighted sum of embedding outer products plus a ridge term. We constrain the weight vector $w \in \mathbb{R}^K$ to the probability simplex Δ_K and denote an optimal solution by $\nu \in \Delta_K$.

$$\begin{aligned} \nu &\in \arg\max_{w \in \Delta_K} \log\det M(w), \qquad M(w) = \gamma I_D + \sum_{i=1}^{K} w_i e_i e_i^\top, \\ \Delta_K &= \left\{ w \in \mathbb{R}^K : w \geq \mathbf{0}_K,\ \sum_{i=1}^{K} w_i = 1 \right\}, \qquad \gamma > 0. \end{aligned} \tag{4}$$

Each outer product $e_i e_i^\top$ is a rank-one matrix, and w_i controls its contribution to $M(w)$. The ridge term γI_D makes $M(w)$ positive definite. Without it, the sum has rank at most K and is singular whenever $K < D$, making the log determinant undefined. The regularization also reduces the sensitivity of the objective to small differences between nearly redundant embeddings. Geometrically,

the ellipsoid $\{x : x^\top M(w)^{-1}x \le 1\}$ has volume proportional to $\sqrt{\det M(w)}$, so maximizing the log determinant maximizes its volume (Boyd & Vandenberghe, 2004; Lattimore & Szepesvári, 2020). Each weighted outer product $w_i e_i e_i^\top$ enlarges the ellipsoid along e_i, and additional weight along an already large direction increases the log determinant less. In the middle panel of Figure 2, $e_1, \dots, e_4$ point in nearly the same direction, while e_5 and e_6 enlarge the ellipsoid along directions that $e_1, \dots, e_4$ do not cover. As shown in the right panel, the optimal allocation therefore splits weight among the four similar embeddings (gray), so each receives less weight than e_5 or e_6 (green).

Because a larger ν_i reflects a greater contribution to the group's diversity, we convert the allocation into a reward multiplier. We rescale ν so that the multipliers q_i have a mean of one.

$$q_i = K\nu_i, \qquad \frac{1}{K}\sum_{i=1}^{K} q_i = 1. \tag{5}$$

We then multiply each image's scalar reward r_i by q_i.

$$\widetilde{r}_i = r_i q_i. \tag{6}$$

Intuitively, the D-optimal weights give more reward weight to images that add distinct feature directions to the prompt group ($q_i > 1$) and less to images with redundant features ($q_i < 1$). When $q_i = 1$, the reward is unchanged. We apply reward reshaping to every prompt group and recompute advantages from the reshaped rewards $\widetilde{r}_i$.

$$\widetilde{A}_i = \frac{\widetilde{r}_i - \overline{r}_{\boldsymbol{c}}}{\widetilde{\sigma}}, \qquad \overline{r}_{\boldsymbol{c}} = \frac{1}{K}\sum_{k=1}^{K} \widetilde{r}_k. \tag{7}$$

Here, $\widetilde{\sigma}$ is the standard deviation of the reshaped rewards over the full rollout batch across prompts and devices. Section A.3 specifies how we control the reshaping strength by interpolation and stabilize the denominator. For reverse-trajectory RL, we replace A_i with $\widetilde{A}_i$ in Equation (1). For forward-matching RL, we use $\widetilde{A}_i$ to recompute the loss weights while retaining the original sampling and update rules. In the DiffusionNFT loss in Equation (2), the recomputed advantages give the branch weights $\widetilde{\omega}_i = \frac{1}{2}[1 + \mathrm{clip}(\widetilde{A}_i/A_{\max}, -1, 1)]$ and $1 - \widetilde{\omega}_i$.

Solving the allocation problem. The objective in Equation (4) is concave in w on the simplex Δ_K. We use a pairwise Frank–Wolfe solver, starting from uniform weights $w_i = 1/K$. Each iteration computes the gradient entries $h_i = e_i^\top M(w)^{-1} e_i$ and transfers weight from the active image with the smallest h_i to the image with the largest h_i, using exact line search. We stop when the Frank–Wolfe gap falls below the tolerance specified in Section A.2. Each iteration inverts the $D \times D$ design matrix, costing $O(D^3)$ per step.

Reducing the cost with SVD. Inverting the $D \times D$ design matrix at every solver iteration is costly when D is large, yet each prompt group contains only K embeddings, which span at most K dimensions. We therefore solve the allocation problem in the span of the embeddings, whose dimension is $n = \mathrm{rank}(E) \le \min(K, D)$. A compact singular value decomposition (SVD) gives coordinates in this span.

$$E = U\Sigma V^\top, \qquad \widehat{E} = EV = U\Sigma. \tag{8}$$

The columns of $V \in \mathbb{R}^{D\times n}$ form an orthonormal basis for this span, and the ith row of $\widehat{E}$ is the reduced embedding $\widehat{e}_i^\top$. Because every e_i lies in the column space of V, the reduced coordinates preserve pairwise inner products ($\widehat{e}_i^\top \widehat{e}_j = e_i^\top e_j$) and each outer product $e_i e_i^\top$ acts only within this n-dimensional span. We form the reduced design matrix $\widehat{M}(w) = \gamma I_n + \sum_{i=1}^{K} w_i \widehat{e}_i \widehat{e}_i^\top$. Directions orthogonal to the embedding span contribute only the ridge term, so in exact arithmetic

$$\log\det M(w) = (D - n)\log\gamma + \log\det \widehat{M}(w). \tag{9}$$

The additive term is independent of w, so the full and reduced problems have the same optimal allocations ν and reward multipliers q_i. Section A.1 gives the proof and the numerical SVD details.

We run the same pairwise Frank–Wolfe solver on the reduced embeddings. For $K \le D$, the SVD costs $O(DK^2)$ once per group, and each subsequent solver iteration costs $O(n^3 + Kn^2 + K\log K) = O(K^3)$ instead of $O(D^3)$ in the original space. With $K = 16$ images per prompt and embedding dimensions $D \in \{768, 1024\}$, the SVD reduces the design matrix from $D \times D$ to at most 16×16.

Table 1: **State-of-the-art comparison on SD3.5-M.** We evaluate AWM and DiffusionNFT on DrawBench. Our method improves overall quality and within-prompt diversity over both baselines, and outperforms GARDO on all six quality metrics. **Bold** marks the best result within each algorithm. Dashes denote unavailable or inapplicable entries. † marks our method.

Method	Training-reward metrics↑			Held-out metrics↑			Overall quality gain↑ (%)	Diversity↑ ($\times 10^{-3}$)
	PickScore	CLIPScore	HPSv2.1	Aesthetic	ImgRwd	UniRwd		
SD3.5-M (w/o CFG)	20.51	0.237	0.204	5.13	-0.58	2.02	—	—
+ CFG	22.34	0.285	0.279	5.36	0.85	3.03	—	—
AWM								
Baseline	23.41	**0.298**	0.317	6.01	1.42	3.34	0.00%	6.30
+ Ours†	**23.69**	0.297	**0.332**	**6.15**	**1.50**	**3.40**	**+2.62%**	**9.18**
+ GARDO	22.69	0.286	0.316	5.83	1.37	3.18	-3.09%	5.91
DiffusionNFT								
Baseline	23.34	0.284	0.327	6.19	1.42	3.36	0.00%	9.61
+ Ours†	**23.70**	**0.292**	**0.344**	**6.22**	**1.52**	**3.47**	**+3.38%**	10.42
+ GARDO	23.46	0.285	0.333	6.12	1.42	3.44	+0.61%	**10.57**

4 EXPERIMENTS

We evaluate whether reward-model embeddings provide useful diversity feedback for flow RL. With AWM and DiffusionNFT, we examine six quality metrics and within-prompt diversity, time to attain baseline rewards, and the effect of the embedding source.

Lowering classifier-free guidance (CFG) can increase variation, but also affects image quality and prompt alignment (Karras et al., 2024). We disable CFG during both training and evaluation, so our comparisons assess diversity feedback during RL at a fixed guidance setting.

4.1 EXPERIMENTAL SETUP

Reward Models. Following the model-based reward stage of DiffusionNFT (Zheng et al., 2026), we jointly optimize PickScore (Kirstain et al., 2023), CLIPScore (Hessel et al., 2021), and HPSv2.1 (Wu et al., 2023), summing the three scores with PickScore divided by 26 to match the scale of the other two. We report these scores as training-reward metrics. We also report Aesthetic (Schuhmann, 2022), ImageReward (ImgRwd) (Xu et al., 2023), and UnifiedReward (UniRwd) (Wang et al., 2025) as held-out metrics, which we exclude from training to test whether improvements extend beyond the optimized rewards. We average each of the six scores over evaluation images and define the evaluation reward as the mean training reward over these images.

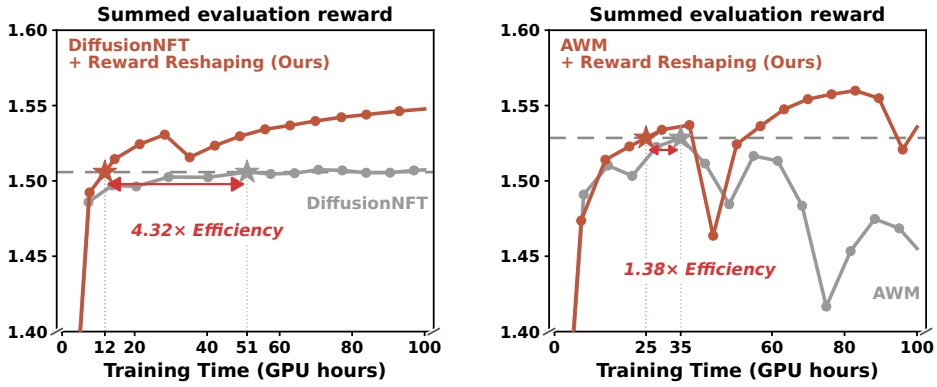
Prompt Datasets. All methods train on the 25,432-prompt Pick-a-Pic training set (Kirstain et al., 2023) and are evaluated on the 200 DrawBench prompts (Saharia et al., 2022).

Training. We use the AWM (Xue et al., 2026) and DiffusionNFT (Zheng et al., 2026) implementations in Flow-Factory (Ping et al., 2026). As in DiffusionNFT, we fine-tune SD3.5-M (Esser et al., 2024) at 512×512 resolution using low-rank adaptation (LoRA) (Hu et al., 2022). During training, we generate 16 images per prompt.

Evaluation. As in DiffusionNFT, we generate four images per DrawBench prompt with a 40-step ordinary differential equation (ODE) sampler. We define *overall quality gain* as $(100/6)\sum_{m=1}^{6}(s_m - s_m^{\text{base}})/s_m^{\text{base}}$, where s_m and s_m^{base} are the mean scores of a method and its baseline on quality metric m. We define *Diversity* as the mean of $d_i = 1 - \max_{j\neq i} z_i^\top z_j$ over evaluation images, where z_i is the unit-normalized PickScore embedding of image i and j ranges over the other images in its prompt group. Higher values indicate less visual redundancy. Section B lists the other settings.

4.2 RESULTS

Baseline comparison. As shown in Table 1, our method improves overall quality and within-prompt diversity for both AWM and DiffusionNFT. We achieve overall quality gains of 2.62% for AWM and



(a) DiffusionNFT. (b) AWM.

Figure 3: **Training efficiency.** Curves show Pick-a-Pic evaluation reward versus GPU hours, using the reward defined in Section 4.1. Gray dashed lines mark each baseline's highest reward within 60 GPU hours, and stars mark when each run reaches this target. Our method attains the target $4.32\times$ faster with DiffusionNFT and $1.38\times$ faster with AWM.

Table 2: **Embedding ablation on SD3.5-M.** We vary the embedding source for diversity weighting and evaluate on DrawBench. All three sources yield positive overall quality gains with both AWM and DiffusionNFT. Gray cells mark training-reward metrics. **Bold**/underline mark the best/second-best within each algorithm. † marks our method.

Method	Embedding source	PickScore	CLIPScore	HPSv2.1	Aesthetic	ImgRwd	UniRwd	Overall quality gain↑ (%)
SD3.5-M (w/o CFG)	—	20.51	0.237	0.204	5.13	-0.58	2.02	—
+ CFG	—	22.34	0.285	0.279	5.36	0.85	3.03	—
Our joint-reward fine-tuning of SD3.5-M								
+ **AWM**	—	23.02	0.292	0.304	5.82	1.25	3.17	0.00%
+ Ours†	PickScore	**23.56**	0.294	0.335	**6.13**	**1.45**	3.31	+6.37%
+ Ours†	CLIPScore	23.45	**0.302**	**0.341**	6.06	1.43	3.26	**+6.42%**
+ Ours†	HPSv2.1	23.51	0.298	0.335	6.08	1.39	**3.31**	+5.68%
+ **DiffusionNFT**	—	23.29	0.284	0.327	**6.19**	1.42	3.36	0.00%
+ Ours†	PickScore	**23.75**	0.292	**0.341**	6.14	1.51	**3.47**	**+2.98%**
+ Ours†	CLIPScore	23.58	**0.294**	0.337	6.19	**1.52**	3.45	+2.90%
+ Ours†	HPSv2.1	23.53	0.291	0.334	6.17	1.50	3.45	+2.19%

3.38% for DiffusionNFT over their respective baselines. We improve all three held-out metrics with both algorithms, showing that the quality gains extend beyond the training rewards. Our method also increases within-prompt diversity by 45.61% for AWM and 8.45% for DiffusionNFT.

Comparison with GARDO. We compare our method with GARDO (He et al., 2026) under both AWM and DiffusionNFT, using the settings in Section B.2. As shown in Table 1, our method outperforms GARDO on all six quality metrics, with mean relative score gains over GARDO of 5.90% for AWM and 2.76% for DiffusionNFT. Compared with GARDO, our method achieves higher diversity with AWM and comparable diversity with DiffusionNFT.

Training efficiency. We evaluate training efficiency on Pick-a-Pic by the GPU hours each run needs to reach a common evaluation reward target. Following the protocol in Section B.3, we set this target to each baseline's highest evaluation reward within 60 GPU hours. As shown by the curves of evaluation reward versus GPU hours in Figure 3, our method reaches the target $4.32\times$ faster for DiffusionNFT and $1.38\times$ faster for AWM.

Embedding ablation. We test whether diversity weighting retains overall quality gains across reward-model embedding sources. We vary the embedding source among PickScore, CLIPScore,



Figure 4: **Selected qualitative examples on DrawBench with AWM.** Each row shows four samples for the stated prompt. For the car prompt, the baseline repeatedly produces boxy, mostly red cars, whereas our method produces cars with clearer body contours and more varied shapes, headlight designs, and colors, including orange sedans, a yellow-and-silver pair, and yellow compact cars. For the storefront prompt, the baseline repeats red outlined lettering, while our method combines clearer window frames and brickwork with varied sign colors, awnings, and facade geometry, including flat and curved frontages.

and HPSv2.1 for diversity weighting, keeping the joint training rewards and other settings fixed across the three embedding variants of each algorithm. In this ablation, we apply the diversity weights directly to advantages. We compute overall quality gain relative to each algorithm's baseline in this ablation.

As shown in Table 2, our method generalizes across the three tested reward-model embeddings and is not tied to the embeddings of a specific reward model. With each embedding source, we achieve overall quality gains of 5.68% to 6.42% for AWM and 2.19% to 2.98% for DiffusionNFT. We also improve all three training-reward metrics and the held-out ImgRwd and UniRwd scores with every embedding source. Because each tested embedding is computed when its reward model scores the image during training, we can obtain diversity feedback from any of the three reward models without an additional encoder forward pass while retaining overall quality gains with both RL algorithms.

Qualitative analysis. We qualitatively examine how our method affects both image quality and variation within a prompt group. As shown in Figure 4, the AWM baseline often repeats similar high-reward visual patterns across samples, while our method produces more distinct images.

For the prompt "Two cars on the street," the baseline repeatedly generates similar red vehicles with closely matched compositions. Our method instead varies vehicle colors, shapes, viewpoints, and

surrounding street scenes while consistently depicting the requested pair of cars. A similar pattern appears for "A storefront with 'Hello World' written on it." The baseline samples share similar brick facades and signage layouts, with inconsistent rendering of the requested text. Our samples exhibit greater variation in facade color, window arrangement, and sign design, while rendering "Hello World" more consistently and legibly. These examples illustrate how diversity weighting can favor distinct images that remain consistent with the prompt.

5 Related Work

RL for Diffusion and Flow Models. Reverse-trajectory and forward-matching RL differ in how they use generated outputs as training inputs (Xu et al., 2026). Reverse-trajectory methods use final-image rewards to guide policy-gradient updates on sampled denoising transitions, as in DDPO (Black et al., 2024) and DPOK (Fan et al., 2023). Flow-GRPO (Liu et al., 2025) extends this approach to flow models through stochastic sampling and group-relative advantages. Forward-matching methods instead add fresh noise to completed images and train with a reward-dependent prediction loss, decoupling training inputs from the original sampling trajectory. DiffusionNFT (Zheng et al., 2026), AWM (Xue et al., 2026), and AdvantageFlow (Kveton et al., 2026) follow this paradigm with different advantage-weighted losses. Both families use image rewards to compute advantages. Our method reshapes rewards before advantage computation and preserves the sampler, advantage normalization rule, and policy-update rule. It can therefore be applied to any flow RL algorithm with this reward interface, given grouped images and reward-model embeddings.

Scalar Rewards and Diversity Collapse in RLHF. Optimizing rewards assigned independently to each image can reduce diversity by repeatedly reinforcing visually similar samples (Liu et al., 2026a; Tan et al., 2026). GARDO (He et al., 2026) weights positive advantages using nearest-neighbor distances in DINOv3 space, together with gated KL regularization and adaptive reference updates. DiverseGRPO (Liu et al., 2026a) clusters images using DreamSim distances and combines cluster-size rewards with regularization at early denoising steps. DRIFT (Liu et al., 2026b) changes training-sample selection and prompt conditioning and adds a diversity reward. TMPO (Li et al., 2026) matches trajectory probabilities to a reward-induced distribution and shares denoising prefixes during sampling. SQDF (Kang et al., 2026) encourages diverse generation through a maximum-entropy objective, using reparameterized policy gradients of a soft Q-function. Our method instead reshapes rewards within generated prompt groups using existing reward-model embeddings, without an additional encoder or changes to the sampler or policy-update rule. We compute diversity weights with regularized D-optimal design, drawing on related criteria for preference-data selection (Mukherjee et al., 2024; Kveton et al., 2025; Thekumparampil et al., 2025), bandit exploration (Mukherjee et al., 2023), and supervised fine-tuning data selection (Deb et al., 2025).

6 Conclusions

We show that reward-model embeddings can provide diversity feedback that complements scalar quality scores. Our method uses regularized D-optimal design to reshape rewards before advantage computation, without an additional encoder or changes to sampling and policy updates. With AWM and DiffusionNFT on SD3.5-M, we improve overall quality and within-prompt diversity on DrawBench, including gains on all three held-out metrics. We also reach baseline rewards in less training time on Pick-a-Pic.

Limitations. Our experiments use a single backbone at 512×512 resolution and two forward-matching RL algorithms. Further evaluation on larger models, higher resolutions, and reverse-trajectory RL methods is needed to assess how broadly these findings apply. We also use one embedding source per prompt group; combining multiple sources is left for future work.

Artificial intelligence use statement

We used AI coding assistants to help implement methods and evaluations in Flow-Factory, check experiment records and reported statistics, interpret results, and review mathematical exposition. We also used generative AI tools to help revise the writing, format LaTeX, and check spelling and grammar. We did not use generative AI tools to generate synthetic datasets. We have reviewed the AI-assisted outputs and take responsibility for the final content of this work.

Ethics statement

This work does not involve human subjects or new data collection. All experiments use publicly available models, datasets, and reward models. Text-to-image models can produce harmful or biased content; we encourage appropriate safety filtering when deploying fine-tuned models. All authors have read and adhere to the ICLR Code of Ethics.

Reproducibility statement

Experimental hyperparameters are listed in Tables 3 and 4, including shared settings and per-run differences. The models, prompt datasets, and reward definitions are described in Section 4.1; checkpoint selection and evaluation protocols are specified in Section B.3. The allocation solver and SVD reduction are specified in Sections A.1 and A.2. We build on the publicly available Flow-Factory codebase (Ping et al., 2026) and will release our implementation upon acceptance.

References

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

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. *arXiv:2506.15742*, 2025.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Rohan Deb, Kiran Koshy Thekumparampil, Kousha Kalantari, Gaurush Hiranandani, Shoham Sabach, and Branislav Kveton. FisherSFT: Data-efficient supervised fine-tuning of language models using information gain. In *ICML*, 2025.

Mischa Dombrowski, Weitong Zhang, Sarah Cechnicka, Hadrien Reynaud, and Bernhard Kainz. Image generation diversity issues and how to tame them. In *CVPR*, 2025.

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

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

Haoran He, Yuxiao Ye, Jie Liu, Jiajun Liang, Zhiyong Wang, Ziyang Yuan, Xintao Wang, Hangyu Mao, Pengfei Wan, and Ling Pan. GARDO: Reinforcing diffusion models without reward hacking. In *NeurIPS*, 2026.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

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

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 *WACV*, 2025.

Hyeongyu Kang, Jaewoo Lee, Woocheol Shin, Kiyoung Om, and Jinkyoo Park. Diffusion fine-tuning via reparameterized policy gradient of the soft Q-function. In *ICLR*, 2026.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *NeurIPS*, 2024.

J. Kiefer and J. Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960.

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

Branislav Kveton, Xintong Li, Julian McAuley, Ryan Rossi, Jingbo Shang, Junda Wu, and Tong Yu. Active learning for direct preference optimization. *arXiv:2503.01076*, 2025.

Branislav Kveton, Anup Rao, Subhojyoti Mukherjee, Krishna Kumar Singh, and Viet Dac Lai. AdvantageFlow: Advantage-weighted least squares for RL in flow models. *arXiv:2605.26013*, 2026.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Jiaming Li, Chenyu Zhu, Nanxi Yi, Youjun Bao, Li Sun, Quanying Lv, Xiang Fang, Daizong Liu, Jianjun Li, Kun He, Bowen Zhou, and Zhiyuan Ma. TMPO: Trajectory matching policy optimization for diverse and efficient diffusion alignment. In *NeurIPS*, 2026.

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

Henglin Liu, Huijuan Huang, Jing Wang, Chang Liu, Xiu Li, and Xiangyang Ji. DiverseGRPO: Mitigating mode collapse in image generation via diversity-aware GRPO. In *CVPR*, 2026a.

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

Jinmei Liu, Haoru Li, Zhenhong Sun, Chaofeng Chen, Yatao Bian, Bo Wang, Daoyi Dong, Chunlin Chen, and Zhi Wang. Beyond the Dirac delta: Mitigating diversity collapse in reinforcement fine-tuning for versatile image generation. *arXiv:2601.12401*, 2026b.

Subhojyoti Mukherjee, Ruihao Zhu, and Branislav Kveton. Efficient and interpretable bandit algorithms. *arXiv:2310.14751*, 2023.

Subhojyoti Mukherjee, Anusha Lalitha, Kousha Kalantari, Aniket Deshmukh, Ge Liu, Yifei Ma, and Branislav Kveton. Optimal design for human preference elicitation. In *NeurIPS*, 2024.

Bowen Ping, Chengyou Jia, Minnan Luo, Hangwei Qian, and Ivor Tsang. Flow-Factory: A unified framework for reinforcement learning in flow-matching models. *arXiv:2602.12529*, 2026.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

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

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *Transactions on Machine Learning Research*, 2026.

Xiaofeng Tan, Jun Liu, Bin-Bin Gao, Yuanting Fan, Xi Jiang, Chengjie Wang, Hongsong Wang, and Feng Zheng. When policy entropy constraint fails: Preserving diversity in flow-based RLHF via perceptual entropy. In *NeurIPS*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv:2503.20314*, 2025.

Kiran Koshy Thekumparampil, Gaurush Hiranandani, Kousha Kalantari, Shoham Sabach, and Branislav Kveton. Comparing few to rank many: Active human preference learning using randomized Frank-Wolfe method. In *ICML*, 2025.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation. *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: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 *NeurIPS*, 2023.

Yixian Xu, Yuanrui Zhang, Shengjie Luo, Liwei Wang, and Di He. Designing reinforcement learning for diffusion models: A unified path-space view. In *NeurIPS*, 2026.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning RL with pretraining in diffusion models. In *ICML*, 2026.

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 *ICLR*, 2026.

A Additional Method Details

A.1 SVD Reduction

For the exact compact SVD in Equation (8), complete the columns of V to an orthonormal basis of $\mathbb{R}^D$. In this basis, the full design matrix $M(w)$ is block diagonal, with blocks $\widehat{M}(w)$ and γI_{D-n}. Taking log determinants gives Equation (9). Since the two objectives differ by a constant independent of w, they have the same set of optimal allocations.

We compute the SVD in double precision and retain singular values above $\epsilon_{\mathrm{mach}} \max(K, D)\sigma_1$, where ϵ_{mach} is machine precision and σ_1 is the largest singular value of E. The resulting coordinates preserve pairwise inner products up to numerical precision; the equivalence above applies to the exact reduction.

A.2 Allocation Solver

For an update at allocation w, let $a \in \arg\max_i h_i$ be the image receiving weight and $b \in \arg\min_{i:w_i>0} h_i$ the image giving up weight. We update their weights by exact line search.

$$\begin{aligned} \eta^\star &\in \operatorname*{arg\,max}_{0\le\eta\le w_b} \log\det\big(M(w) + \eta(e_a e_a^\top - e_b e_b^\top)\big)\,, \\ w_a &\leftarrow w_a + \eta^\star, \qquad w_b \leftarrow w_b - \eta^\star. \end{aligned} \tag{10}$$

All other weights remain unchanged. To decide when to stop, we use the Frank–Wolfe gap

$$G(w) = \max_i h_i - \sum_{i=1}^{K} w_i h_i. \tag{11}$$

By concavity, $\log\det M(\nu) - \log\det M(w) \le G(w)$, so this gap bounds how much the objective can still improve. For our fixed-strength PickScore configuration, we use $\gamma = 10^{-3}$ and stop when $G(w) \le 10^{-6}$.

A.3 Reward Reshaping Strength

To control the strength of reward reshaping, we interpolate between the original and reshaped rewards, $r_i^\alpha = (1-\alpha)r_i + \alpha\widetilde{r}_i$, with $\alpha \in [0, 1]$. At $\alpha = 0$ we recover the original rewards, and at $\alpha = 1$ we apply the full multiplication. We substitute r_i^α for $\widetilde{r}_i$ in Equation (7) and recompute the reward mean and standard deviation. For numerical stability, our AWM and DiffusionNFT implementations floor the reward standard deviation at 10^{-6} when computing advantages.

Table 3: **AWM hyperparameters for Table 1.** The three variants share optimizer and sampling settings; GARDO uses a larger KL weight. Dashes denote inapplicable settings.

Hyperparameter	Baseline	Ours	GARDO
Optimization			
Optimizer		AdamW	
Learning rate		3×10^{-4}	
Weight decay		10^{-4}	
AdamW (β_1, β_2)		(0.9, 0.999)	
AdamW ϵ		10^{-8}	
Maximum gradient norm		1	
LoRA rank		32	
LoRA scaling parameter		64	
Random seed		42	
Sampling and updates			
Prompts per iteration		32	
Images per prompt		16	
Images per device microbatch		8	
Number of devices		8	
Rollout ODE steps		14	
Training timesteps		6	
Timestep shift		3	
Gradient accumulation steps		48	
Inner epochs		1	
Optimizer updates per iteration		1	
EMA update interval		1	
Timestep sampling interval		$[0, 0.9]$	
EMA decay at optimizer step s		$0.3 \min(s/1000, 1)$	
Algorithm and method settings			
KL weight	10^{-4}	10^{-4}	0.04
Advantage clipping	± 1	± 1	± 1
Generalized Huber exponent		0.25	
EMA KL weight		1	
Ratio-clip offsets		$[-1, 1]$	
Diversity embedding	—	PickScore	DINOv3
Reward interpolation α	—	0.01	—

B Experimental Details

B.1 Training

Tables 3 and 4 list the training hyperparameters for the main comparison. For the embedding ablation in Table 2, we use the corresponding baseline optimization and sampling settings, with advantage clipping of ± 1 for AWM and ± 1.5 for the DiffusionNFT embedding variants (± 5 for its baseline).

During each gradient update, we optimize the current model while holding the rollout and reference models fixed. We apply diversity weighting only during training and evaluate with the original scalar rewards.

For our method in Table 1, we use the reward interpolation in Section A.3 with the embedding source and strength specified in Tables 3 and 4. For both baselines and our runs, we subtract the prompt-group mean reward and divide by the population reward standard deviation over all 512 rollouts across devices.

Table 4: **DiffusionNFT hyperparameters for Table 1.** Optimizer and sampling settings are shared, while KL weights and advantage clipping vary across variants. Dashes denote inapplicable settings.

Hyperparameter	Baseline	Ours	GARDO
Optimization			
Optimizer		AdamW	
Learning rate		3×10^{-4}	
Weight decay		10^{-4}	
AdamW (β_1, β_2)		(0.9, 0.999)	
AdamW ϵ		10^{-8}	
Maximum gradient norm		1	
LoRA rank		32	
LoRA scaling parameter		64	
Random seed		42	
Sampling and updates			
Prompts per iteration		32	
Images per prompt		16	
Images per device microbatch		8	
Number of devices		8	
Rollout ODE steps		10	
Training timesteps		9	
Timestep shift		3	
Gradient accumulation steps		72	
Inner epochs		1	
Optimizer updates per iteration		1	
EMA update interval		1	
Timestep sampling interval		$[0, 0.99]$	
EMA decay at optimizer step s		$\min(0.001s, 0.5)$	
Algorithm and method settings			
KL weight	0.001	10^{-4}	0.04
Advantage clipping	± 5	± 1.5	± 5
NFT β		1	
Diversity embedding	—	PickScore	DINOv3
Reward interpolation α	—	0.01	—

B.2 GARDO

Our GARDO (He et al., 2026) comparisons include diversity weighting of positive advantages, gated KL regularization, and adaptive reference updates. Using DINOv3 (Siméoni et al., 2026) embeddings, we compute each image's cosine distance to its nearest other image from the same prompt and normalize by the prompt-group mean distance. We multiply positive advantages by these normalized distances before clipping. We retain the joint training reward and use the full rollout batch's reward standard deviation to normalize advantages. The gate's uncertainty score is PickScore's win rate minus the mean win rate of HPSv2.1 and CLIPScore. For AWM, we retain the separate KL term involving its exponential moving average (EMA) policy.

The gate fraction starts at 0.1 and is bounded to $[0.02, 0.2]$. We use an uncertainty window of 20 iterations and multiply the gate fraction by 1.1 or 0.9 when uncertainty rises above or falls below the window range, respectively. We reset the reference model when the KL exceeds 0.15 or after 15 updates.

B.3 Evaluation

We evaluate EMA parameters every 50 rollout iterations. For Tables 1 and 2, we evaluate the final step-1,000 EMA checkpoints on DrawBench. We compute the six quality metrics and diversity in

Table 1 from the same 800 images per checkpoint. For Table 2, we use logged DrawBench scores for the training-reward metrics and compute the held-out metrics from the saved EMA checkpoints. The qualitative examples use selected prompt groups from these embedding-ablation runs.

Training efficiency. We average the summed PickScore, CLIPScore, and HPSv2.1 scores over images generated for Pick-a-Pic prompts, using the scaling defined in Section 4.1. We define training efficiency as the ratio of the training time when the baseline attains its highest reward to the training time when our method first attains the same or a higher reward. We use the baseline's highest recorded reward within the first 60 GPU hours and compute both training times by linear interpolation between evaluations.