REDIF: RESOURCE-EFFICIENT FEW-STEP DIFFUSION DISTILLATION VIA REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Step distillation accelerates diffusion sampling by training a few-step student to imitate a many-step teacher, but distillation itself remains expensive. Typically, this requires thousands of GPU-hours and a large pre-generated trajectory dataset. We introduce ReDiF, which casts step distillation as terminal-reward policy optimization rather than step-wise regression. The student is optimized against a reward computed on the terminal sample, measuring alignment with the teacher's output, instead of matching the teacher's intermediate trajectory under a reconstruction or consistency loss. Because the reward need not be differentiable or trajectory-aligned, ReDiF admits non-differentiable objectives, multi-objective combinations, and preferences the teacher does not express, while exploration lets the student find sampling paths matched to its own step schedule. ReDiF converges in 400 policy updates with 3200 rollouts on a single A100 GPU with 1,000 noise-class pairs and no paired dataset: about 4 GPU-hours, against roughly 336 A100-hours reported for DMD2 on the same EDM teacher. At 8 steps on ImageNet-64, it achieves an FID 3.64 points better than the strongest retrained distillation baseline in the low-training regime under the same single-GPU budget. The formulation is also orthogonal to existing distillation objectives: added to the DMD2 loss, it further improves DMD2's FID by 4.4.

1 INTRODUCTION

Sampling from diffusion models can be slow and computationally expensive because high-quality generation typically requires iterative denoising over many steps, while naively reducing the step count can degrade sample quality (Ho et al., 2020; Song et al., 2021b; Karras et al., 2022). Training-free approaches try to reduce the step count with better numerical solvers; they are cheap to apply but degrade sharply at very low step counts, where numerical approximation error leads to reduced fidelity and loss of fine-grained detail (Song et al., 2021a;b; Lu et al., 2022; Zheng et al., 2023a; Shih et al., 2023). Distillation instead adapts the model itself: a teacher running many denoising steps transfers its behavior to a cheaper student, trained to operate effectively at very low step counts where conventional solvers can suffer substantial quality degradation (Salimans & Ho, 2022; Song et al., 2023; Zheng et al., 2023b). Distillation can reduce the number of denoising steps, the model size, or both; we focus on step distillation, although the formulation is not tied to a particular student architecture.

Existing distillation methods differ primarily in how they supervise the student. Trajectory-based approaches learn from the teacher's sampling trajectory: Progressive Distillation halves the number of steps at each training stage, so that each stage only has to absorb a small reduction in step count (Salimans & Ho, 2022), and Consistency Distillation trains the student to map any point on the teacher's trajectory directly to its endpoint (Song et al., 2023; Luo et al., 2023a; Kim et al., 2024). Distribution-matching methods such as DMD, DMD2, SiD, and SiDA instead optimize objectives that approximate the teacher's output distribution (Yin et al., 2024b;a; Zhou et al., 2024a; 2025b).

These approaches have substantially improved few-step generation, to the point where one-step students match or exceed their teachers (Yin et al., 2024a; Zhou et al., 2024a; 2025b), but at a high training cost: several thousand A100-hours, at effective batch sizes in the thousands (Zhou et al., 2024b; Liu et al., 2024; Yin et al., 2024a). For many of these methods, the effective batch size is a particularly restrictive constraint. While computation can be traded against wall-clock time, a large

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

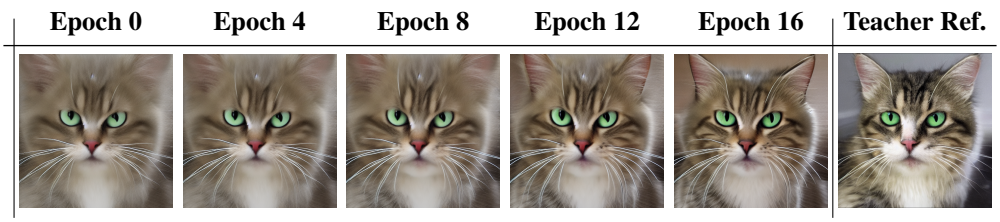


Figure 1: ReDiF training on the Stable Diffusion v1.5 branch ($50 \rightarrow 5$ steps). Within a few epochs a student sampling at a tenth of the teacher's step count reaches a quality close to the teacher's reference (rightmost column), although the only supervision is a scalar computed on the final image. Columns are successive epochs of one run at fixed conditioning. Figure 4 continues this run past the budget.

effective batch imposes a hardware requirement that cannot be removed simply by training longer on a single GPU, which can make the distillation of large models difficult without access to multi-GPU infrastructure. This motivates a different question: can diffusion distillation be carried out with far less compute, training time, and batch size, within the limited resource budget of a single GPU?

We introduce ReDiF, which formulates step distillation as terminal-reward policy optimization. Starting from the teacher's parameters, the student generates a few-step sample and is optimized against a scalar reward measuring its alignment with the teacher's output from the same conditioning. Because the student is initialized with the teacher's weights, optimization focuses on correcting the discrepancy introduced by the coarser sampling schedule rather than learning the generative process from scratch. Unlike conventional distillation objectives, ReDiF requires neither step-wise supervision, auxiliary score networks or discriminators, nor paired teacher–student trajectories, and the reward may be non-differentiable or multi-objective.

ReDiF distills a 256-step EDM teacher to 8 steps on a single A100 using 1,000 noise-class pairs in approximately 4 GPU-hours, compared with the $\sim$336 A100-hours reported for DMD2 on the same teacher. Under the same single-GPU budget, ReDiF achieves an FID 3.64 points lower than the strongest retrained distillation baseline. Moreover, adding ReDiF to the DMD2 objective further improves DMD2's FID by 4.4 points, demonstrating that the proposed objective is complementary to existing distillation approaches.

2 Related Work

Trajectory-based distillation. KDIGM (Luhman & Luhman, 2021) first explored compressing a multi-step diffusion model into a one-step student, but at a substantial cost in sample quality. Progressive Distillation (Salimans & Ho, 2022) halves the teacher's sampling steps over multiple training stages, while Guided Distillation (Meng et al., 2023) extends this approach to classifier-free guided samplers. Consistency Distillation (Song et al., 2023) replaces the recursive procedure with a self-consistency objective along the probability-flow ODE, training the student to map any point on the trajectory to its endpoint. RDD (Feng et al., 2024) and SFD (Zhou et al., 2024b) refine this objective, LCM and LCM-LoRA (Luo et al., 2023a;b) extend it to latent diffusion, and CTM (Kim et al., 2024) generalizes it to arbitrary trajectory jumps. InstaFlow (Liu et al., 2024) instead straightens the sampling trajectory using rectified flow. These methods derive supervision from teacher trajectories or intermediate states, which can constrain the student's sampling behavior under aggressive step reduction.

Distribution matching. An alternative line of work matches the student's output distribution to the teacher's rather than explicitly imitating its trajectory. DMD (Yin et al., 2024b) uses an approximate reverse KL estimated from score differences, while SiD (Zhou et al., 2024a) relies on score identities for distribution matching. DMD2 (Yin et al., 2024a), SiDA (Zhou et al., 2025b), and ADD (Sauer et al., 2024) incorporate adversarial components. These methods achieve highly competitive few-step generation quality, but introduce additional machinery, such as auxiliary networks and, for the adversarial variants, real training images. TFD (Zhang et al., 2026) removes the auxiliary network through a hidden-state-based objective, but still relies on reference data in the settings it considers.

Data-free distillation. Data-free methods remove the dependence on real training images by using synthetic teacher-generated supervision. BOOT (Gu et al., 2023) trains on consecutive teacher samples, DKDM (Xiang et al., 2025) manages the required synthetic data, and PMDD (Tran et al., 2025) extends the setting to progressive multi-step distillation. Guided SiD (Zhou et al., 2025a), DiffDFKD (Qi et al., 2025), and Random Conditioning Distillation (Kim et al., 2025) address conditioning-space coverage without real images. Although these methods eliminate real data, they generally retain expensive supervised or distribution-matching objectives and rely on large synthetic datasets or a large number of conditioning signals.

Reward-based optimization. Reward-based optimization has primarily been used to align diffusion models with external objectives. DDPO (Black et al., 2024) formulates diffusion sampling as an MDP and optimizes terminal rewards using policy gradients, while DPOK (Fan et al., 2023), Diffusion-DPO (Wallace et al., 2024), LOOP (Gupta et al., 2026), and RLCM (Oertell et al., 2024) explore related alignment and optimization settings. DRaFT (Clark et al., 2024) backpropagates rewards through the sampling chain, and LaSRO (Jia et al., 2025) learns a latent surrogate reward. Closest to our setting, RG-LCD (Li et al., 2024) directly optimizes a reward to compensate for the quality degradation of LCM, while retaining the consistency loss as its primary objective. Unlike ReDiF, it does not use reinforcement learning and backpropagates the reward directly. Its main contribution is preventing reward hacking, while using rewards for alignment and quality improvement rather than as the sole objective for distillation.

Gap and our approach. Existing distillation methods primarily rely on trajectory-level supervision or distribution-matching objectives, while reward-based approaches have mainly been used for alignment or as auxiliary terms in distillation. ReDiF addresses this gap by making terminal-reward RL-based optimization the sole training objective for step distillation. The student is initialized with the teacher's weights and optimized using a reward computed on its final sample, without step-wise reconstruction, auxiliary networks, real training images, or paired teacher–student data. Appendix A compares ReDiF with existing methods along these axes.

3 METHOD

ReDiF formulates diffusion step distillation as RL-based terminal-reward policy optimization. The student rolls out a complete denoising trajectory with its current weights, its final image is compared with the teacher's final image from the same noise and conditioning, and the resulting scalar is the only supervision it receives; no intermediate state is ever given a target. Figure 2 shows the loop.

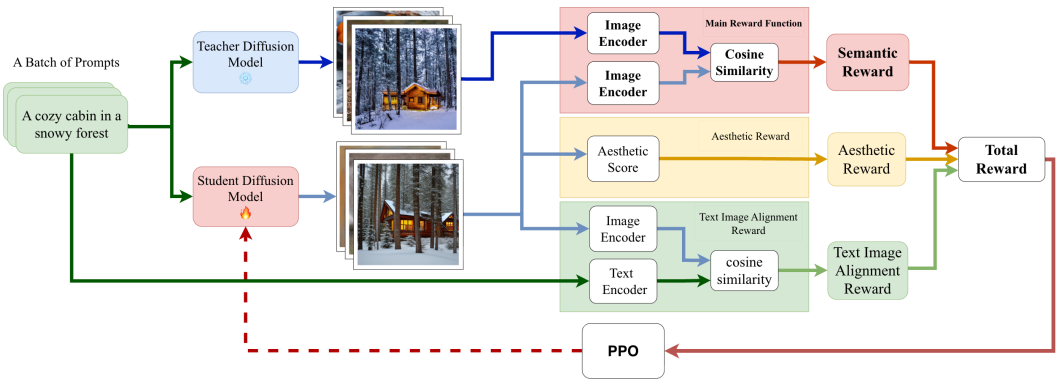


Figure 2: Overview of ReDiF. The same noise and conditioning are provided to both the frozen teacher and the trainable student. Their final images are embedded using frozen, pretrained encoders, and the similarity between the embeddings serves as the terminal reward, which PPO maximizes by updating only the student. Everything to the right of the two generators is frozen and operates in a forward-only manner, while the teacher reference is generated by the teacher's own sampler within the training loop. Aesthetic and Text-Image Alignment are two auxiliary preference rewards.

3.1 WHY PER-STEP SUPERVISION IS COSTLY

Let Φ^* be the teacher's N-step sampler and Φ_θ the student's M-step sampler, $M \ll N$. Trajectory-matching methods give each supervised transition a target y_k and minimize $\mathcal{L}_{\text{step}}(\theta) = \sum_k \mathbb{E}\,\ell(\Phi_\theta(x_{t_k}, t_k), y_k)$, with y_k a teacher prediction for reconstruction and progressive losses and an EMA copy of the student for consistency losses; distribution-matching methods match the two output distributions through an auxiliary score network. Three sources of cost motivate our design; they are arguments, not lower bounds, and Section 4 tests the design empirically.

Per-step errors compound. A deviation at one transition is propagated by all later ones. If the teacher's maps between consecutive student grid points are L_k-Lipschitz, the terminal error is at most $\Gamma \max_k \varepsilon_k$ with $\Gamma = \sum_k \prod_{j<k} L_j$, which is at least M whenever $L_j \geq 1$ and exponential in M when $L_j \geq L > 1$ (Lemma 1, Appendix B.1). A terminal tolerance δ is therefore guaranteed by a per-step tolerance of δ/Γ: stricter than the accuracy that is eventually measured, which is taken on the final image alone.

Moving targets. When y_k comes from a network that itself changes during training, as the EMA target of consistency distillation or the fake-score network of distribution matching, its error has a systematic component that more samples of the same target do not average away (Appendix B.2). Such networks must also be re-fitted to the student throughout training, at a cost of the same order as the student's own update (Appendix C), and published distribution-matching methods train at effective batch sizes from 128 to 8,192 (Table 8). We treat this as an empirical constraint of single-device training rather than as a property we prove.

The problem is a fine-tuning problem. The student starts at the teacher's weights and needs to compensate only for a coarser schedule; it does not learn the generative process from scratch. Policy optimization is known to adapt pretrained diffusion models to a new reward in comparatively few updates (Black et al., 2024; Oertell et al., 2024), and step distillation satisfies that premise by construction. This is a working hypothesis, not a theorem: we do not bound how far the optimal student lies from the teacher, and check it on the curves of Appendix I.1.

Together these suggest measuring agreement only where it is evaluated, at the terminal sample, estimating no intermediate target, and training nothing besides the student: scoring the finished image with frozen encoders satisfies all three.

3.2 STEP DISTILLATION AS A POLICY OPTIMIZATION PROBLEM

Each student transition conditions only on the current state $s_t = (x_t, t, c)$, so the sampler is Markov by construction and defines a finite-horizon MDP with horizon M, pure noise as the initial state and the generated image as the terminal state. PPO needs a likelihood ratio, so the action must have a density. Following DDPO (Black et al., 2024), the action is the sampled next latent,

$$a_t = x_{t-1} \sim \pi_\theta(x_{t-1} \mid s_t) = \mathcal{N}\big(\mu_\theta(x_t, t, c),\, \sigma_t^2 I\big),$$

where μ_θ is the teacher's sampler update on the student's M-step schedule, applied to the student's denoiser output, and σ_t^2 is the variance of the stochastic form of that update, fixed by the scheduler. Only μ_θ is learned, and only through the LoRA adapter (Hu et al., 2022).

Supervision exists only once a trajectory has produced an image, so the reward is sparse and terminal: $r_t = 0$ for $t > 0$, $r_0 = R(x_0)$ and $J(\theta) = \mathbb{E}_\tau[R(x_0)]$. The score-function identity (Williams, 1992; Sutton et al., 1999) spreads the scalar over all transitions,

$$\nabla_\theta J = \mathbb{E}\Big[(R - b) \textstyle\sum_{k=1}^{M} \nabla_\theta \log \pi_\theta(x_{t_{k-1}} \mid x_{t_k}, t_k, c)\Big],$$

for any baseline b independent of the actions (Proposition 1, Appendix B.3). No intermediate target appears, and the reward enters only through its value, so it may be non-differentiable. The price is variance: one scalar multiplies a sum of M score terms, and the variance grows with the horizon. This is one of the two pressures that fix the student step count (Appendix M).

The ratio requires $\sigma_t > 0$, so rollouts inject noise at every transition, whereas evaluation uses the deterministic sampler ($\eta = 0$). Training changes only μ_θ, so the deployed sampler applies the function that was optimized, but along noise-free rather than perturbed states. We do not claim this mismatch is harmless; every reported number is measured in the deployed configuration, so it includes whatever the mismatch costs.

3.3 THE SEMANTIC TERMINAL REWARD

We compute the reward in representation space,

$$R(I_T, I_S) = \sum_{i=1}^{K} \lambda_i \, \mathcal{S}_i(f_i(I_T), f_i(I_S)) \,, \qquad \sum_i \lambda_i = 1,$$

where I_T and I_S are the teacher and student images from the same (x_T, c), f_i are frozen encoders (CLIP (Radford et al., 2021), DINOv3 (Siméoni et al., 2025), the Perception Encoder (PE) (Bolya et al., 2025) and LPIPS (Zhang et al., 2018)), and $\mathcal{S}_i$ is cosine similarity between normalized embeddings, or the negative distance for LPIPS. The teacher side $f_i(I_T)$ does not depend on θ: I_T comes from the teacher's own sampler inside the training loop, on the conditioning signal the rollout already uses. Drawing a reference anew per rollout or reusing it across rollouts that share a conditioning signal leaves the objective unchanged; neither assembles a dataset of teacher trajectories.

Because teacher and student start from the same noise and conditioning, the reward asks the student to land where the teacher lands on that input, not only to match its output distribution. The teacher's sampler is stochastic, so the reference is a sample from its conditional distribution rather than a fixed target: the student is scored per sample without score estimation, on a scale comparable across conditionings, so a high reward means agreement rather than an easy conditioning. Section 3.4 relies on this when it standardizes rewards over the batch.

A pixel-space reward penalizes every deviation from the teacher's image, including texture and sampling noise that do not affect perceived agreement. A representation-space reward is, to first order, invariant to every perturbation in the null space of the encoder's Jacobian (Proposition 2, Appendix B.4), so it constrains fewer directions. This cuts both ways: the student may reach any terminal state the encoder cannot distinguish from the teacher's, by whichever path suits its schedule, but those same directions are where the policy drifts once the reward stops improving elsewhere (Appendix J). Summing encoders shrinks the invariant set to the intersection of the null spaces; whether that helps depends on the encoder and branch (Appendix I).

3.4 OPTIMIZATION

We optimize the clipped PPO surrogate (Schulman et al., 2017),

$$\mathcal{L}_{\mathrm{PPO}}(\theta) = \mathbb{E}_t\Big[\min\Big(\rho_t(\theta)\hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big], \qquad \rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)},$$

with π_θ the policy of Section 3.2 under the guidance convention set out at the end of this section. The horizon is short and the reward terminal, so we use no value function and no learned baseline. All transitions of a trajectory share one advantage, the trajectory's reward standardized over the rollout batch $\mathcal{B}$ of that iteration, as in DDPO (Black et al., 2024): $\hat{A} = \mathrm{clip}\left((R - \mu_{\mathcal{B}})/(\sigma_{\mathcal{B}} + \varepsilon), -A_{\max}, A_{\max}\right)$, with ε keeping it defined when all rewards agree and $A_{\max} = 5$ bounding an outlying rollout. We do not group rollouts by conditioning and use no group-relative advantage: since $\mathcal{B}$ mixes conditionings, this removes the difficulty of the batch rather than of one conditioning, and a shared baseline is reasonable because each rollout is scored against its own teacher reference (Section 3.3). Each iteration collects rollouts with θ_{old} frozen, stores the sampled latents and their log-densities, then runs E inner epochs of minibatch updates. Following DDPO the clip ranges are small (Table 4), because one update of shared weights moves the log-density of every transition at once.

Clipping bounds each update but not the cumulative displacement over many, so we add a regularizer toward the frozen teacher policy, $\mathcal{L} = \mathcal{L}_{\mathrm{PPO}} - \beta D_{\mathrm{KL}}(\pi_\theta \| \pi_T)$, which for Gaussian policies with shared variance reduces to $\|\mu_\theta - \mu_T\|^2/(2\sigma_t^2)$ per transition. As a loss term rather than a reward term, its strength is set independently of the reward scale and does not pass through standardization. Since the student is the teacher plus a LoRA adapter, μ_T comes from disabling the adapter, at one teacher evaluation per transition when $\beta > 0$ (Appendix C). We use $\beta = 0.02$ on ImageNet-64 and $\beta = 0$ on COCO. Because the reward keeps rising after quality saturates, the training length is fixed from held-out FID rather than from the reward (Appendix J). Algorithm 1 (Appendix E) summarizes the procedure.

LoRA as an implicit constraint. Training the student as a low-rank adapter also restricts how far the policy can move, a role the clip range and the KL term play on other quantities. It keeps FID and seed spread stable across clip ranges, whereas a full-parameter student depends strongly on both (Appendix L).

Guidance-consistent policies. The PPO ratio assumes that actions are drawn from the distribution whose log-density the update evaluates. Classifier-free guidance changes that distribution: with scale w the transition mean is $\mu_\theta^{(w)} = \mu_\theta(x_t, t, \varnothing) + w(\mu_\theta(x_t, t, c) - \mu_\theta(x_t, t, \varnothing))$, so trajectories are generated by $\pi_\theta^{(w)} = \mathcal{N}(\mu_\theta^{(w)}, \sigma_t^2 I)$, not by the conditional branch alone. Sampling from $\pi_{\theta_{\text{old}}}^{(w)}$ while evaluating log-densities under the unguided π_θ compares two distributions rather than successive iterates of one policy: the ratio differs from 1 already at $\theta = \theta_{\text{old}}$, so the first inner epoch is biased and the clip range no longer bounds the policy change it is meant to bound. Consistency requires the same w in rollout and update.

The distinction is vacuous for the class-conditional EDM teacher, which we run without guidance. For Stable Diffusion the guided composition with $w = 5$ is the policy in both rollout and update, doubling the network evaluations per transition; the reward, a function of the terminal image only, is unaffected.

3.5 RL versus Direct Backpropagation

The rewards of Section 3.3 are computed by differentiable encoders, so they could be backpropagated through the unrolled sampler as in DRaFT (Clark et al., 2024). We treat them as black boxes for four reasons. *Credit assignment:* policy optimization draws several trajectories per iteration, forms a baseline from their rewards, and reinforces the transitions of trajectories that scored above it while suppressing those that scored below. Every transition, including an early one whose only merit is setting up a good endpoint, is therefore credited by comparing sampled alternatives, without differentiating the sampler. Direct backpropagation also credits early transitions, but through the local derivative of a single deterministic trajectory, carried back through the Jacobians of every later step, and a truncated backward pass gives the earliest transitions no credit at all. In expectation over the same stochastic chain the two estimate the same gradient (Mohamed et al., 2020); they differ in variance and in what each sample requires. *Memory:* a pathwise gradient must retain, or recompute, the graph of every sampler step it differentiates through, whereas the score-function estimator needs one transition at a time. *Generality:* policy optimization uses only the value of the reward, so non-differentiable rewards need no change; the preference terms below are exactly that. *Reward exploitation:* pathwise ascent follows the encoder's input gradient exactly in every sample, including directions that raise the score without improving the image (Li et al., 2024), whereas policy optimization never evaluates that gradient, follows it only in expectation and smoothed by the policy noise, and clips each update; it is not immune (Appendix J). Section 4.2 compares the two under an identical reward, initialization and budget; the direct-backpropagation rows also use a deterministic chain and a truncated backward pass, so the comparison measures these differences together.

Adding preferences to the distillation reward. Because distillation is already cast as reward maximization, a preference is added by adding a term. With R_{pref} a score on the student's image alone,

$$R_{\text{tot}}(I_T, I_S) \;=\; R(I_T, I_S) \;+\; \gamma\, R_{\text{pref}}(I_S). \tag{1}$$

Nothing else in Sections 3.2–3.4 changes: the rollouts, the teacher reference, the advantage and the PPO update are identical, and R_{pref} is evaluated on the same decoded image at one extra function evaluation per rollout, with no reference image and no gradient. We use the JPEG-encoded size and a counting-based sharpness score built from the width of luminance transitions, both piecewise constant, so a pathwise method receives zero gradient (Appendix K).

This is not a new capability: the score-function estimator has always admitted such rewards, and DDPO (Black et al., 2024) already optimizes JPEG compressibility. What the terminal formulation contributes is that distillation and preference are the same kind of objective, so combining them needs no second loss, optimizer or surrogate.

Two practical points govern equation 1. Advantages are standardized after the two terms are summed, so the preference's influence is set by γ times the spread of R_{pref} across the rollout batch relative to that of R, not by their absolute scales. And the two conflict by construction, since the

teacher reference does not have the preferred property, so the optimum is a compromise in which the distillation term anchors the student; preference rewards are proxies whose cheapest optimum is often degenerate (Appendix J). That no change to the procedure is needed we check on the COCO student, trained under the settings and budget of Table 1 with the added term as the only difference: Figure 8 follows two prompts and seeds through training under each preference, which pull in opposite directions on the same image statistic. It shows ease of use, not how much fidelity a preference costs.

4 EXPERIMENTS

4.1 SETUP

Teachers. We distill a class-conditional EDM teacher (Karras et al., 2022) on ImageNet-64 in pixel space from 256 to 8 steps, and Stable Diffusion v1.5 (Rombach et al., 2022) in latent space from 50 to 5 steps, evaluated on COCO (Lin et al., 2014); SD v1.5 rather than SDXL (Podell et al., 2024) because every experiment in this paper, including every retrained baseline, runs on one NVIDIA A100 (80 GB).

Metrics and protocol. We report FID (Heusel et al., 2017), PRDC (Naeem et al., 2020) and, on COCO, CLIPScore (Hessel et al., 2021), treating Recall and Coverage as primary alongside FID because reward-driven objectives admit reward-maximizing solutions with degenerate diversity. FID uses 10,000 generated samples against 10,000 real validation images on ImageNet-64 and 5,000 validation prompts on COCO. Absolute values under this protocol are higher than published ones (the 256-step teacher reaches 13.74 against a published 1.36), so they are comparable only within a table and published numbers are kept separate (Appendix N). All reported checkpoints are the last checkpoint of the budget, ours and the baselines' alike. Appendix P gives the full protocol.

Student step count. M is bounded below by the need for a horizon over which the policy can explore, and above by estimator variance and per-iteration cost. We use $M = 8$ on ImageNet-64 and $M = 5$ on COCO; Appendix M reports the sweep over $M \in \{1, 2, 4, 8, 16\}$ behind the ImageNet-64 choice.

Training and cost accounting. ReDiF trains a LoRA adapter of rank 4, with effective batch 32 on ImageNet-64 and 8 on COCO, from conditioning pools of 1,000 (noise, class-label) pairs and 160 prompts. No real images enter training and no auxiliary network is trained: each rollout is scored against a reference produced by the teacher's own sampler during training, so no paired dataset exists at any point. Those teacher evaluations are the largest single term in the training cost and are included in the PFLOPs column (Appendix C). Two reward configurations are carried forward from Appendix I: $\mathbf{R_1}$ is CLIP alone, the cheapest instance of Section 3.3, and $\mathbf{R_2}$ sums CLIP, DINOv3, PE and LPIPS with equal weights. R_1 is the core reward function and works well on its own, which is why we carry it throughout as the single-encoder reference. R_2 is our main reward: it tracks R_1 closely, is ahead of it in places, and because a sample must satisfy four encoders at once rather than one, it is the more robust of the two against the single-encoder shortcuts behind reward hacking (Appendix J). Hyperparameters are in Appendix H.

4.2 COMPARISON IN THE LOW-TRAINING REGIME

Our claim is about the low-training regime: what an objective delivers on one GPU in a few hours, not what it reaches at convergence. Table 1 compares methods inside it. Every row is trained by us from the same teacher initialization, on the same A100, at the same student step count, from the authors' official implementation adapted to a LoRA student, at the hyperparameters their own papers report (Appendix O). CTM and TFD are absent: no setting we tried put them at a reasonable operating point under this budget, so we quote them only at their authors' (Appendix P). Every row trains for the same number of epochs over the same conditioning pool and reads the same pool of teacher samples, synthesized once before training; the PFLOPs column is the analytical cost of reproducing a row from scratch, pool included, and is defined in Appendix C. Table 8 is the complementary view, in which each method is quoted at the budget its authors judged it to need.

Results. On ImageNet-64 both ReDiF configurations reach the lowest FID, 18.60 and 18.84 against 22.24 for the strongest baseline, consistency distillation; the gap of 3.64 exceeds the across-seed

Table 1: **The low-training regime.** All rows are trained by us on one A100, from the same teacher initialization and for the same number of epochs, under an otherwise identical protocol (Appendix P). PFLOPs is the analytical cost of reproducing a row from scratch, including the teacher-sample pool that is synthesized once and read by every row and that dominates it; Appendix C defines it. GPU-h is wall-clock on the same device; cost columns are not ranked. Pub. FID is the authors' own number under their protocol and step count and is *not* comparable to the FID column. Mean ± sample sd, last checkpoint; per branch, best in bold, second best bold and dash-underlined, teacher rows unranked.

Method	Steps	Batch	GPU-h	PFLOPs	P ↑	R ↑	D ↑	C ↑	FID ↓	Pub. FID
(a) EDM teacher, ImageNet-64 (256 → 8)										
Teacher (256 steps, reference)	256	–	–	–	0.8310	0.7103	0.8930	0.7831	13.74	1.36
Teacher (8 steps)	8	–	–	–	0.5308	0.4780	0.3910	0.5326	35.04	–
Progressive Distill. (Salimans & Ho, 2022)	256→8	32	3.39	1137.6	0.5995 ± 0.0017	0.5591 ± 0.0073	0.6171 ± 0.0010	0.6770 ± 0.0001	38.02 ± 0.19	15.39
Consistency Distill. (Song et al., 2023)	256→8	32	3.30	1138.5	0.7110 ± 0.0028	0.5387 ± 0.0190	0.4004 ± 0.0043	0.6219 ± 0.0051	22.24 ± 0.68	6.20
DMD (Yin et al., 2024b)	256→8	128	3.25	1191.4	0.7327 ± 0.0043	0.4596 ± 0.0072	**0.7751 ± 0.0031**	0.7602 ± 0.0043	32.83 ± 0.66	2.62
DMD2 (Yin et al., 2024a)	256→8	128	4.07	1365.9	**0.7791 ± 0.0039**	0.6162 ± 0.0058	0.5349 ± 0.0016	0.7881 ± 0.0034	25.67 ± 0.45	1.28
SiD (Zhou et al., 2024a)	256→8	32	3.93	1153.5	0.7054 ± 0.0037	0.5453 ± 0.0001	0.6978 ± 0.0006	0.7621 ± 0.0032	40.31 ± 0.06	1.524
SiDA (Zhou et al., 2025b)	256→8	32	3.03	1157.6	0.7057 ± 0.0057	0.5281 ± 0.0006	0.7304 ± 0.0023	0.7313 ± 0.0042	35.36 ± 0.13	1.110
Direct backprop. (R_1: CLIP)	256→8	32	3.36	1156.0	0.6747 ± 0.0600	0.5842 ± 0.0037	0.6273 ± 0.0081	0.7980 ± 0.0320	21.72 ± 1.33	–
Direct backprop. (R_2: CLIP+DINOv3+PE+LPIPS)	256→8	32	3.94	1157.8	0.6631 ± 0.0024	0.5783 ± 0.0134	0.4716 ± 0.0072	0.6823 ± 0.0193	19.76 ± 0.24	–
ReDiF (R_1: CLIP)	256→8	32	3.67	1212.1	**0.7529 ± 0.0180**	**0.6265 ± 0.0213**	0.6318 ± 0.0291	**0.8223 ± 0.0169**	**18.84 ± 2.01**	–
ReDiF (R_2: CLIP+DINOv3+PE+LPIPS)	256→8	32	3.84	1213.5	0.7432 ± 0.0084	**0.6221 ± 0.0158**	**0.7991 ± 0.0101**	**0.8022 ± 0.0129**	**18.60 ± 1.63**	–
(b) Stable Diffusion v1.5 teacher, COCO (50 → 5)										
Teacher (50 steps, reference)	50	–	–	–	0.6103	0.2029	0.8878	0.6832	24.59	8.74
Teacher (5 steps)	5	–	–	–	0.4420	0.1762	0.6291	0.5813	31.45	–
Progressive Distill. (Salimans & Ho, 2022)	50→5	8	0.93	18.16	0.4395 ± 0.0012	0.1653 ± 0.0010	0.6420 ± 0.0011	0.6032 ± 0.0002	31.60 ± 2.38	26.00
Consistency Distill. (Song et al., 2023)	50→5	8	0.68	17.90	0.4414 ± 0.0021	0.1742 ± 0.0004	0.6443 ± 0.0071	0.6023 ± 0.0009	31.79 ± 0.22	23.62
DMD (Yin et al., 2024b)	50→5	8	0.58	16.32	**0.5446 ± 0.0002**	0.1781 ± 0.0009	0.6720 ± 0.0011	0.6033 ± 0.0020	32.04 ± 0.70	11.49
DMD2 (Yin et al., 2024a)	50→5	8	1.12	19.03	0.4655 ± 0.0003	0.1771 ± 0.0013	0.6420 ± 0.0029	0.6052 ± 0.0017	31.60 ± 0.02	8.35
Direct backprop. (R_1: CLIP)	50→5	8	0.68	16.48	0.4474 ± 0.0051	0.1679 ± 0.0046	0.6427 ± 0.0074	0.6001 ± 0.0061	31.82 ± 0.24	–
Direct backprop. (R_2: CLIP+DINOv3+PE+LPIPS)	50→5	8	0.72	16.52	0.4716 ± 0.0112	**0.1911 ± 0.0034**	**0.6954 ± 0.0203**	0.6312 ± 0.0071	29.56 ± 0.57	–
ReDiF (R_1: CLIP)	50→5	8	0.54	17.02	0.4890 ± 0.0062	0.1864 ± 0.0047	**0.7411 ± 0.0087**	**0.6682 ± 0.0065**	**27.42 ± 0.84**	–
ReDiF (R_2: CLIP+DINOv3+PE+LPIPS)	50→5	8	0.58	17.04	**0.5193 ± 0.0083**	**0.1896 ± 0.0054**	0.6301 ± 0.0095	**0.6812 ± 0.0056**	**26.36 ± 0.78**	–

standard deviation of either, and ReDiF's Coverage exceeds the 256-step teacher's (0.8223 against 0.7831), while DMD2 keeps the highest Precision. On COCO the baselines barely move from the 5-step teacher (FID 31.60–32.04 against 31.45) while ReDiF reaches 26.36 and 27.42 with the highest Coverage and Density. Held-out samples are in Appendix G.

Policy optimization versus direct backpropagation. The two direct-backpropagation rows share the reward, initialization, schedule, conditioning pool, batch and budget of the corresponding ReDiF rows; only the optimizer differs, its backward pass truncated to fit in 80 GB. On ImageNet-64 policy optimization wins on FID for both rewards (18.84 against 21.72, 18.60 against 19.76); on COCO it wins in FID, Precision, Coverage and Density, while direct backpropagation with R_2 takes the highest Recall. The advantage is consistent in FID, not uniform across diversity metrics.

Are the baselines simply under-trained? Not relative to ReDiF: every row saw the same epochs and the same pool of teacher samples. They are far from their published numbers because this regime lies two to four orders of magnitude below those budgets, and because DMD, DMD2, SiD and SiDA are run outside the one-step regime they were built for. Table 8 shows each method at its authors' budget, and Appendix D plots FID against epochs: ReDiF leads every baseline at every evaluated epoch, by the widest margin in the low-budget regime, and no baseline reaches it anywhere in the measured range.

4.3 ReDiF as a Plug-In Objective

The construction needs only a stochastic policy over denoising transitions and a scalar terminal reward, so it can be added to existing methods as well as substituted for them. Table 2 tests two integration modes on the EDM teacher. In the *structural* mode, progressive and consistency distillation keep their sampler or parameterization, including the intermediate teachers and the EMA target, but their supervised loss is replaced by the PPO objective; in the *loss* mode, DMD2 keeps its auxiliary networks and optimizes its loss jointly with the semantic reward. Every pair uses the budget of Section 4.2, so the baseline rows repeat Table 1.

In the loss mode, adding the terminal reward to DMD2 improves all five metrics, lowering FID from 25.67 to 21.32 and raising Coverage from 0.7881 to 0.8742: the two signals are not redundant, since at single-device batch sizes distribution matching alone reaches a worse point than ReDiF. The combination nonetheless stays above ReDiF alone in FID and keeps its auxiliary-network cost. In the structural mode, the terminal reward lowers progressive distillation's FID by 3.9 and raises its Precision and Coverage, at a larger seed spread; for consistency distillation it raises Precision,

Table 2: ReDiF's PPO objective combined with existing methods on the EDM teacher, ImageNet-64 ($256 \rightarrow 8$), under the same epoch budget as Section 4.2. Each "+RL" row keeps its baseline's architectural modification. PFLOPs is total analytical training compute, teacher forwards included. Mean $\pm$ sample sd over seeds, last checkpoint. The DMD2 rows keep the auxiliary score network and are not cost-comparable to the rest; ReDiF alone reaches 18.60. Within each baseline/"+RL" pair the better entry is bold; rows are compared within a pair, not across the table.

Method	Mode	Aux. net	PFLOPs	P ↑	R ↑	D ↑	C ↑	FID ↓
Progressive Distill. (Salimans & Ho, 2022)	–	interm. teachers	1137.6	0.5995 ± 0.0017	$\mathbf{0.5591 \pm 0.0073}$	0.6171 ± 0.0010	0.6770 ± 0.0001	38.02 ± 0.19
+ RL (ours)	structural	interm. teachers	1342.4	$\mathbf{0.7321 \pm 0.0424}$	0.5250 ± 0.1136	$\mathbf{0.6472 \pm 0.0601}$	$\mathbf{0.7950 \pm 0.0451}$	$\mathbf{34.08 \pm 2.71}$
Consistency Distill. (Song et al., 2023)	–	EMA target	1138.5	0.7110 ± 0.0028	0.5387 ± 0.0190	0.4004 ± 0.0043	0.6219 ± 0.0051	$\mathbf{22.24 \pm 0.68}$
+ RL (ours)	structural	EMA target	1213.5	$\mathbf{0.7620 \pm 0.0221}$	$\mathbf{0.5932 \pm 0.0520}$	$\mathbf{0.6490 \pm 0.0680}$	$\mathbf{0.8160 \pm 0.0300}$	22.68 ± 6.80
DMD2 (Yin et al., 2024a)	–	score net + disc.	1365.9	0.7791 ± 0.0039	0.6162 ± 0.0058	0.5349 ± 0.0016	0.7881 ± 0.0034	25.67 ± 0.45
+ RL (ours)	loss	score net + disc.	1420.3	$\mathbf{0.7982 \pm 0.0126}$	$\mathbf{0.6915 \pm 0.0109}$	$\mathbf{0.7672 \pm 0.0171}$	$\mathbf{0.8742 \pm 0.0051}$	$\mathbf{21.32 \pm 0.86}$

Density and Coverage but leaves FID unchanged within noise, with a seed spread ten times larger (Section 4.4).

4.4 Discussion and Limitations

ReDiF produces high-quality students at low training cost: on a single GPU it reaches quality the retrained baselines do not, while incorporating preferences through the same terminal objective and with no extra machinery. Appendix A compares it with existing methods.

Why the terminal reward may not help consistency distillation. Table 2 shows no FID improvement and increased seed variance for consistency distillation, unlike progressive distillation and DMD2. One explanation is that it constrains intermediate states to the teacher's probability-flow trajectory more tightly than the other two, leaving little room for terminal-reward exploration; ordering the three by that tightness matches the ordering of the gains. We have not tested the mechanism directly, so we offer it as a hypothesis rather than a result.

Diversity. ReDiF achieves the highest Recall, Density, and Coverage on ImageNet-64 and the highest Coverage on COCO. Since all samples use $\eta = 0$, these metrics measure diversity across noise and conditioning rather than policy stochasticity; we report the observed effect without attributing it to a specific mechanism.

Limitations. The central limitation is reward hacking. Policy optimization reaches a useful local optimum quickly, which is what enables low-cost training, but the reward can keep improving after sample quality has plateaued: past the fixed budget, samples develop hacking artifacts (Appendix J) that limit the attainable FID. The problem is more pronounced at larger scales and resolutions, where convergence is faster and hacking emerges earlier and is harder to detect. Multi-encoder rewards mitigate it partly, by requiring samples to satisfy several objectives at once, but do not eliminate it. This limitation arises from the reward and optimization process rather than terminal supervision itself; more direct suppression could potentially raise the quality ceiling, as illustrated by DPOK (Fan et al., 2023) relative to DDPO (Black et al., 2024).

The advantage of ReDiF is also budget-dependent, and the method is most suitable for the few-step rather than one-step regime, where longer horizons provide greater room for policy exploration (Appendix M).

5 Conclusion

We introduced ReDiF, a reinforcement learning framework for diffusion step distillation that replaces step-wise supervision with a terminal reward on the final sample. By scoring the endpoint with frozen encoders it avoids step-wise targets, moving-target supervision and auxiliary-network training, and trains only a LoRA adapter, on one GPU, from at most a thousand conditioning signals.

In the low-training regime, ReDiF achieves lower FID and higher Coverage than baselines trained on the same teacher-sample budget, across both teachers, and it improves DMD2 when combined with its loss. These gains matter most in low-budget settings and remain limited by reward over-optimization. Since distillation is itself reward optimization, further preferences, including non-differentiable ones, are added as extra reward terms, whose fidelity cost is left to future work.

AI USE STATEMENT

We used generative AI tools throughout this work: as writing assistants for language editing and clarity of exposition, as a sounding board while developing the analysis, and as a coding assistant during implementation. The research idea, the method, the experimental design and the claims are our own. All AI-assisted text was reviewed and verified by the authors; every formal statement and proof was stated and checked by the authors independently of any tool output; all code was read, executed and validated by the authors; and all numbers quoted from prior work were verified against the original publications. We take responsibility for the final content of this work, including any errors.

ETHICS STATEMENT

This work introduces no new dataset and no human-subject data, and uses only publicly available pretrained models and standard benchmarks. The method distills an existing teacher and inherits its biases and failure modes; the distillation reward is an imitation objective whose optimum is the teacher itself, so on its own it neither amplifies nor corrects what the teacher already does. The preference term of Section 3.5 changes this: it lets a student depart from its teacher in a chosen direction. We demonstrate it only with stylistic properties (encoded size and a sharpness score, Appendix K). Fairness or debiasing criteria that cannot be written as a reconstruction loss could in principle be optimized the same way, but we have not tested this and do not claim it works; the same freedom that admits such objectives is what makes the reward exploitable (Appendix J), and a debiasing reward would be subject to the same over-optimization. The same mechanism could equally be used to install undesirable preferences. By lowering the compute needed to obtain a fast sampler, the method broadens access to accelerated generative models, which carries the general dual-use risks of cheap high-quality image synthesis.

REPRODUCIBILITY STATEMENT

The training procedure is given in Algorithm 1 (Appendix E) and all hyperparameters and hardware in Appendix H. The evaluation protocol, including sample counts and reference sets, is stated in Section 4.1 and Appendix P. Assumptions and proofs for every formal statement are in Appendix B. The preference rewards are defined exactly, with all thresholds, in Appendix K. For retrained baselines, Appendix O reports the learning rate of every method and where it comes from, and Appendix N marks every quoted number as author-disclosed or a third-party estimate. Source code, configuration files and the launch scripts for every experiment are available at `https://anonymous.4open.science/r/ReDiF-7BAF/`.

REFERENCES

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

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Bangalath, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Shang-Wen Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. In *Advances in Neural Information Processing Systems*, 2025. URL `https://arxiv.org/abs/2504.13181`.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=1vmSEVL19f`.

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 *Advances in Neural Information Processing Systems*, volume 36, 2023.

Weilun Feng, Chuanguang Yang, Zhulin An, Libo Huang, Boyu Diao, Fei Wang, and Yongjun Xu. Relational diffusion distillation for efficient image generation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 205–213, 2024.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. BOOT: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023. URL https://arxiv.org/abs/2306.05544.

Shashank Gupta, Chaitanya Ahuja, Tsung-Yu Lin, Sreya Dutta Roy, Harrie Oosterhuis, Maarten de Rijke, and Satya Narayan Shukla. A simple and effective reinforcement learning method for text-to-image diffusion fine-tuning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=i8WJhKn455.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 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.

Zhiwei Jia, Yuesong Nan, Huixi Zhao, and Gengdai Liu. Reward fine-tuning two-step diffusion models via learning differentiable latent-space surrogate reward. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12912–12922, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

Dohyun Kim, Sehwan Park, Geonhee Han, Seung Wook Kim, and Paul Hongsuck Seo. Random conditioning with distillation for data-efficient diffusion model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18607–18618, 2025.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ymjI8feDTD.

Jiachen Li, Weixi Feng, Wenhu Chen, and William Yang Wang. Reward guided latent consistency distillation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=z116TO4LDT.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755. Springer, 2014.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1k4yZbbDqX.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://openreview.net/forum?id=2uAaGwlP_V.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. URL https://arxiv.org/abs/2101.02388.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023a. URL https://arxiv.org/abs/2310.04378.

Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. LCM-LoRA: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023b. URL https://arxiv.org/abs/2311.05556.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pp. 7176–7185. PMLR, 2020.

Owen Oertell, Jonathan D Chang, Yiyi Zhang, Kianté Brantley, and Wen Sun. RL for consistency models: Faster reward guided text-to-image generation. In *Reinforcement Learning Conference (RLC)*, 2024. URL https://arxiv.org/abs/2404.03673.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, 2024.

Xiaohua Qi, Renda Li, Long Peng, Qiang Ling, Jun Yu, Ziyi Chen, Peng Chang, Mei Han, and Jing Xiao. Data-free knowledge distillation with diffusion models. In *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2025. doi: 10.1109/ICME59968.2025.11209042.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 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.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024.

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

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:4263–4276, 2023.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

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*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Ngoc Duy Tran, Kien Do, Duc Thanh Nguyen, and Truyen Tran. Progressive multistep data-free diffusion distillation. OpenReview submission, 2025. URL https://openreview.net/forum?id=MbFWSPWsJl.

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

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Qianlong Xiang, Miao Zhang, Yuzhang Shang, Jianlong Wu, Yan Yan, and Liqiang Nie. DKDM: Data-free knowledge distillation for diffusion models with any architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2955–2965, 2025.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in Neural Information Processing Systems*, 37:47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024b.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Yuan Zhang, Chenyi Li, Guoqing Ma, Jiajun Zha, Yuanming Yang, Bo Wang, Wei Tang, Wenbo Li, Haoyang Huang, and Nan Duan. Teacher-feature drifting: One-step diffusion distillation with pretrained diffusion representations. *arXiv preprint arXiv:2605.07327*, 2026. URL https://arxiv.org/abs/2605.07327.

Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *International Conference on Machine Learning*, pp. 42390–42402. PMLR, 2023a.

Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=HDxgaKk956l.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *International Conference on Machine Learning*. PMLR, 2024a. URL https://arxiv.org/abs/2404.04057.

Mingyuan Zhou, Zhendong Wang, Huangjie Zheng, and Hai Huang. Guided score identity distillation for data-free one-step text-to-image generation. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2406.01561.

Method	Step Distill.	One-Step	Data-Free	RL	Few Pr.	Non-Diff.	Low-Comp.
Trajectory matching							
Progressive Distill. (Salimans & Ho, 2022)	✓	✓	✗	✗	✗	✗	✗
Guided Distill. (Meng et al., 2023)	✓	✓	✗	✗	✗	✗	✗
Consistency (Song et al., 2023)	✓	✓	✗	✗	✗	✗	✗
LCM / LCM-LoRA (Luo et al., 2023a;b)	✓	✓	✗	✗	✗	✗	✗
CTM (Kim et al., 2024)	✓	✓	✗	✗	✗	✗	✗
InstaFlow (Liu et al., 2024)	✓	✓	✗	✗	✗	✗	✗
Distribution matching, adversarial and drifting							
DMD / DMD2 (Yin et al., 2024b;a)	✓	✓	✗	✗	✗	✗	✗
SiD (Zhou et al., 2024a)	✓	✓	✓	✗	✗	✗	✗
SiDA (Zhou et al., 2025b)	✓	✓	✗	✗	✗	✗	✗
ADD / SD-Turbo (Sauer et al., 2024)	✓	✓	✗	✗	✗	✗	✗
Guided SiD (Zhou et al., 2025a)	✓	✓	✓	✗	✗	✗	✗
TFD (Zhang et al., 2026)	✓	✓	✗	✗	✗	✗	✗
Data-free trajectory methods							
BOOT (Gu et al., 2023)	✓	✓	✓	✗	✗	✗	✗
DKDM (Xiang et al., 2025)	✓	–	✓	✗	✗	✗	✗
PMDD (Tran et al., 2025)	✓	–	✓	✗	✗	✗	✗
Reward-based							
DDPO (Black et al., 2024)	✗	–	✓	✓	✓	✓	✓
RLCM (Oertell et al., 2024)	✗	–	✓	✓	✓	✓	✓
DRaFT (Clark et al., 2024)	✗	–	✓	✗	✓	✗	✓
RG-LCD (Li et al., 2024)	✓	–	✗	✗	✗	✗	✗
ReDiF (ours)	✓	✗	✓	✓	✓	✓	✓

Table 3: Comparison of diffusion acceleration and distillation methods. Data-Free indicates that no real images are used for training: SiD and Guided SiD qualify but retain an auxiliary score network; SiDA and ADD do not, because their discriminators are trained against real data; TFD does not, because its drifting field is estimated against a real-data reference set where available; RG-LCD does not, because its consistency loss is computed on real image–text data. The reward-based alignment methods use prompts only. ReDiF uses real images only for evaluation and to fix the training length once (Appendix H). One-Step indicates that the source paper demonstrates a single-step student; – marks methods for which it is not reported or not applicable, and ReDiF is marked ✗ because it targets the few-step rather than the one-step regime (Appendix M). Low-Comp. indicates methods that avoid heavy supervised reconstruction and large-scale synthetic datasets.

Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=lS2SGfWizd.

Zhenyu Zhou, Defang Chen, Can Wang, Chun Chen, and Siwei Lyu. Simple and fast distillation of diffusion models. *Advances in Neural Information Processing Systems*, 37:40831–40860, 2024b.

A Method Comparison Table

Table 3 places ReDiF against the families reviewed in Section 2 on seven axes: whether the method performs step distillation, whether it has been demonstrated at a single step, whether it uses no real training images, whether it optimizes by reinforcement learning, whether it trains from few conditioning signals, whether it admits non-differentiable rewards, and whether it avoids heavy supervised reconstruction and large-scale synthetic datasets.

B Supporting Analysis

This appendix collects the formal statements used in Section 3. We keep only statements that are either standard or proved in full here; the remaining arguments of Section 3.1 are motivation and are tested empirically in Section 4. Throughout, $\{t_k\}_{k=0}^{M}$ is the student schedule, $\Phi_{\theta,k}$ the student map from x_{t_k} to $x_{t_{k-1}}$, Φ_k^* the teacher run over the same interval with its own finer steps, and x_0^θ, x_0^* the terminal samples from a shared (x_T, c).

B.1 Error Amplification along the Trajectory

Lemma 1 (Error amplification). *Suppose each teacher map Φ_k^* is L_k-Lipschitz and the student deviates from it by at most $\varepsilon_k = \sup_x \|\Phi_{\theta,k}(x) - \Phi_k^*(x)\|$ over the states it visits. Then*

$$\|x_0^\theta - x_0^*\| \le \sum_{k=1}^{M} \varepsilon_k \prod_{j<k} L_j \le \Gamma \max_k \varepsilon_k, \qquad \Gamma := \sum_{k=1}^{M} \prod_{j<k} L_j.$$

If $L_j \ge 1$ for all j then $\Gamma \ge M$, with equality exactly when $L_j = 1$ for $j < M$; if $L_j \ge L > 1$ for all j then $\Gamma \ge (L^M - 1)/(L - 1)$.

Proof. Let $e_k = \|x_{t_k}^\theta - x_{t_k}^*\|$, with $e_M = 0$ since both chains start from the same x_T. Adding and subtracting the teacher map applied to the student's state,

$$x_{t_{k-1}}^\theta - x_{t_{k-1}}^* = \big[\Phi_{\theta,k}(x_{t_k}^\theta) - \Phi_k^*(x_{t_k}^\theta)\big] + \big[\Phi_k^*(x_{t_k}^\theta) - \Phi_k^*(x_{t_k}^*)\big],$$

where the first bracket is bounded by ε_k and the second by $L_k e_k$. Hence $e_{k-1} \le L_k e_k + \varepsilon_k$, and unrolling from $e_M = 0$ gives $e_0 \le \varepsilon_1 + L_1\varepsilon_2 + L_1L_2\varepsilon_3 + \cdots = \sum_k \varepsilon_k \prod_{j<k} L_j$. Each summand of Γ is a product of Lipschitz constants, the $k = 1$ term being the empty product 1. If all $L_j \ge 1$ every summand is at least 1, so $\Gamma \ge M$, with equality exactly when every product equals 1. If $L_j \ge L > 1$, the summands are at least L^{k-1} and the geometric sum gives the last bound. □

The bound is an upper bound, so $\varepsilon_k \le \delta/\Gamma$ is *sufficient* for terminal accuracy δ, not necessary. Its use in Section 3.1 is accordingly limited: it shows what per-step control guarantees, and why guaranteeing terminal accuracy through per-step losses is demanding when Γ is large. If the fitted per-step error decreases as $v/\sqrt{n}$ with n samples, this sufficient condition is met once $n \gtrsim \Gamma^2 v^2/\delta^2$; the rate is an assumption about the fitting procedure, not a result.

B.2 Moving Targets

Remark 1. Suppose the target is available only through an estimate $\hat{y}_k$ with systematic error b_k that does not decrease with the number of samples drawn from it. Then the per-step deviation cannot fall below b_k, and the guarantee of Lemma 1 cannot fall below $\Gamma \max_k b_k$ at any sample budget. For reconstruction and progressive targets, which are deterministic functions of a frozen teacher, $b_k = 0$. For an EMA target, which lags the student by construction, and for a fake-score network fitted to a generator that changes at every update, the systematic component is refreshed as training proceeds rather than averaged away. This is an informal argument about the guarantee, not a lower bound on the error of any specific method.

B.3 Terminal Credit Assignment

Proposition 1 (Policy gradient with a terminal reward). *For the Gaussian policy of Section 3.2, a bounded terminal reward $R(x_0)$ and any baseline b that does not depend on the actions of the trajectory,*

$$\nabla_\theta J(\theta) = \mathbb{E}\left[(R - b)\sum_{k=1}^{M} \nabla_\theta \log \pi_\theta\big(x_{t_{k-1}} \mid x_{t_k}, t_k, c\big)\right].$$

Proof. This is the likelihood-ratio (REINFORCE) gradient (Williams, 1992) specialized to a terminal reward. The trajectory density given (x_T, c) is $p_\theta(\tau) = \prod_{k=1}^{M} \pi_\theta(x_{t_{k-1}} \mid x_{t_k}, t_k, c)$, and the dynamics contribute no other θ-dependent factor, so $\nabla_\theta \log p_\theta(\tau) = \sum_k \nabla_\theta \log \pi_\theta(x_{t_{k-1}} \mid x_{t_k}, t_k, c)$. The score-function identity $\nabla_\theta J = \mathbb{E}[R\, \nabla_\theta \log p_\theta(\tau)]$ follows by exchanging differentiation and integration, which is justified by dominated convergence since R is bounded and the Gaussian score is integrable. The reward is terminal and multiplies the whole sum. A baseline independent of the actions leaves the expectation unchanged because $\mathbb{E}[\nabla_\theta \log p_\theta(\tau)] = 0$. Nothing in the argument uses differentiability of R, which is what permits the piecewise-constant rewards of Appendix K. □

The batch-standardized advantage of Section 3.4 satisfies this condition only approximately: $\mu_{\mathcal{B}}$ is computed from the same rollouts it is subtracted from, so it is not independent of the actions, division by $\sigma_{\mathcal{B}}$ rescales the gradient by a quantity that varies from iteration to iteration, and the clip at $A_{\max}$ truncates the largest deviations. A leave-one-out mean and an unnormalized advantage would remove the first two effects; we follow DDPO and use the batch statistics directly, accepting a small bias in exchange for variance reduction.

B.4 Invariance of the Semantic Reward

Proposition 2 (First-order invariance)**.** *Let $f : \mathbb{R}^D \to \mathbb{R}^d$, $d < D$, be continuously differentiable and let the reward be $R(I) = \mathcal{S}(z_T, f(I))$ for a fixed z_T and a C^1 similarity $\mathcal{S}$. Then for every $v \in \ker Df(I)$, $R(I + \epsilon v) = R(I) + o(\epsilon)$, and $\dim \ker Df(I) \geq D - d$. For a sum of rewards built on encoders $f_1, \ldots, f_K$, the same holds for every $v \in \bigcap_i \ker Df_i(I)$.*

Proof. By the chain rule $\nabla_I R = Df(I)^\top \nabla_z \mathcal{S}(z_T, f(I))$, which lies in the row space of $Df(I)$ and is therefore orthogonal to $\ker Df(I)$. The directional derivative of R along any $v \in \ker Df(I)$ is thus zero, which gives the expansion. Rank–nullity gives $\dim \ker Df(I) = D - \operatorname{rank} Df(I) \geq D - d$. For a sum of rewards the gradient is the sum of the individual gradients, each orthogonal to its own kernel, so their sum is orthogonal to the intersection. □

The statement concerns the reward, not the optimizer. It does not imply that the policy-gradient estimator has lower variance, which depends on the dimension of the action space rather than on that of the embedding. It is used in Section 3.3 and Appendix J only to explain which image changes the reward does not see, and why the reward can keep increasing while image-level metrics do not.

B.5 Scope

These statements are either standard (Proposition 1) or elementary (Lemma 1, Proposition 2), and Remark 1 is informal. None is a convergence result for policy optimization, and none establishes that ReDiF requires fewer updates or smaller batches than other methods, or bounds how far the optimal student lies from the teacher; those are empirical claims, tested in Section 4 and Appendices D, I.1 and M.

C Cost Model

Every row of Table 1 is trained on the same teacher, on the same sampler grid, at the same student step count, from the same conditioning pool, for the same number of epochs over that pool, on the same A100. An epoch is 32 conditioning signals on ImageNet-64 and 16 on COCO, so the 100- and 20-epoch budgets are 3,200 and 320 training samples drawn from pools of 1,000 (noise, class) pairs and 160 prompts. Training compute follows from those choices; this section states what the PFLOPs column of Table 1 counts.

Pool synthesis. Training without real images begins by turning the conditioning pool into teacher samples. A trajectory is N_T teacher network evaluations, $N_T = 2N - 1 = 511$ on ImageNet-64 (Heun, $N = 256$) and $N_T = 2N = 100$ on COCO ($N = 50$, conditional and unconditional branch), and it carries every grid point it passes: one pass over the pool supplies both the terminal references ReDiF scores against and the trajectory targets progressive and consistency distillation regress on. The pool is written once, before training, and read by every row that uses it. Because N_T is two orders of magnitude larger than the student step count M, this term dominates the totals in Table 1, and it is the same term for every row.

Training cost. Let C_S and C_T be the forward cost of one student and one teacher evaluation, C_{enc} the summed cost of the reward encoders over one image and C_{dec} of one VAE decode; FLOPs are counted as 2×MACs and backward cost as twice forward. Over I iterations with G rollouts of M steps, E inner epochs and K encoders, training costs

$$C_{\mathrm{train}}^{\mathrm{ReDiF}} = I\Big[\underbrace{GMC_S}_{\text{rollout}} + \underbrace{3EGMC_S}_{\text{surrogate}} + \underbrace{\mathbb{1}[\beta > 0]\, EGMC_T}_{\text{KL}} + \underbrace{GKC_{\mathrm{enc}}}_{\text{reward}} + \underbrace{\mathbb{1}[\text{latent}]\, GC_{\mathrm{dec}}}_{\text{decode}}\Big],$$

with the student terms doubled under guidance (Section 3.4). The KL term is the only teacher call ReDiF makes during training and is absent on COCO, where $\beta = 0$. The objectives differ in which of these terms they carry. Progressive and consistency distillation replace the reward and surrogate terms by a regression loss on stored teacher or EMA targets and add nothing else. A distribution-matching iteration evaluates the teacher score and trains an auxiliary score network, and in DMD2 and SiDA a discriminator, on samples drawn from the student during training; those teacher calls are part of its training cost and are not served by the pool.

What the table reports. The PFLOPs column of Table 1 is the analytical cost of reproducing a row from scratch: pool synthesis plus that row's training cost. It is dominated by the pool, which is shared, so the column should be read as the cost of obtaining a student on one device rather than as a measure of how much optimization each objective performs. The GPU-h column is the measured counterpart, wall-clock on the same device for the same runs.

D Quality as a Function of Training Budget

A single matched budget invites the objection that it is the one that favors us. This appendix therefore reports quality along the budget axis rather than at one operating point.

D.1 FID versus Training Epochs on ImageNet-64

Figure 3 plots FID against training epoch for ReDiF and the four strongest baselines of Table 1 on the EDM teacher ($256 \rightarrow 8$). All five students share the teacher, the student step count, the checkpoint grid and the evaluation protocol of Appendix P. DMD and SiD are omitted for legibility; they track DMD2 and SiDA at a uniformly higher FID.

ReDiF leads across the whole range, and by the widest margin at low budgets. ReDiF is below every baseline at every evaluated epoch. It separates first: by epoch 60 it is near 27 FID while consistency distillation, the best baseline, is near 28.5 and the other three lie between 31 and 35, and by epoch 100 it reaches roughly 18.5 while the best baseline is near 21. After that ReDiF is essentially flat, moving by less than 1.5 FID over the remaining 100 epochs, while the baselines keep descending; none of them reaches it. At epoch 200 ReDiF is near 18 and the closest three, SiDA, consistency distillation and DMD2, are between 20.8 and 21.3. The margin of Table 1 is therefore widest in the low-training regime, where the baselines have not yet recovered from their initialization, but it does not close: SiDA and progressive distillation are still descending at the right edge of the plot, so we report the ordering over the measured range and do not extrapolate past it.

An epoch is the same amount of data for every curve. Every curve is trained on the same teacher, the same conditioning pool and the same number of samples per epoch, so the horizontal axis carries the same meaning for all of them. The objectives do not cost the same per epoch: the regression losses of progressive and consistency distillation are the cheapest, and the distribution-matching rows train an auxiliary network alongside the student (Appendix C). The ordering above is therefore one at equal data, not at equal compute per epoch.

Two baselines are non-monotone. SiDA and progressive distillation rise above their initialization before improving, peaking near epochs 80 and 100 respectively at FID above 35. A run of either method stopped inside that window returns a checkpoint no better than its initialization. Their entries in Table 1 should therefore be read as what these objectives deliver at this budget; their results at their own budgets are in Table 8.

D.2 The SD v1.5 Teacher

We did not run the equivalent sweep on the SD v1.5 teacher and make no quantitative claim about the shape of its budget curve. What we did observe is that runs continued past the fixed budget show the reward-hacking artifacts of Figure 4, so we expect the advantage in Table 1(b) to shrink or reverse at larger budgets. This is the reason we do not present ReDiF as a drop-in replacement for distribution-matching distillation at large budgets.

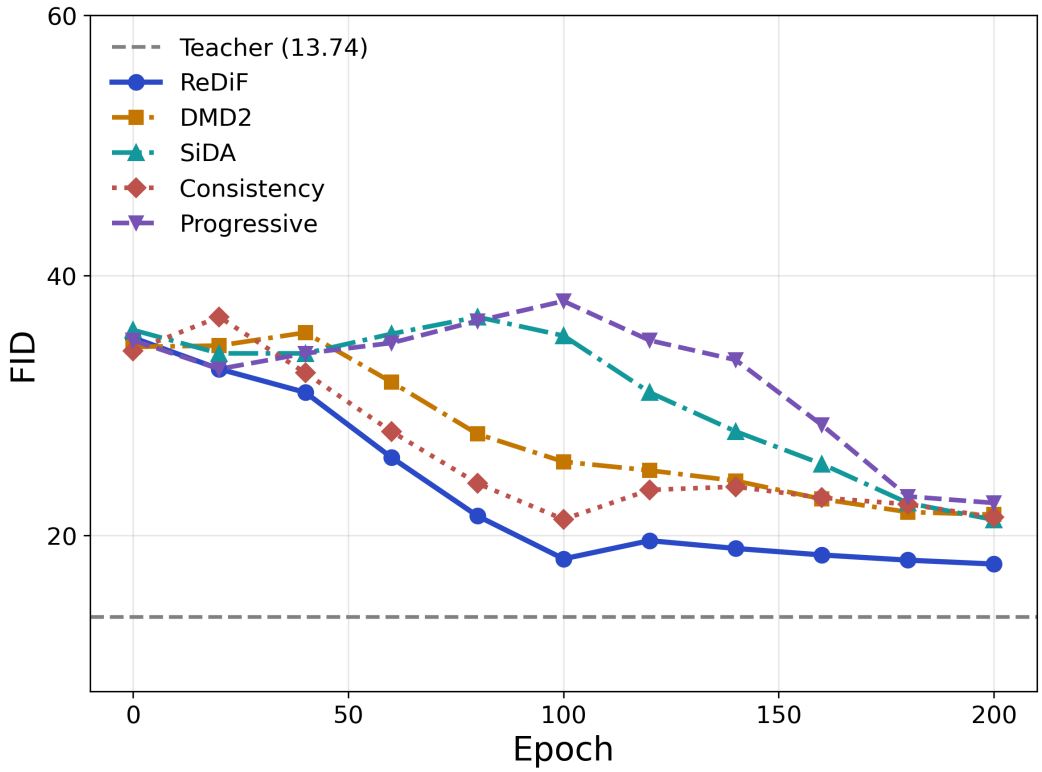


Figure 3: FID versus training epoch on ImageNet-64 ($256 \to 8$, EDM teacher). Markers denote evaluated checkpoints; the dashed line is the 256-step teacher (FID $= 13.74$) under the same protocol. The horizontal axis counts epochs; because the rest of the protocol is held fixed, an epoch is the same number of training samples for every curve (Appendix C).

E REDIF ALGORITHM

Algorithm 1 summarizes the pipeline. The teacher is used only to draw a terminal reference image for each rollout and, when $\beta > 0$, to evaluate the KL term; it never supervises intermediate states, and its outputs are never stored. The student receives supervision only after producing its final sample, and the reward, optionally augmented with the preference term of equation 1, is propagated to all of its denoising actions by policy optimization.

F QUALITATIVE TRAINING PROGRESSION

Figures 1 and 4 follow one SD v1.5 student's output for a fixed conditioning and seed across training, with the teacher's reference on the right; holding the generation conditions fixed isolates the effect of the policy updates. Figure 1 covers the 20-epoch budget of the reported COCO runs: although the only supervision is a scalar computed on the final image, detail and agreement with the teacher improve within a few epochs. Figure 4 continues the same run to 50 epochs. The sample stays stable for a few epochs past the budget and then degrades, with structured artifacts appearing by epoch 35 that the reward does not penalize; this is the over-optimization analyzed in Appendix J.

G QUALITATIVE SAMPLES, SD V1.5

Figures 5 and 6 place teacher samples above student samples on the COCO branch. The teacher runs 50 steps; the student runs 5, trained with the R_2 reward and taken at the last checkpoint of the fixed budget. Prompts are held out from the training pool. Columns are not seed-matched, so the comparison is of output quality rather than of the same latent rendered twice.

Algorithm 1 ReDiF: reinforcement learning based diffusion step distillation

Require: Frozen teacher G_T; student G_S initialized from G_T; conditioning set $\mathcal{D}$; encoders $\{f_i\}_{i=1}^K$; weights $\{\lambda_i\}_{i=1}^K$; optional preference reward R_{pref} with weight γ ($\gamma = 0$ by default); teacher steps N; student steps M; rollouts per iteration B.

Ensure: Distilled student G_S.

1: **for** each optimization iteration **do**
2: Sample a batch $\mathcal{B}$ of B conditionings from $\mathcal{D}$.
3: For each $(x_T, c) \in \mathcal{B}$, obtain a reference $I_T \leftarrow G_T(x_T, c, N)$ from the teacher's sampler and embed $z_T^{(i)} \leftarrow f_i(I_T)$; no dataset of teacher trajectories is assembled.
4: Roll out one student trajectory per conditioning, $I_S \leftarrow G_S(x_T, c, M)$, storing the sampled latents and their log-densities.
5: Compute $z_S^{(i)} \leftarrow f_i(I_S)$ and $R \leftarrow \sum_{i=1}^K \lambda_i \mathcal{S}_i(z_T^{(i)}, z_S^{(i)}) + \gamma R_{\text{pref}}(I_S)$.
6: Standardize over the batch: $\hat{A} = \text{clip}\left((R - \mu_{\mathcal{B}})/(\sigma_{\mathcal{B}} + \varepsilon), -A_{\max}, A_{\max}\right)$.
7: For E inner epochs, update the student by maximizing $\mathcal{L} = \mathcal{L}_{\text{PPO}} - \beta D_{\text{KL}}(\pi_\theta \| \pi_T)$.
8: **end for**
9: **return** G_S.



Figure 4: The SD v1.5 student of Figure 1 trained for 50 epochs, well past the 20-epoch budget of the reported COCO runs, for the same conditioning and seed; the rightmost column is the teacher's reference. Quality stays stable for a few epochs and then degrades as the student exploits directions the reward does not penalize.

H Implementation Details

All experiments run on a single NVIDIA A100 (80 GB). The PPO hyperparameters of Table 4 follow the Stable Diffusion configuration of DDPO (Black et al., 2024), which optimizes the same clipped surrogate on the same backbone; we did not search them. The effective batch is 32 on ImageNet-64 and 8 on COCO, in both cases without gradient accumulation. The student is a LoRA adapter of rank 4 on the teacher and is optimized with PPO; the teacher and the reward encoders stay frozen. Training uses fewer than one thousand conditioning signals, resampled throughout optimization. No dataset of teacher trajectories is built: references come from the teacher's own sampler inside the loop, and student rollouts are kept only for the inner epochs of the iteration that produced them. When several encoders are used the reward is their equally weighted sum. All reported metrics use the same inference settings and random seeds across compared methods.

Held-out monitoring and training length. Because the reward keeps improving after sample quality saturates (Appendix J), the reward cannot be used to decide when to stop. We monitor FID on a small held-out set generated from conditioning excluded from training and from evaluation, and use it once to fix the training length of each branch. All reported numbers, for ReDiF and for every baseline, are the last checkpoint of that fixed budget.

The remaining hyperparameters are listed in Table 4.

I Reward Ablation

Section 3.3 motivates computing the terminal reward in a representation space rather than in pixels. This appendix tests that choice and selects the two configurations carried into Section 4 as R_1 and R_2. Every configuration is optimized by the identical PPO procedure, from the same initialization, with the same conditioning pool, batch, budget and seeds, so the reward is the only difference between

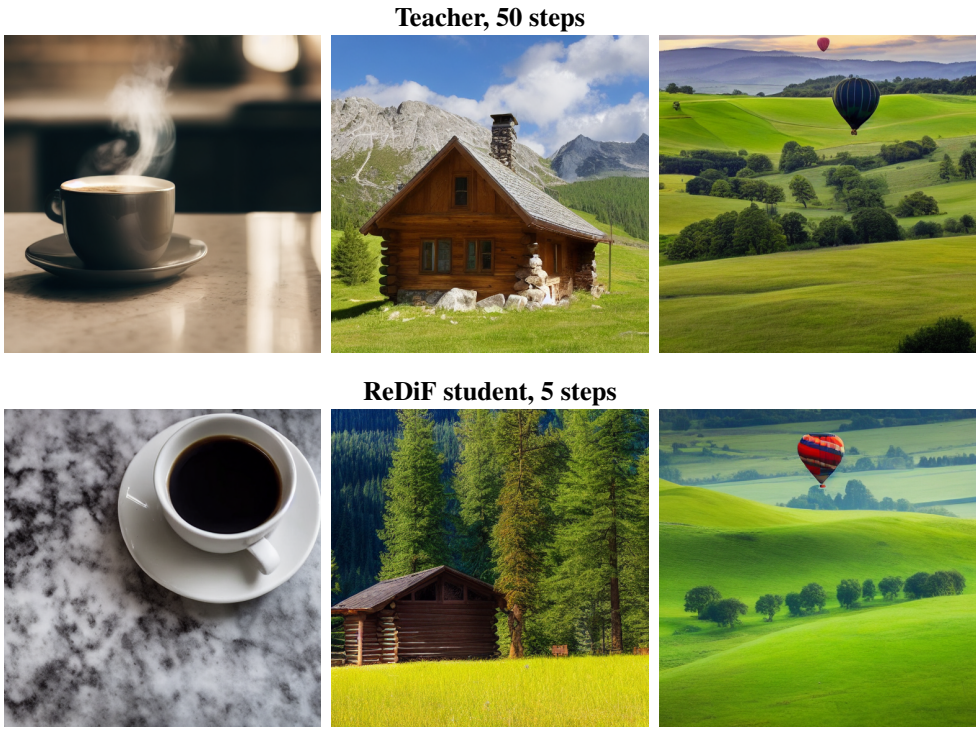


Figure 5: Teacher and student samples, SD v1.5, first three held-out prompts.

Table 4: Hyperparameters used in all reported ReDiF runs. Both branches use AdamW with β_1=0.9, β_2=0.999, weight decay 10^{-4}, ϵ=10^{-8} and gradient-norm clipping at 1.0. The preference runs of Appendix K use the COCO column with the preference weight added.

Hyperparameter	**EDM / ImageNet-64**	**SD v1.5 / COCO**
Optimizer	AdamW	AdamW
Trainable parameters	LoRA, rank / alpha 4 / 4	LoRA, rank / alpha 4 / 4
LoRA target modules	UNet attention (q, k, v, out)	UNet attention (q, k, v, out)
Learning rate	3×10^{-4}	3×10^{-4}
PPO clip range ϵ	10^{-4}	10^{-4}
Inner epochs per rollout	4	4
Rollouts per iteration	32	16
Advantage clip $A_{\max}$	5	5
Effective batch (no accumulation)	32	8
Training epochs	100	20
Rollouts	3200	320
KL weight β	0.02	0
Guidance scale (teacher / student)	–	5.0 / 5.0
Preference weight γ (Appendix K)	–	2
Conditioning pool	1,000 (noise, class) pairs	160 COCO prompts
Reported checkpoint	last	last

rows. We group rewards into single encoders, combinations of encoders, and two non-semantic controls: the negative pixel MSE between teacher and student images from the same noise, and a divergence-based reward. Tables 5 and 6 report the two branches separately. The ImageNet branch has no caption to align against, so it reports CLIP image-to-image similarity to the teacher sample (CLIP-T) in place of CLIPScore.

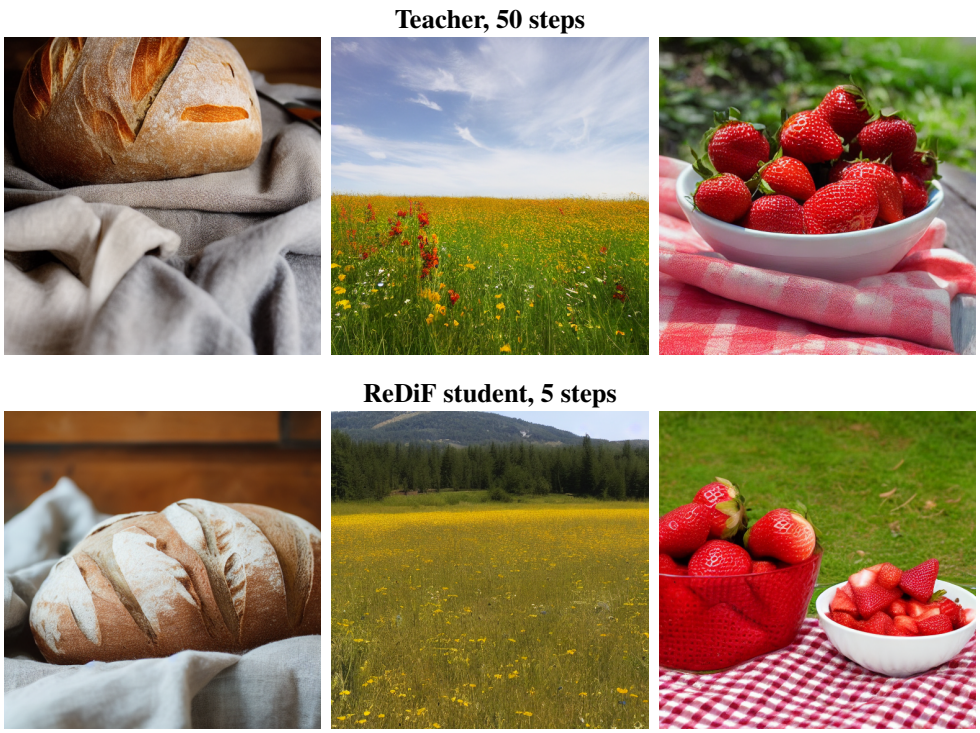


Figure 6: Teacher and student samples, SD v1.5, remaining three held-out prompts.

Table 5: **Reward ablation, EDM teacher on ImageNet-64.** Identical PPO settings, initialization, conditioning pool, budget and seeds across rows; only the reward differs. Last checkpoint, $M = 8$, $\eta = 0$, 10,000 generated samples against 10,000 real validation images; the R_1 and R_2 rows are the ReDiF rows of Table 1. Mean $\pm$ sample sd over seeds. CLIP-T is the CLIP image-to-image similarity to the teacher sample; for rewards containing a CLIP term it is part of the training objective and not an independent measure. Best entry in bold, second best bold and dash-underlined; teacher rows are not ranked.

Reward	P ↑	R ↑	D ↑	C ↑	CLIP-T ↑	FID ↓
Teacher (256 steps, reference)	0.8310	0.7103	0.8930	0.7831	–	13.74
Teacher (8 steps)	0.5308	0.4780	0.3910	0.5326	–	35.04
Single-encoder semantic rewards						
CLIP (R_1)	0.7529 ± 0.0180	$\mathbf{0.6265 \pm 0.0213}$	0.6318 ± 0.0291	0.8223 ± 0.0169	$\underline{\mathbf{0.8836 \pm 0.0028}}$	$\underline{\mathbf{18.84 \pm 2.01}}$
DINOv3	0.7198 ± 0.0161	0.5654 ± 0.0367	0.5972 ± 0.0418	0.7828 ± 0.0191	0.8675 ± 0.0058	33.11 ± 9.95
PE	0.6992 ± 0.0152	0.5685 ± 0.0182	0.5323 ± 0.0247	0.7452 ± 0.0226	0.8522 ± 0.0054	25.67 ± 3.62
LPIPS	0.7058 ± 0.0091	0.5924 ± 0.0125	0.5409 ± 0.0232	0.7542 ± 0.0206	0.8728 ± 0.0036	21.11 ± 1.55
Combined semantic rewards						
CLIP + DINOv3	0.7456 ± 0.0051	0.5861 ± 0.0418	0.5947 ± 0.0031	0.7792 ± 0.0177	0.8658 ± 0.0060	20.12 ± 1.91
CLIP + PE	$\mathbf{0.7841 \pm 0.0229}$	0.6152 ± 0.0141	0.6638 ± 0.0402	0.8098 ± 0.0127	0.8789 ± 0.0045	20.75 ± 0.12
CLIP + LPIPS	0.6862 ± 0.0082	0.5364 ± 0.0144	0.5333 ± 0.0118	0.7419 ± 0.0101	0.8385 ± 0.0080	23.21 ± 0.79
DINOv3 + PE	$\underline{\mathbf{0.7819 \pm 0.0051}}$	0.5271 ± 0.0003	$\underline{\mathbf{0.7617 \pm 0.0073}}$	$\mathbf{0.8381 \pm 0.0019}$	0.8586 ± 0.0024	31.92 ± 1.75
DINOv3 + LPIPS	0.7342 ± 0.0036	0.5926 ± 0.0084	0.6202 ± 0.0139	0.7798 ± 0.0002	0.8641 ± 0.0082	20.53 ± 0.63
PE + LPIPS	0.7499 ± 0.0021	0.5911 ± 0.0059	0.6159 ± 0.0031	0.7937 ± 0.0043	0.8723 ± 0.0005	22.14 ± 0.31
CLIP + DINOv3 + PE	0.7281 ± 0.0149	0.6109 ± 0.0061	0.5723 ± 0.0017	0.7837 ± 0.0063	0.8738 ± 0.0067	19.42 ± 2.05
CLIP + DINOv3 + PE + LPIPS (R_2)	0.7432 ± 0.0084	$\underline{\mathbf{0.6221 \pm 0.0158}}$	$\mathbf{0.7991 \pm 0.0101}$	0.8022 ± 0.0129	0.8793 ± 0.0030	$\mathbf{18.60 \pm 1.63}$
Non-semantic controls						
Negative MSE (pixel)	0.7618 ± 0.0091	0.6018 ± 0.0131	0.6702 ± 0.0090	$\underline{\mathbf{0.8268 \pm 0.0063}}$	0.8801 ± 0.0015	20.55 ± 3.05
Negative KL (divergence)	0.7439 ± 0.0111	0.6151 ± 0.0334	0.6628 ± 0.0042	0.8191 ± 0.0127	$\mathbf{0.8863 \pm 0.0032}$	23.01 ± 2.35

Findings. The four-encoder reward R_2 has the lowest FID on both branches, and CLIP alone (R_1) is second on ImageNet-64 and within about one FID of the best on COCO at the lowest evaluation cost; these are the two configurations carried forward. The pixel control behaves differently on the two branches. On COCO both non-semantic controls are worse in FID than every semantic configuration (29.80 and 30.13 against 26.36–27.99). On ImageNet-64 the pixel MSE is competitive (20.55): it is beaten by R_1, R_2, CLIP + DINOv3 + PE, CLIP + DINOv3 and, marginally, DINOv3 +

Table 6: **Reward ablation, Stable Diffusion v1.5 on COCO** ($50 \to 5$)**.** Same design as Table 5: identical optimization, only the reward differs. Last checkpoint, $M = 5$, 5,000 COCO validation prompts; the R_1 and R_2 rows are the ReDiF rows of Table 1. Mean $\pm$ sample sd over seeds. Best entry bold in bold, second best bold and dash-underlined; teacher rows are not ranked.

Reward	P ↑	R ↑	D ↑	C ↑	FID ↓	CLIPScore ↑
Teacher (50 steps, reference)	0.6103	0.2029	0.8878	0.6832	24.59	0.3137
Teacher (5 steps)	0.4420	0.1762	0.6291	0.5813	31.45	0.3041
Single-encoder semantic rewards						
CLIP (R_1)	$\mathbf{0.4890 \pm 0.0062}$	0.1864 ± 0.0047	$\mathbf{0.7411 \pm 0.0087}$	$\mathbf{0.6682 \pm 0.0065}$	27.42 ± 0.84	0.3038 ± 0.0007
DINOv3	0.4310 ± 0.0084	0.1809 ± 0.0061	0.6406 ± 0.0112	0.5903 ± 0.0075	27.40 ± 0.91	0.3046 ± 0.0004
PE	0.4498 ± 0.0072	0.1770 ± 0.0057	0.6733 ± 0.0104	0.5927 ± 0.0069	27.97 ± 0.76	0.3044 ± 0.0001
LPIPS	0.4616 ± 0.0059	0.1770 ± 0.0049	0.6842 ± 0.0091	0.5813 ± 0.0062	27.64 ± 0.69	$\mathbf{0.3053 \pm 0.0008}$
Combined semantic rewards						
CLIP + DINOv3	0.4550 ± 0.0068	0.1813 ± 0.0053	0.6512 ± 0.0101	0.5965 ± 0.0071	27.61 ± 0.82	0.3042 ± 0.0002
CLIP + PE	0.4497 ± 0.0075	0.1820 ± 0.0058	0.6420 ± 0.0098	0.6043 ± 0.0065	27.72 ± 0.73	0.3043 ± 0.0005
CLIP + LPIPS	0.4410 ± 0.0062	0.1714 ± 0.0051	0.6402 ± 0.0107	0.5939 ± 0.0070	27.48 ± 0.68	0.3037 ± 0.0007
DINOv3 + PE	0.4150 ± 0.0071	$\mathbf{0.1865 \pm 0.0055}$	0.6368 ± 0.0115	0.5997 ± 0.0068	27.99 ± 0.85	0.3041 ± 0.0009
DINOv3 + LPIPS	0.4503 ± 0.0065	0.1719 ± 0.0052	0.6451 ± 0.0103	0.6126 ± 0.0074	27.44 ± 0.71	0.3043 ± 0.0003
PE + LPIPS	0.4743 ± 0.0069	0.1732 ± 0.0059	$\mathbf{0.7406 \pm 0.0108}$	0.5182 ± 0.0077	27.97 ± 0.79	0.3051 ± 0.0001
CLIP + DINOv3 + PE	0.4490 ± 0.0076	0.1770 ± 0.0056	0.7340 ± 0.0119	0.5910 ± 0.0068	$\mathbf{27.01 \pm 0.90}$	0.3035 ± 0.0002
CLIP + DINOv3 + PE + LPIPS (R_2)	$\mathbf{0.5193 \pm 0.0083}$	$\mathbf{0.1896 \pm 0.0054}$	0.6301 ± 0.0095	$\mathbf{0.6812 \pm 0.0056}$	$\mathbf{26.36 \pm 0.78}$	0.3043 ± 0.0003
Non-semantic controls						
Negative MSE (pixel)	0.4714 ± 0.0067	0.1690 ± 0.0052	0.6728 ± 0.0102	0.5838 ± 0.0064	29.80 ± 1.12	$\mathbf{0.3058 \pm 0.0003}$
Negative KL (divergence)	0.4329 ± 0.0069	0.1593 ± 0.0053	0.6632 ± 0.0117	0.5954 ± 0.0042	30.13 ± 1.07	0.3050 ± 0.0002

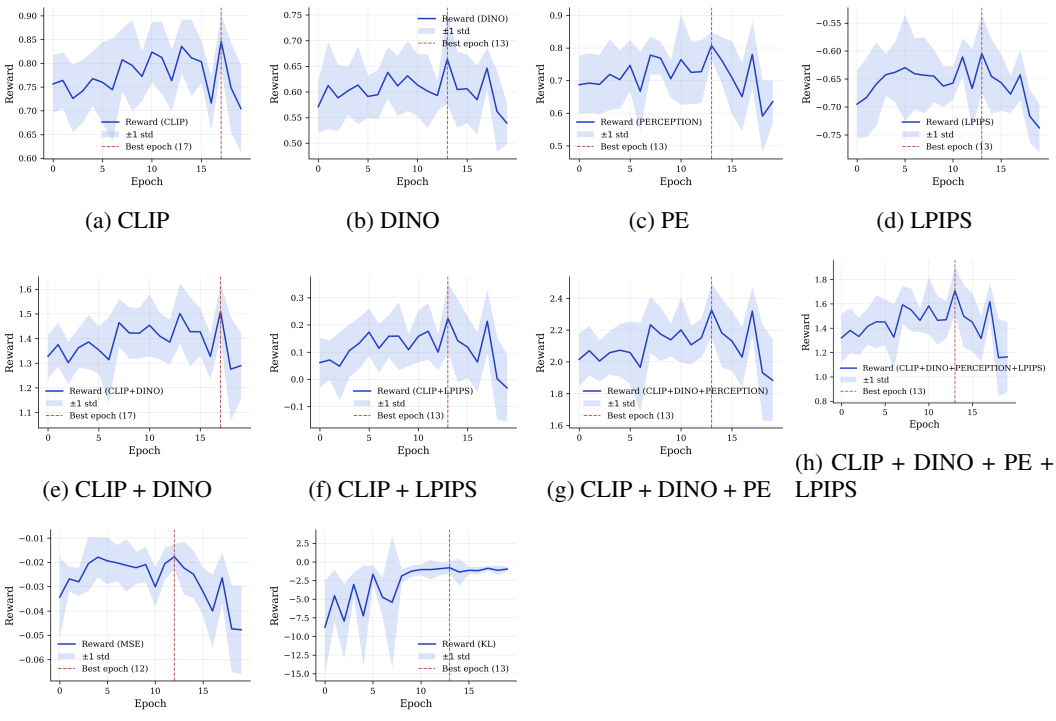
LPIPS, but not by the other seven semantic configurations, and single DINOv3 or DINOv3 + PE rewards are much worse. On this branch the gain of a representation-space reward is therefore specific to the encoder, and it is concentrated in configurations that contain CLIP. Combining encoders does not help monotonically: CLIP + LPIPS is worse than CLIP alone on ImageNet-64, while the full combination R_2 is the best. CLIPScore on COCO is essentially unchanged across rewards (0.3035–0.3058), so none of the rewards trades text alignment for image quality. Each additional encoder costs one more forward pass per trajectory (Appendix C).

I.1 Reward Optimization Curves

The tables summarize the end point of training. Figure 7 reports the training dynamics of every configuration under the same procedure, conditioning pool and seeds: the mean reward per epoch, a ± 1 standard-deviation band across the conditioning signals of that epoch, and the reward-maximizing epoch. Reward scales are not comparable across panels, since single-encoder cosine rewards and their equally weighted combinations both lie in $[-1, 1]$, while the MSE and KL controls are unbounded below. The quantity of interest is the shape of each curve.

Observations. Most of the improvement occurs within the first ten epochs for every configuration, consistent with the fine-tuning view of Section 3.1. The per-epoch reward is noisy, and the reward-maximizing epoch of every semantic configuration falls between 12 and 17 of these 20 epochs. That is where the semantic rewards are optimized in the sense that matters: past it the reward stays high or keeps rising while sample quality no longer improves, which is over-optimization rather than further progress. Figure 4 shows what that looks like on the sample of Figure 1, continued past the budget into the same phase. None of the curves is monotone, and the behavior is analyzed in Appendix J.

The pixel and divergence controls behave differently. The MSE and KL panels are not the semantic curves on a different scale. They rise slowly and are still rising at the end of the run, and the divergence control fluctuates strongly over the first epochs before settling. Neither reaches a comparable operating point within this budget: matching an image pixel by pixel, or matching a distribution, is a much larger fitting problem than agreeing with a frozen encoder, and it grows with the image, so on larger images and correspondingly larger search spaces these rewards converge slowly and are poorly suited to the low-training regime. The compensation is on the other side: because they constrain every direction rather than only the directions an encoder sees, their exploitable directions are far fewer, and over-optimization arrives later than it does for a semantic reward. Two properties follow from the same fact and should be kept in mind when reading these panels. They are unbounded below, so their scale is not comparable with the cosine rewards, which sit in $[-1, 1]$; and Proposition 2 does not apply to them at all, since it is a statement about the null space of an encoder's Jacobian and these rewards have no encoder.



(a) CLIP (b) DINO (c) PE (d) LPIPS

(e) CLIP + DINO (f) CLIP + LPIPS (g) CLIP + DINO + PE (h) CLIP + DINO + PE + LPIPS

(i) Negative MSE (j) Negative KL

Figure 7: Training reward curves for ten of the fourteen reward configurations of Table 5, under identical PPO optimization, conditioning pool and seeds. All panels are from the Stable Diffusion v1.5 branch ($50 \rightarrow 5$ steps); the corresponding curves on ImageNet-64 differ, and the text below says how. These are 20-epoch runs, shorter than the runs reported in Table 5, and are shown for the shape of the curves rather than for their end points. Solid lines show the mean reward per epoch, shaded regions ± 1 standard deviation across conditioning signals, and the dashed vertical line the reward-maximizing epoch. Panels (a)–(d) are single-encoder rewards, (e)–(h) combinations, and (i)–(j) the pixel and divergence controls.

The size of the search space sets the timing. The epoch at which over-optimization begins is not a property of the reward alone. The smaller the space the policy searches, the later it arrives: on ImageNet-64, where the images are 64×64 and the trajectory space is correspondingly smaller than on the 512×512 Stable Diffusion branch shown here, the same rewards stay useful for longer before the student starts finding directions the encoders cannot see. The numbers quoted above are therefore specific to the branch these panels come from, and the 12–17 epoch window should be read as an instance of the procedure in Appendix H, which fixes the training length from held-out FID on each branch, rather than as a constant of the method.

J Over-Optimization

The semantic reward is, to first order, blind to every image change in the null space of its encoders (Proposition 2). Once the student has matched the teacher in the directions the encoders see, further optimization can therefore raise the reward, or keep it constant, while changing the image in directions that image-level metrics do penalize. We observe this in two ways. The reward curves of Figure 7 keep rising or stay at their maximum late in training, after samples have stopped improving visually. When a run is continued past the fixed budget, samples degrade and acquire structured artifacts that the reward does not penalize (Figure 4).

Two consequences follow for the protocol. First, the training reward is not a valid stopping signal, which is why the training length is fixed once from held-out FID (Appendix H). Second, the reward configuration matters: summing encoders shrinks the first-order invariant set to the intersection of

their null spaces, which is consistent with the four-encoder reward R_2 being the best configuration on both branches (Appendix I). The KL regularizer of Section 3.4 bounds the drift of each update relative to the teacher policy, but we have not isolated its effect, and the COCO runs use $\beta = 0$. Preference rewards (Appendix K) are more exposed still, because their degenerate optimum lies along the same direction as the intended effect.

K Non-Differentiable Preference Rewards

Section 3.5 introduces a preference term into the distillation reward and summarizes the corresponding result. This appendix describes the two preference rewards and the evaluation protocol. The rewards were designed to encourage opposing behaviors: the compressibility reward favors removing high-frequency content, whereas the sharpness reward encourages concentrating it into narrow transitions. Our goal is to demonstrate that preference objectives can be incorporated into the distillation process without modifying the underlying procedure. We do not claim novelty in optimizing non-differentiable rewards using policy gradients, as DDPO (Black et al., 2024) already demonstrates this capability through JPEG compressibility optimization. Instead, we show that our framework is compatible with such objectives, enabling preferences to be incorporated into the distillation process.

K.1 Reward Definitions

Let $I \in [0,1]^{3\times H\times W}$ be a decoded sample, $g(I) \in [0,1]^{H\times W}$ its luminance ($0.299r + 0.587g + 0.114b$), and p a pixel. All quantities below are computed on the decoded image.

Compressibility. Let $\mathrm{JPEG}_q(I)$ be the byte string produced by a baseline JPEG encoder at quality q, and $|\cdot|$ its length in kilobytes:

$$R_{\mathrm{jpeg}}(I) \;=\; -\,\frac{\big|\mathrm{JPEG}_q(I)\big|}{\kappa}, \qquad q = 95, \quad \kappa = 100\,\mathrm{kB}. \tag{2}$$

The normalizer κ only changes the value of γ in equation 1 that gives a desired trade-off. The encoder is a discrete pipeline (block DCT, integer quantization, run-length and Huffman coding) whose output is a byte count, so R_{jpeg} is piecewise constant in its input and its gradient is zero wherever it exists. Differentiable approximations of JPEG exist, but they optimize a surrogate rather than the encoded size itself.

Sharpness. An edge is sharp when its transition completes in few pixels, independently of how much contrast it carries. Let $d(p) = \max_{p'\in\mathcal{N}_4(p)} |g(p) - g(p')|$ be the largest luminance step to a 4-neighbor and $r(p)$ the range of g over the 5×5 window at p, and define

$$\mathcal{E} = \big\{\, p : d(p) \ge 0.08 \;\wedge\; r(p) \ge 0.15 \,\big\}, \qquad \mathcal{E}_{\mathrm{sharp}} = \big\{\, p \in \mathcal{E} : d(p) \ge \tau\, r(p) \,\big\}, \tag{3}$$

with $\tau = 0.5$, so that $C(I) = |\mathcal{E}_{\mathrm{sharp}}|/\max(|\mathcal{E}|, 1)$ is the fraction of edge pixels whose transition completes within about $1/\tau = 2$ pixels. Because the test $d(p) \ge \tau r(p)$ compares a step with the local range, it measures edge *width* and is invariant to rescaling the contrast of an edge that is already in $\mathcal{E}$; contrast still matters through the absolute thresholds that define $\mathcal{E}$. With the edge-presence factor $\omega(I) = \min\big(1,\, |\mathcal{E}|/(0.01\,HW)\big)$,

$$R_{\mathrm{sharp}}(I) \;=\; \omega(I)\, C(I) \;\in\; [0,1]. \tag{4}$$

The factor ω stops an image with almost no edges from scoring highly on the strength of a handful of crisp ones. It does not remove the more general weakness of a ratio: C is also raised by suppressing soft edges rather than by sharpening them, and ω catches that only once fewer than 1% of pixels remain in $\mathcal{E}$. Appendix K.3 returns to this. Both C and ω are counts of pixels passing fixed thresholds; the thresholds were set by hand and not tuned against the outcome.

Why these are not differentiable. Each reward is a composition of indicator functions, integer counts, set cardinalities or an entropy coder. Each is constant on an open neighborhood of almost every input, so $\partial R/\partial I = 0$ wherever it is defined. A pathwise method such as DRaFT (Clark et al., 2024) receives an exactly zero gradient, and a supervised distillation loss cannot use them, since none defines a per-step target. The score-function estimator of Proposition 1 is unaffected: it weights $\nabla_\theta \log \pi_\theta$ by the scalar $R - b$ and never differentiates R.

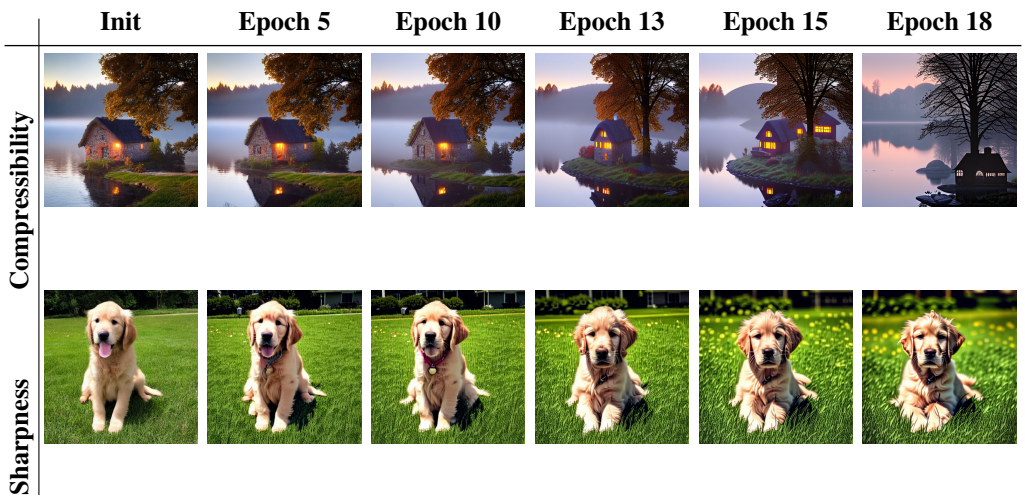


Figure 8: The COCO student trained with the distillation reward plus one non-differentiable preference, for two fixed prompts and seeds: compressibility R_{jpeg} equation 2 above, sharpness R_{sharp} equation 4 below. *Init* is the student before any update. Only the policy's weights differ between columns.

K.2 Protocol

We train the 8-step COCO student exactly as in Section 4, with the distillation reward R_1 and one preference term added as in equation 1. All other settings are those of the COCO column of Table 4: LoRA rank 4 on the UNet attention projections (Hu et al., 2022), learning rate 3×10^{-4}, clip range 10^{-4}, effective batch 8, guidance 5.0, $\beta = 0$, and 20 epochs; the preference weight γ is listed in the same table. Evaluation images are generated at every epoch from a fixed prompt set, each prompt with its own fixed seed, with the sampler settings held constant, so the only quantity that differs between two columns of Figure 8 is the policy's weights, as in Appendix F.

K.3 Results

Figure 8 follows two fixed prompts and one seed through training under each preference. The leftmost column is the student before any update. Under R_{jpeg} the sample loses high-frequency detail and moves toward large regions of near-constant and dark color, the configuration that minimizes encoded size. Under R_{sharp} the transitions tighten and fine detail is retained, the opposite direction in the same image statistic. In both cases the preference was added by adding one term to the reward: no other component of the procedure was modified.

The evidence is qualitative. A single prompt and seed cannot distinguish a systematic effect from a favorable example, and without image-quality metrics the figure does not show how much fidelity each preference costs; quantifying that trade-off requires the reward, FID and CLIPScore on held-out prompts across γ. The degenerate optimum of each preference also lies along the same direction as the intended effect: a flat color field for R_{jpeg}, and for R_{sharp} either haloing and grain or, as noted above, the suppression of soft edges, which raises the ratio C without sharpening anything. A preference metric should therefore never be reported on its own, and the training length should be fixed from held-out quality rather than from the reward, as in Appendix H.

L LoRA and the Clip Range

PPO keeps each update small through the clip range ϵ on the likelihood ratio, and the KL term of Section 3.4 keeps the policy near the teacher. A LoRA student adds a third restriction of a different kind: the adapter starts at zero and confines every update to a rank-r subspace of the attention weights, so the policy can move only along few directions. We test whether this matters

Table 7: Student parameterization crossed with the PPO clip range, EDM teacher, ImageNet-64 ($256 \to 8$), R_2 reward, 100 epochs. Mean $\pm$ sample sd over two seeds, last checkpoint. The full-parameter student uses a learning rate thirty times smaller than the LoRA student. Best in bold, second best bold and dash-underlined.

Student	ϵ	LR	P ↑	R ↑	D ↑	C ↑	FID ↓
LoRA ($r = 4$)	10^{-3}	3×10^{-4}	0.7257 ± 0.0013	$\mathbf{0.6003 \pm 0.0075}$	0.5751 ± 0.0102	0.7857 ± 0.0010	$\mathbf{20.01 \pm 0.46}$
LoRA ($r = 4$)	10^{-4}	3×10^{-4}	0.7391 ± 0.0016	$\underline{\mathbf{0.6145 \pm 0.0171}}$	$\mathbf{0.7829 \pm 0.0084}$	$\mathbf{0.8043 \pm 0.0001}$	$\underline{\mathbf{18.96 \pm 1.66}}$
Full	10^{-3}	1×10^{-5}	$\mathbf{0.7574 \pm 0.0535}$	0.4652 ± 0.1089	0.6131 ± 0.1521	0.6229 ± 0.2581	29.78 ± 0.85
Full	10^{-4}	1×10^{-5}	$\underline{\mathbf{0.7572 \pm 0.0147}}$	0.5677 ± 0.0250	$\underline{\mathbf{0.6624 \pm 0.0137}}$	$\mathbf{0.8129 \pm 0.0199}$	22.22 ± 4.81

for optimization by crossing the parameterization with the clip range on the EDM/ImageNet-64 branch.

Setup. All runs use the R_2 reward, $M = 8$, effective batch 32, four inner epochs, $\beta = 0.02$ and the 100-epoch budget of Table 1, and differ only in the student parameterization and in $\epsilon \in \{10^{-3}, 10^{-4}\}$. The LoRA student is the rank-4 adapter of Table 4 at learning rate 3×10^{-4}; the full-parameter student trains every weight of the UNet at 1×10^{-5}, a rate thirty times smaller, so the full-parameter rows are not disadvantaged by a larger step. Each configuration is run with two seeds and evaluated at the last checkpoint under the protocol of Appendix P. The LoRA, $\epsilon = 10^{-4}$ row is the ReDiF (R_2) configuration of Table 1, run separately.

Findings. Table 7 reports the four configurations. Under LoRA the clip range leaves the operating point intact: FID moves by 1.05 across the two settings (20.01 against 18.96), inside the seed spread of either row, and Recall is unchanged within the same spread. The diversity metrics do move, Density from 0.5751 to 0.7829 and Coverage from 0.7857 to 0.8043, so the adapter does not make the clip range irrelevant; both settings remain usable, and neither collapses. The full-parameter student behaves differently: it depends strongly on ϵ. At $\epsilon = 10^{-3}$ it is the worst configuration in FID (29.78), and one of its two seeds loses most of its diversity (Recall 0.39, Coverage 0.44, against 0.54 and 0.81 for the other), which is the source of the large spread in that row. Tightening the clip to 10^{-4} restores diversity and brings the mean FID to 22.22, but the two seeds still differ by almost seven FID points (18.81 and 25.62). The best single full-parameter run thus matches the LoRA runs, but the full-parameter student reaches that point only with the tighter clip and then unreliably, while the LoRA student reaches it at either setting. The full-parameter student does keep a higher Precision at $\epsilon = 10^{-4}$ (0.7572 against 0.7391), so its advantage, where it has one, is in fidelity rather than in diversity.

We read this as the adapter supplying part of the stabilization that the clip range otherwise has to provide alone: with LoRA neither setting of ϵ produces a degenerate run, and without it one setting does. We do not measure the per-update policy change directly, and with two seeds per configuration the comparison of means is indicative; the contrast in sensitivity, between rows that agree across settings and seeds and rows that do not, is the more robust observation. The ablation concerns ReDiF only and says nothing about whether LoRA helps or hurts the retrained baselines of Table 1.

M Effect of the Student Step Count

Section 4.1 states that the student step count M is bounded below by the need for a horizon over which the policy can explore, and above by the growth of estimator variance and per-iteration cost with the horizon. This appendix reports the measurements behind that choice on the EDM/ImageNet-64 branch, where the absence of guidance removes the confound of Section 3.4.

We repeat the matched-budget run of Section 4.2 for $M \in \{1, 2, 4, 8, 16\}$, holding the reward (R_1), conditioning pool, LoRA configuration, batch and epoch budget fixed, so every step count receives the same number of policy updates. Figure 9 reports FID, Recall, Coverage and Density as functions of M against the 256-step teacher. Each point is the mean over independent runs, between two and nine per step count as annotated, and the band is one standard deviation. The $M = 8$ point corresponds to the ReDiF (R_1) row of Table 1.

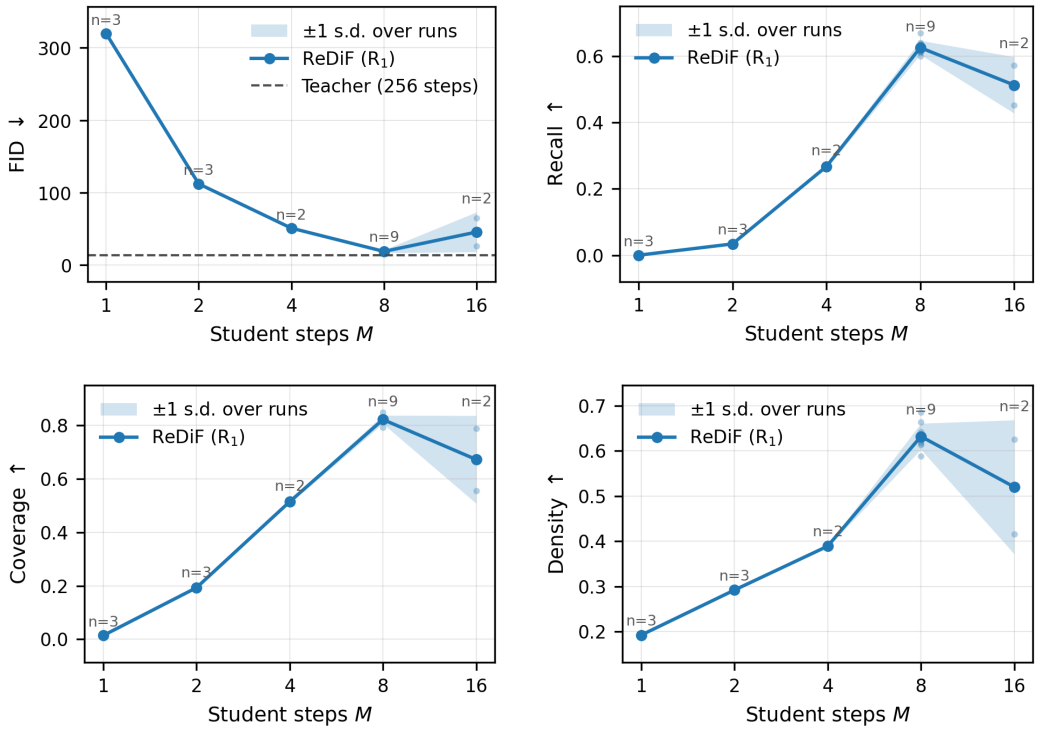


Figure 9: Effect of the student step count M on the EDM/ImageNet-64 branch under a fixed training budget. Top: FID (left) and Recall (right). Bottom: Coverage (left) and Density (right). Dashed lines mark the 256-step teacher and the dotted line the operating point $M = 8$. n is the number of runs at each M and the band is ± 1 s.d. over runs.

Three observations follow. First, very short horizons fail, and they fail most visibly in diversity. At $M \leq 2$ Recall and Coverage are close to zero, and at $M = 4$ they remain well below their best values even though FID has already recovered most of the way from its $M = 1$ value. The reason is specific to how the method works. ReDiF improves a student by exploring: the policy perturbs its own transitions, keeps the perturbations that score well and pushes probability mass toward them. A one- or two-step student has almost nothing to explore over. Its entire trajectory is one or two decisions, each of which fixes a large part of the image at once, so the policy has very little freedom: the perturbations it can make are either too small to change the sample or large enough to destroy it, and there are no later transitions left to correct an early move. What limits quality there is not how well the policy explores but how little there is to explore over. The task also changes character: a one- or two-step student has to memorize a direct map from noise to image rather than learn a policy that generalizes across trajectories, and diversity and generalization matter less to it than raw fitting capacity. Fitting that map is bound by iterations and compute, which is what the one-step literature spends thousands of GPU-hours on, so dense per-step targets have more to offer there than a terminal reward does and the regime is not a low-training one in the first place. This is also why we treat Recall and Coverage as primary metrics: the collapse at $M \leq 2$ shows up in diversity long before it shows up in FID.

Second, $M = 8$ is the best step count on all four metrics. At this point the student closes most of the gap between the 8-step teacher (FID 35.04) and the 256-step teacher (13.74), reaching 18.84. Its Coverage exceeds the teacher's (0.8223 against 0.7831), while Recall (0.6265 against 0.7103) and Density (0.6318 against 0.8930) remain below it.

Third, $M = 16$ regresses relative to $M = 8$ on every metric, and the spread across runs grows sharply. The failure at the long end is the opposite of the one at the short end. What the policy searches over is trajectories, and a trajectory is a choice at each of M transitions: the policy factorizes as $\prod_t \pi_\theta(a_t \mid s_t)$, so the space it must cover is a product over transitions and compounds with

Table 8: **Published results at convergence.** FID as reported by the original authors, except where marked: ImageNet-64 with 50k balanced samples, COCO-2014 zero-shot with 30k samples. Batch is the effective training batch and Devices the reported hardware, both as disclosed; † marks a third-party A100-hour estimate from Zhou et al. (2024b); ‡ marks an FID we quote from Song et al. (2023), who evaluate progressive and consistency distillation on this teacher, rather than from the method's own paper; $*$ marks the batch of the default ablation setting reported by Sauer et al. (2024), which the paper does not state to be the setting of its final run; *n/r* not reported; – no number under this protocol. ReDiF has not been evaluated under these protocols, so its FID is omitted; its numbers under the single-device protocol are in Table 1. Results published only under other COCO protocols are excluded rather than converted. Best published FID in bold, second best bold and dash-underlined; teacher rows are not ranked.

Method	Family	Steps	Aux. net	Trainable	Batch	Devices / training cost (reported)	FID ↓
(a) ImageNet-64							
EDM teacher (Karras et al., 2022)	–	256	–	–	–	–	1.36
Progressive Distill. (Salimans & Ho, 2022)	trajectory	$N \to 1$	interm. teachers	full	2048	∼64 GPUs; ∼5,533 A100-h†	15.39‡
Consistency Distill. (Song et al., 2023)	trajectory	$N \to 1$	target copy	full	2048	∼64 GPUs; ∼7,867 A100-h†	6.20‡
CTM (Kim et al., 2024)	trajectory	$N \to 1$	target copy + disc.	full	2048	∼64 GPUs; ∼902 A100-h†	1.92
DMD (Yin et al., 2024b)	distribution	$N \to 1$	score net	full	n/r	n/r	2.62
DMD2 (Yin et al., 2024a)	distribution	$N \to 1$	score net + disc.	full	280	7×A100, ∼2 d (∼336 A100-h)	**1.28**
SiD (Zhou et al., 2024a)	distribution	$N \to 1$	score net	full	8192	16×A100-40GB (accum. 32)	1.524
SiDA (Zhou et al., 2025b)	distribution	$N \to 1$	score net + disc.	full	8192	8×H100 (accum. 32)	**1.110**
TFD (Zhang et al., 2026)	drifting	$N \to 1$	none	full	n/r	200k iters	1.58
ReDiF (ours)	RL	$256 \to 8$	none	LoRA	32	1×A100-80GB, 3.7–3.8 h	–
(b) COCO, Stable Diffusion v1.5 teacher							
SD v1.5 teacher (Rombach et al., 2022)	–	50	–	–	–	–	8.74
Guided Distill. (Meng et al., 2023)	trajectory	$50 \to 4$	interm. teachers	full	n/r	n/r	–
LCM (Luo et al., 2023a)	trajectory	$50 \to 4$	target copy	full	72	∼32 A100-h	–
InstaFlow (Liu et al., 2024)	rectified flow	$50 \to 1$	none	full	1024	199 A100-days (∼4,780 A100-h)	13.10
ADD / SD-Turbo (Sauer et al., 2024)	adversarial	$50 \to 1$	disc. (+ real data)	full	128*	n/r	–
DMD (Yin et al., 2024b)	distribution	$50 \to 1$	score net	full	n/r	72×A100, ∼36 h (∼2,600 A100-h)	11.49
DMD2 (Yin et al., 2024a)	distribution	$50 \to 1$	score net + disc.	full	128	64×A100, ∼60 h (∼3,840 A100-h)	**8.35**
Guided SiD (GDD) (Zhou et al., 2025a)	distribution	$50 \to 1$	score net	full	n/r	20M fake images seen	**8.15**
ReDiF (ours)	RL	$50 \to 5$	none	LoRA	8	1×A100-80GB, 0.54–0.58 h	–

the horizon rather than growing in proportion to it. Coarsely, if one counts trajectories that differ at a fixed resolution on the state space, doubling the horizon squares that count; for continuous actions this is a counting heuristic rather than a statement about the measure, and the exact version is only that the trajectory distribution lives on a space of M times the per-state dimension. Either way the supervision does not grow with it: a single terminal scalar is still all the estimator has, and it is now spread over twice as many score terms (Proposition 1). Reaching a good policy over the larger space therefore takes more updates, and at a fixed batch that means more epochs, which is exactly what the low-training regime does not allow. With the epoch budget held at the value used everywhere else, the $M = 16$ runs have not converged: which part of the space each run settles in still varies between seeds, and that is what the widening band reports. With two runs at $M = 16$ we do not separate these effects quantitatively. We read the result as showing that, at this budget, the longer horizon does not pay for itself, not as evidence that longer students are worse at convergence; $M = 8$ is where the horizon is long enough to explore and short enough to cover within the budget.

N Published Results at Convergence

Section 4.2 compares algorithms in the low-training regime, which invites the objection that the baselines did not receive the compute their authors gave them. Table 8 represents each method by the best result its own authors report, quoted without re-implementation, together with the effective batch and hardware they disclose. The point is position on the quality–cost plane rather than state-of-the-art FID, so the cost columns are first-class. These methods are far ahead of where we retrained them, and the distance is the budget.

We report cost as disclosed and mark *n/r* where a paper does not state it, with one exception: for progressive distillation, consistency distillation and CTM on ImageNet-64 we quote the third-party A100-hour estimates of Zhou et al. (2024b), marked †. Batch is listed separately from accelerator-hours because the two are not interchangeable at fixed hardware.

These methods are not subsumed by our results and we do not claim otherwise. GDD (Zhou et al., 2025a) reaches the strongest COCO FID without real images but retains an auxiliary score network; TFD (Zhang et al., 2026) removes that network by working in the teacher's hidden states but esti-

Table 9: Learning rate used for each retrained baseline in Table 1. Values are taken from each method's own paper where it reports a setting for this teacher, and otherwise from the work reporting its strongest published result; none was selected by us. Objectives that regress onto a fixed teacher target take a larger rate than those whose target moves with an auxiliary network. Direct reward backpropagation on COCO uses the ReDiF rate because it optimizes the same reward. "Aux." is the rate of the fake-score network where one exists. All students are LoRA adapters trained with AdamW.

Branch	Method	LR	Aux. LR	Target type
ImageNet-64	Progressive Distillation	3×10^{-4}	–	fixed teacher target
	Consistency Distillation	3×10^{-4}	–	fixed teacher target
	DMD	1×10^{-4}	3×10^{-4}	moving target (aux. network)
	DMD2	1×10^{-4}	3×10^{-4}	moving target (aux. network)
	SiD	1×10^{-4}	3×10^{-4}	moving target (aux. network)
	SiDA	1×10^{-4}	3×10^{-4}	moving target (aux. network)
	Direct reward backprop.	1×10^{-4}	–	reward gradient
COCO	Progressive Distillation	1×10^{-4}	–	fixed teacher target
	Consistency Distillation	1×10^{-4}	–	EMA target
	DMD	5×10^{-5}	1×10^{-4}	moving target (aux. network)
	DMD2	5×10^{-5}	1×10^{-4}	moving target (aux. network)
	Direct reward backprop.	3×10^{-4}	–	matched to ReDiF

mates its drifting field against real data where available; DMD2 and the adversarial variants require real images outright. All keep a full-parameter student and a budget far above ours, and our position is the LoRA-scale, single-device corner of the same design space. Taken with Table 1, the ordering at convergence is substantially a function of the budget.

O Baseline Learning Rates

Every baseline retrained in Section 4.2 is run at a single learning rate, chosen per method rather than shared, because these objectives do not tolerate the same step size: losses that regress onto a fixed teacher target accept a larger rate than those whose target moves with an auxiliary network. We take each value from the method's own paper where it reports a setting for this teacher, and otherwise from the work that reports the strongest published result for that method. No value was selected by us. ReDiF's own hyperparameters are likewise adopted rather than searched: they follow the Stable Diffusion configuration of DDPO (Black et al., 2024), which optimizes the same clipped surrogate on the same backbone. Table 9 records the rate used for every method.

P Evaluation and Baseline Protocol

This section states the protocol behind Tables 1, 2, 5 and 6 in full, and how it relates to the published numbers of Table 8.

Evaluation. On ImageNet-64 we generate $10{,}000$ class-conditional samples with the deterministic sampler ($\eta = 0$) and compute FID and PRDC against $10{,}000$ real ImageNet-64 validation images. On COCO we generate one image for each of $5{,}000$ MS-COCO validation prompts and compare against the validation images. The same prompts, seeds and inference settings are used for every method. This protocol is chosen to be affordable on one device and to be identical across all retrained rows; it is not the protocol of the published numbers, which use 50k samples on ImageNet-64 and the 30k COCO-2014 zero-shot protocol of Yin et al. (2024b). Absolute values are therefore higher than published ones, and numbers from the two protocols are never placed in the same column. Every table reports the distillation ratio as teacher steps $\rightarrow$ student steps.

Baseline selection. A method is retrained on a branch only if its authors report results with that teacher, so that its published number in Table 8 refers to the same teacher. Direct reward backpropagation in the style of DRaFT (Clark et al., 2024) appears only in the matched-budget comparison:

it is not a step-distillation method and has no published step-distillation number, but it is the control that isolates the contribution of policy optimization. CTM (Kim et al., 2024) and TFD (Zhang et al., 2026) appear only in Table 8 and not in Table 1. Both were difficult to bring into the protocol: each carries a larger set of interacting hyperparameters than the other baselines (CTM through its jump distribution, adversarial term and target schedule, TFD through the hidden-state objective and the schedule of its drifting field), and their retrained quality was very sensitive to that setting. Within the single-device budget and the uniform learning-rate policy of Appendix O we did not find a configuration that reached a reasonable operating point, and reporting a number we could not tune would understate them rather than compare them. We therefore quote both at the budget their own authors used and make no matched-budget claim about them.

Budget procedure. All rows are trained for the same number of epochs on the same teacher, sampler grid, student step count, conditioning pool and hardware, so the budget is controlled by the epoch count rather than set per method. Two configurations depart from an otherwise uniform setup: DMD and DMD2 on ImageNet-64 use a larger batch, and the ImageNet-64 distribution-matching students use a higher LoRA rank than the rank-4 adapter used elsewhere, in both cases because the smaller setting left them at a degenerate point. The realized totals are in the PFLOPs column of Table 1, which Appendix C defines. All methods are reported at the last checkpoint of the budget. Learning rates are given in Appendix O. Gradient checkpointing and mixed precision are enabled for all methods. Because the direct-backpropagation control differentiates through the sampling chain, it cannot be memory-matched by default; its backward pass is truncated to the trailing sampler steps that fit in 80 GB, and this truncation is part of the comparison.

Training FLOPs. We compute analytical training FLOPs as the sum, over all trained and frozen networks, of (number of forward passes)$\times$(forward FLOPs) plus (number of backward passes)$\times$(backward FLOPs), taking backward cost as twice forward cost. This is independent of hardware, kernel efficiency and implementation quality. Every row trains a LoRA adapter rather than the full backbone, so the trainable parameter count is a small fraction of the model and the backward term is an upper bound on the realized cost; the convention is the same for all rows. Pool synthesis is included in the totals rather than counted separately (Appendix C) and is identical across rows. Inference FLOPs are never combined with training FLOPs.