REINFORCEMENT NOISE ADAPTATION FOR DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Creating data from noise is generative modeling; its converse, creating noise from data, has been neglected yet could be unexpectedly non-trivial for diffusion models. Conventional diffusion models use a stepwise noising and denoising training mechanism, which can be viewed as a serial generation of training samples for different noise levels. Major diffusion models predefine the noising schedule, which treats the samples from the easy-to-learn and the hard-to-learn distributions equally. In this work, we investigate the adaptive noise schedule. We propose the Reinforcement Noise Adaptation (ReiNA), a framework that casts forward noise scheduling as a sequential decision-making problem. During the forward and backward learning process of the diffusion models, an adaptive policy observes the current learning state and adjusts the noise schedule, thereby reallocating learning capacity to the under-learned distributions. On the public datasets (CIFAR-10, CelebA-HQ, and LSUN Bedroom), ReiNA consistently improves FID, IS, Precision, and Recall compared with some SOTA methods. In addition, experiments on structured material simulation (StructSim) show the adaptive framework can be applied to sophisticated physical simulation tasks. The ablation studies and further analyses verify the contribution of each ReiNA component and reveal that ReiNA learns phase-aware and batch-adaptive noising strategies.

1 INTRODUCTION

Diffusion models acquire denoising capability progressively across noise levels, where each noise level corresponds to a distinct reconstruction ability. The noise schedule controls how the learning capacity is distributed across these noise levels. Existing noise schedules, either the predefined static curves (Ho et al., 2020; Nichol & Dhariwal, 2021) or the data-driven learned schedules, remain fixed throughout training. However, the training of diffusion models exhibits phase transitions (Sclocchi et al., 2025; Wang et al., 2025). In different phases of training, the granularity of denoising shall be different. A static schedule cannot follow this shifting demand, causing redundant learning on well-learned distributions while under-learned distributions receive insufficient attention. This results in low training efficiency and stagnant loss reduction especially when the diffusion models try to reconstruct the low-level hidden features from noise (Sclocchi et al., 2025).

To address the limitations of the static noising schedules, we propose the Reinforcement Noise Adaptation (ReiNA), which makes the forward noising adaptive to the current denoising residual, redirecting learning capacity toward under-learned distributions. We propose that the noise schedule in the diffusion models should be conditioned on the training dynamics and be reflected in the training objective during the learning. This step-wise noising procedure could be modeled by a Markov Decision Process (MDP). Building upon this insight, we designed a reinforcement learning (RL) framework to control the forward noising process.

By continuously adjusting the training based on the evolving denoising residuals, our proposed ReiNA avoids the stagnation issue that occurs in static scheduling and consequently achieves a faster convergence rate and better generation quality on multiple tasks. We illustrate the comparison between the conventional static noising and the adaptive noising (i.e. ReiNA) in Figure 1. The diffusion models adopt a stepwise noising and denoising training mechanism, which can be viewed as a serial learning process over the noisy samples from different noise levels. Since the samples may contain both easy-to-learn and hard-to-learn distributions, those hard-to-learn distributions should

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.

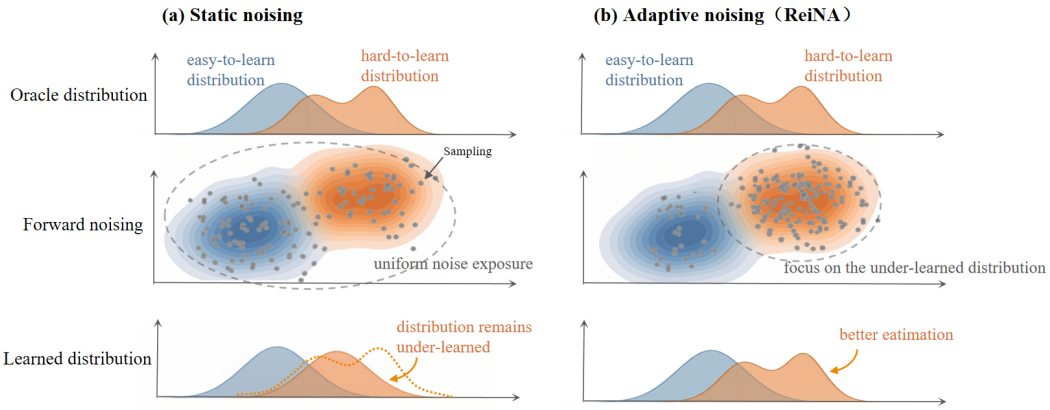


Figure 1: **Static noising treats easy-to-learn and hard-to-learn distributions equally, leaving the hard ones under-learned, while ReiNA reallocates learning capacity to provide a better estimation.** In each round of noising and denoising, the static noising schedule leads to a uniform sampling, providing insufficient attention to the hard-to-learn distributions (left). ReiNA redirects the learning capacity toward the under-learned distribution, effectively generating more samples for the hard-to-learn distributions at each denoising step (right).

receive more attention, i.e., get more samples for training. Our proposed ReiNA adaptively reallocates the learning capacity during the learning of the diffusion models.

Our contributions are summarized as follows:

- We propose ReiNA, a reinforcement learning framework that formulates adaptive forward noise scheduling for the diffusion models as a Markov Decision Process. ReiNA paves a novel way for the forward process of the diffusion models.
- ReiNA consistently improves the image generation quality on CIFAR-10 (32×32), CelebA-HQ (256×256), and LSUN Bedroom (256×256). A StructSim experiment demonstrates that the framework is also applicable to physical simulation tasks, where the sophisticated physical bounds challenge the conventional generation methods.

2 Related Work

2.1 Noise Scheduling in Diffusion Models

The sampling efficiency and quality of diffusion models are fundamentally governed by the noise schedule of the forward process. Early methods rely on predefined static schedules such as linear or cosine decay (Ho et al., 2020; Nichol & Dhariwal, 2021), which are now widely recognized as suboptimal. Subsequent work improves scheduling from multiple perspectives (Sakurama & Brian, 2025), beginning with DDIM's adjustable deterministic sampling scheme (Song et al., 2020) and later accelerated by ODE-based solvers such as DPM-Solver that implicitly optimize time-step utilization (Lu et al., 2022; Li, 2024). More recent efforts explicitly treat the schedule itself as an optimization variable (Wang et al., 2025; Zheng et al., 2024; Zhang et al., 2025; Zheng et al., 2026), including attempts to learn noise injection strategies during training (Bartosh et al., 2024). Notably, MuLAN (Sahoo et al., 2024) introduces a multivariate, instance-conditional noise schedule that achieves state-of-the-art density estimation by learning per-pixel noise strategies. Despite these advances, the resulting policies remain fixed after training and cannot adapt to the model's intermediate learning states or the current denoising residual, leaving both the forward noise level and its induced loss weight $w(t)$ static. Consequently, even learned schedules are fundamentally static, offering a dataset-level optimum that may be suboptimal for the evolving training dynamics.

Concurrently, several works optimize the timestep sampling distribution $p(t)$ rather than the noise schedule itself. Wang et al. (2025) propose asymmetric timestep weighting (SpeeD) that accelerates training by up-weighting under-represented diffusion times, while Zheng et al. (2024; 2026)

learn non-uniform sampling tprobabilities. These approaches modulate which timesteps are visited, whereas ReiNA modulates the noise level at each timestep together with its induced loss weight $w(t)$, and does so in response to the current training state.

2.2 Reinforcement Learning for Controllable Generation

RL has emerged as a powerful framework for adaptive control in generative models. Foundational works such as SeqGAN (Yu et al., 2017) pioneered the use of policy gradient methods to treat generation as sequential decision-making, a paradigm quickly extended to image generation (Miao et al., 2024; Geng et al., 2025). In the context of diffusion models, RL is predominantly applied during fine-tuning to align outputs with human preferences (Cao et al., 2023; Yuan et al., 2025; Franceschelli & Musolesi, 2024; Ganin et al., 2018). Beyond diffusion, work on large language models has demonstrated the utility of RL for adaptive computation, such as dynamically allocating computational budgets based on the model's current state (Jiang et al., 2024; Dargiry, 2025; Wu et al., 2025; Arora & Zanette, 2025).

However, existing RL-based controllers for diffusion almost exclusively focus on the reverse denoising trajectory or parameter tuning. They leave the forward noising process unaddressed, where the consequence of a single noise injection step only manifests after multiple training iterations. This gap motivates our formulation that treats forward noise scheduling as a sequential decision-making problem under delayed feedback.

3 Method

3.1 Preliminaries

A diffusion model defines a forward process that gradually corrupts data $x_0 \sim q(x_0)$ via a variance schedule $\{\beta_t\}_{t=1}^T$. Letting $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$, one can sample an arbitrary noisy latent directly:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{1}$$

A denoising network ϵ_θ is trained to recover ϵ from x_t. The network is conditioned on the realized noise level through the log signal-to-noise ratio:

$$\gamma_t = \log \frac{\bar{\alpha}_t}{1 - \bar{\alpha}_t}. \tag{2}$$

The standard ϵ prediction loss is:

$$\mathcal{L}(\theta) = \mathbb{E}_{t, x_0, \epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, \gamma_t)\|^2 \right]. \tag{3}$$

The fixed schedule β_t and uniform prior $p(t)$ define a static training scheme that cannot adapt to the model's evolving learning dynamics, motivating the adaptive formulation of ReiNA.

3.2 Reinforcement Noise Adaptation

3.2.1 Formulation as an MDP

A noise schedule $\sigma(t)$ controls two aspects of diffusion training, namely where information is destroyed and how sharply it is destroyed, which correspond to the inflection point and the slope. Early training benefits from broad, uniform noise exposure to learn global structure, while late training requires concentrated noise at specific levels to refine fine-grained details (Sclocchi et al., 2025). A static noising schedule does not distinguish the easy-to-learn and the hard-to-learn distributions, so the well-learned distributions keep occupying the learning capacity.

To address this, we adjust the noise distribution by controlling its overall shape while keeping the schedule endpoints fixed so that $\sigma(0) = \sigma_{\min}$ and $\sigma(1) = \sigma_{\max}$. The schedule derivative $\sigma'_k(t)$ directly enters the training loss weight, so reshaping the noise distribution at every iteration redirects capacity toward the noise levels and samples that remain poorly learned.

3.2.2 THE STATE SPACE $\mathcal{S}$

The state $s_k \in \mathcal{S}$ summarizes the training dynamics observed before the current RL decision. At iteration k, ReiNA uses a compact state representation

$$s_k = [k/K_{\max},\, \ell_k,\, g_k,\, f_k,\, h_k]^\top. \tag{4}$$

Detailed definitions of each state component are provided in Appendix A.1. Together, these signals provide stage, optimization, data, and temporal context, which allows the policy to adapt the noise schedule to the current training phase and the recently observed batch characteristics.

3.2.3 THE STATE TRANSITION $\mathcal{P}$

Given (s_k, a_k), the policy instantiates $\sigma_k(t)$. The environment then samples a minibatch $x_0^{(k)}$, diffusion times t, and Gaussian noise ϵ, constructs $x_t^{(k)}$, and updates the denoising model:

$$\theta_{k+1} = \theta_k - \eta_\theta \nabla_\theta \mathcal{L}_{\text{new}}(\theta_k; x_t^{(k)}, \gamma_k(t)). \tag{5}$$

The next state s_{k+1} is computed from the updated loss, gradient norm, data features, and action history. Thus, an MDP transition corresponds to one policy-controlled training update.

3.2.4 THE ADAPTIVE NOISING POLICY

Building on Section 3.2.1, we capture the noise shape with a slope m_k and midpoint n_k at iteration k. Therefore, the action space $\mathcal{A}$ is defined as a two-dimensional continuous domain. The policy π_ψ outputs raw parameters $a_k = (u_k, v_k)$, which are transformed into schedule parameters m_k and n_k as follows:

$$m_k = \text{Clamp}(\text{Softplus}(u_k) + \epsilon_0, 0, 20), \quad n_k = \text{Sigmoid}(v_k). \tag{6}$$

These parameters instantiate a differentiable cumulative noise schedule over normalized diffusion time $t \in [0, 1]$:

$$\sigma_k(t) = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \cdot \frac{\text{Sigmoid}\left(m_k(t - n_k)\right) - \text{Sigmoid}\left(-m_k n_k\right)}{\text{Sigmoid}\left(m_k(1 - n_k)\right) - \text{Sigmoid}\left(-m_k n_k\right)}. \tag{7}$$

This formulation guarantees $\sigma'(t) \geq 0$, ensuring a valid monotonic noising process, where $\sigma_k(t)$ denotes the cumulative noise standard deviation ($\bar{\alpha}_k(t) = 1 - \sigma_k(t)^2$); Appendix A.6 details the connection to the resulting loss weight. After selecting a_k, the environment samples diffusion times t and constructs noisy samples:

$$x_t^{(k)} = \sqrt{1 - \sigma_k(t)^2}\, x_0^{(k)} + \sigma_k(t)\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{8}$$

3.2.5 THE BOOMERANG REWARD $\mathcal{R}$

Unlike standard RL rewards that only evaluate the terminal outcome of an episode, ReiNA's reward makes use of the inherent forward-reverse loop in diffusion. A noising action produces a corrupted sample, which the reverse denoiser should recover, and the recovery quality directly measures the utility of the original noising decision. We term this the $\mathcal{R}_{\text{boomerang}}$, reflecting that each forward action returns through the reverse path as a training signal. Formally, at iteration k:

$$\mathcal{R}_{\text{boom}}(k) = \sum_{i \in \mathcal{P}} \lambda_i\, R_i(k) \;-\; \sum_{j \in \mathcal{N}} \lambda_j\, R_j(k), \tag{9}$$

where $\mathcal{P} = \{\text{cyc}, \text{align}, \text{learn}, \text{temp}\}$ contains four signal terms that score recoverability, residual alignment, learnability, and delayed progress, and $\mathcal{N} = \{\text{cost}, \text{smooth}\}$ contains two regularization terms that penalize excessive corruption and abrupt policy changes. Full definitions are in Appendix A.2.

Particularly, the temporal term captures this delayed effect by checking whether the noise levels emphasized by action a_k become easier to denoise after a short horizon:

$$R_{\text{temp}}(k) = \text{clip}\left(\bar{e}_{\mathcal{B}_k}(k) - \bar{e}_{\mathcal{B}_k}(k + H), -c_{\text{temp}}, c_{\text{temp}}\right), \tag{10}$$

where $\bar{e}_{\mathcal{B}_k}$ denotes the validation residual averaged over the noise levels emphasized by action a_k, and H is a short training horizon. PPO (Schulman et al., 2017) then updates the policy and value network from discounted trajectories of $\mathcal{R}_{\text{boom}}$, as detailed in Appendix A.3.

Algorithm 1 Reinforcement Noise Adaptation (ReiNA)

Require: Diffusion model ϵ_{θ_0}, policy π_{ψ_0}, value network V_{ξ_0}, dataset $\mathcal{D}$, PPO interval C, temporal horizon H.

1: **for** training iteration $k = 1, 2, \ldots$ **do**
2: Observe state s_k (progress, loss, gradient, batch features, action history)
3: Sample action $a_k{=}(u_k, v_k) \sim \pi_\psi(\cdot \mid s_k)$; calculate (m_k, n_k) based on Eq. 6
4: Instantiate the normalized schedule $\sigma_k(t)$ (Eq. 7)
5: Compute $\sigma_k'(t)$ and $\gamma_k(t)$; sample t; construct $x_t^{(k)} = \sqrt{1-\sigma_k(t)^2}\, x_0^{(k)} + \sigma_k(t)\,\epsilon$
6: Update $\theta_k \leftarrow \theta_{k-1} - \eta_\theta \nabla_\theta \mathcal{L}_{\text{new}}(\theta)$ with $\epsilon_\theta(x_t^{(k)}, \gamma_k(t))$ and weight $w(t) = 2\sigma_k'(t)/\sigma_k(t)^3$
7: Compute $\mathcal{R}_{\text{boom}}(k)$ from cycle consistency, residual alignment, learnability, and temporal return
8: Update π_ψ, V_ξ via PPO (Appendix A.3)
9: **end for**

3.3 Train the ReiNA Model

The adaptive selection of the noise schedule $\sigma_k(t)$ directly affects the training signal received by the model at each step. To quantify this effect, we derive the loss function of the diffusion model from a continuous-time perspective, thereby clearly incorporating the dynamic changes of the noise schedule into the training objective.

We express the forward process under the policy-instantiated schedule $\sigma_k(t)$. For a fixed training iteration k, we omit the subscript k and write $\sigma(t)$ for clarity. A key quantity characterizing the forward process is the signal-to-noise ratio (SNR), defined as $V(t) = (1-\sigma(t)^2)/\sigma(t)^2$. Its derivative with respect to time captures the rate at which information is corrupted:

$$V'(t) = \frac{d}{dt}\left(\frac{1-\sigma(t)^2}{\sigma(t)^2}\right) = -\frac{2\sigma(t)\sigma'(t)}{\sigma(t)^4} = -\frac{2\sigma'(t)}{\sigma(t)^3}. \tag{11}$$

When reformulating the ELBO for a non-stationary and differentiable noise schedule, the derivative $V'(t)$ naturally appears in the loss weighting. Under this approximation, the loss term for predicting x_0 from x_t is weighted by $-V'(t)$, leading to the following reformulated training objective:

$$\mathcal{L}_{\text{new}}(\theta) = \frac{1}{2}\,\mathbb{E}_{x_t,t}\left[\frac{2\sigma'(t)}{\sigma(t)^3}\cdot\|x_0 - \hat{x}_0(x_t, \gamma(t))\|_2^2\right], \tag{12}$$

where $\gamma(t) = \log\big((1-\sigma(t)^2)/\sigma(t)^2\big)$ and $\hat{x}_0(x_t, \gamma(t)) = \frac{x_t - \sigma(t)\epsilon_\theta(x_t,\gamma(t))}{\sqrt{1-\sigma(t)^2}}$ is the prediction of the clean data. A detailed derivation of $\mathcal{L}_{\text{new}}(\theta)$ is provided in Appendix A.6.

The outer-loop RL policy π_ψ, by controlling $\sigma(t)$ and implicitly $\sigma'(t)$, directly shapes the inner-loop optimization landscape of θ. In particular, when the policy increases the slope m_k, this unimodal weighting $\sigma'(t)$ becomes more concentrated around the inflection point n_k, causing $w(t)$ to place more weight there; the optimizer can then focus its capacity there. Shifting n_k moves this range across the noise spectrum.

The parameters (m_k, n_k) thus implement an implicit curriculum that emerges from reward optimization. Unlike a static schedule, this curriculum continuously aligns the noise level and its loss weight with the model's current denoising residual, redirecting capacity toward under-learned noise levels and avoiding the mid-stage gradient oscillation.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets and backbone

We evaluate on three image datasets spanning two resolutions: *(i)* CIFAR-10 (Krizhevsky et al., 2009) (32×32, class-conditional, 50K/10K train/test); *(ii)* CelebA-HQ (Karras et al., 2017)

(256×256, unconditional, 27K/3K train/test); *(iii)* LSUN Bedroom (Yu et al., 2015) (256×256, unconditional, $\approx$3.03M train images, 300 test).

For the 32×32 setting, all methods share a U-Net backbone; training runs for 800 epochs ($\approx$313K steps) with batch size 128. For the 256×256 setting, we adopt a larger U-Net backbone; CelebA-HQ trains for 500 epochs ($\approx$210K steps) and LSUN Bedroom for 50 epochs ($\approx$2.37M steps), both with batch size 64. All experiments use Adam (β_1=0.9, β_2=0.999). Architecture details are in Appendix A.7, and full reward weights and network details are listed in Appendix A.4.

In addition to the image datasets, we also prepare a structured materials configuration dataset of ANSYS CFX boundary blocks for cross-domain validation, which is written in Structured Materials Simulation Script (StructSim), a script language for physical simulation. Each sample is a short structured text specifying boundary options, numeric fields, and optional child blocks. Dataset construction procedure and the schema are detailed in Appendix A.20.

4.1.2 BASELINES

We compare against three families of generative models: *(i) Static/Fixed:* DDPM linear (Ho et al., 2020) and cosine (Nichol & Dhariwal, 2021) schedules; *(ii) Static/Learned:* VDM (Kingma et al., 2021), NFDM (Bartosh et al., 2024), and MuLAN-style (Sahoo et al., 2024); *(iii) Adaptive/Sampling:* SpeeD-style asymmetric timestep weighting (Wang et al., 2025). On CIFAR-10, all six baselines are compared; on CelebA-HQ and LSUN Bedroom, we compare the five strongest baselines (DDPM linear/cosine, NFDM, MuLAN, SpeeD). Two diagnostic heuristic schedulers (Grad-React, Loss-Trig) are reported in Appendix A.12.

4.1.3 EVALUATION PROTOCOL

Generative quality is measured by FID (Heusel et al., 2017), Inception Score (IS), Precision, and Recall, computed via torch-fidelity on 10 000 generated samples versus the respective validation split of each dataset. All models use 1000-step ancestral sampling. For ReiNA, sampling uses the schedule induced by the converged training-time controller, denoted as $\sigma_{\mathrm{gen}}(t)$, and the denoiser is queried with the corresponding $\gamma_{\mathrm{gen}}(t)$. All main-table results report mean $\pm$ std over 3 independent seeds ($\{0,1,2\}$). ReiNA improves FID significantly on all three datasets ($p<0.05$). Training efficiency is measured by FID at epoch 50 (CIFAR-10). All experiments run on a single NVIDIA A800-SXM4-80GB GPU. Full evaluation details are in Appendix A.10.

4.2 PERFORMANCE ON THE IMAGE DATASETS

Table 1 summarizes the results across all three image datasets.

CIFAR-10 (32×32). ReiNA achieves the lowest FID (6.51 ± 0.08) and highest IS, Precision, and Recall among all methods, with the FID improvement over MuLAN (7.07 ± 0.11) statistically significant ($p<0.05$). VDM lags behind even static DDPM schedules in our controlled setting (explained in Appendix A.17); SpeeD-style reweighting improves only marginally over DDPM-linear.

CelebA-HQ and LSUN Bedroom (256×256). On CelebA-HQ, ReiNA achieves FID 9.91 ± 0.15, reducing the gap from MuLAN (10.66 ± 0.19) by 0.75 FID points and from DDPM-linear (12.79 ± 0.34) by 2.88 points. On LSUN Bedroom, ReiNA obtains FID 7.91 ± 0.18, outperforming MuLAN (8.57 ± 0.23) by 0.66 points. As Figure 2 shows, ReiNA produces results superior to static DDPM linear and cosine schedules under matched initial noise. These results demonstrate that ReiNA generalizes from low-resolution class-conditional generation to high-resolution unconditional settings. Additional image visualization results are shown in Appendix A.21.

Early-stage convergence. At epoch 50 on CIFAR-10 (seed 0), ReiNA already reaches FID 9.7, compared to 14.8 for DDPM-linear and 11.2 for MuLAN, indicating faster early-stage convergence driven by adaptive forward noising. The normalized training residual curves in Appendix A.11 further show that ReiNA reduces residual error faster and avoids the mid-stage stall observed in static schedules.



Figure 2: **Qualitative comparison on LSUN Bedroom.** Each column is generated from the same initial noise x_T. Top: DDPM (linear). Middle: DDPM (cosine). Bottom: ReiNA (ours). Red boxes mark structural artifacts of the static schedules. ReiNA yields cleaner layout, more stable lighting, and fewer texture artifacts than the two static schedules, consistent with the FID ranking in Table 1.

Table 1: Main results across three datasets (mean $\pm$ std over 3 seeds). Best in **bold**, second-best underlined.

Dataset	Method	FID↓	IS↑	Prec.↑	Recall↑
CIFAR-10	DDPM (linear)	10.35 $\pm$ 0.19	8.73 $\pm$ 0.06	0.55 $\pm$ 0.02	0.42 $\pm$ 0.02
	DDPM (cosine)	9.51 $\pm$ 0.14	8.88 $\pm$ 0.04	0.57 $\pm$ 0.02	0.44 $\pm$ 0.02
	VDM (VLB)	11.84 $\pm$ 0.28	8.51 $\pm$ 0.06	0.54 $\pm$ 0.03	0.38 $\pm$ 0.03
	VDM (MSE)	12.63 $\pm$ 0.34	8.32 $\pm$ 0.04	0.51 $\pm$ 0.03	0.36 $\pm$ 0.03
	NFDM	7.65 $\pm$ 0.13	9.26 $\pm$ 0.05	0.60 $\pm$ 0.02	0.49 $\pm$ 0.02
	MuLAN	7.07 $\pm$ 0.11	9.48 $\pm$ 0.05	0.62 $\pm$ 0.02	0.53 $\pm$ 0.02
	SpeeD-style	8.91 $\pm$ 0.16	9.04 $\pm$ 0.03	0.58 $\pm$ 0.02	0.47 $\pm$ 0.02
	ReiNA (ours)	**6.51** $\pm$ 0.08	**9.77** $\pm$ 0.04	**0.66** $\pm$ 0.01	**0.56** $\pm$ 0.01
CelebA-HQ	DDPM (linear)	12.79 $\pm$ 0.34	3.04 $\pm$ 0.08	0.61 $\pm$ 0.02	0.40 $\pm$ 0.02
	DDPM (cosine)	12.06 $\pm$ 0.25	3.13 $\pm$ 0.07	0.62 $\pm$ 0.02	0.42 $\pm$ 0.02
	NFDM	11.24 $\pm$ 0.23	3.25 $\pm$ 0.07	0.65 $\pm$ 0.02	0.45 $\pm$ 0.02
	MuLAN	10.66 $\pm$ 0.19	3.29 $\pm$ 0.07	0.66 $\pm$ 0.02	0.46 $\pm$ 0.02
	SpeeD-style	11.78 $\pm$ 0.28	3.17 $\pm$ 0.05	0.63 $\pm$ 0.02	0.43 $\pm$ 0.02
	ReiNA (ours)	**9.91** $\pm$ 0.15	**3.47** $\pm$ 0.06	**0.70** $\pm$ 0.01	**0.51** $\pm$ 0.01
LSUN Bed.	DDPM (linear)	10.48 $\pm$ 0.39	3.24 $\pm$ 0.10	0.58 $\pm$ 0.02	0.38 $\pm$ 0.02
	DDPM (cosine)	9.82 $\pm$ 0.36	3.34 $\pm$ 0.07	0.59 $\pm$ 0.02	0.40 $\pm$ 0.02
	NFDM	9.08 $\pm$ 0.27	3.47 $\pm$ 0.09	0.61 $\pm$ 0.02	0.43 $\pm$ 0.02
	MuLAN	8.57 $\pm$ 0.23	3.51 $\pm$ 0.09	0.63 $\pm$ 0.02	0.44 $\pm$ 0.02
	SpeeD-style	9.56 $\pm$ 0.30	3.36 $\pm$ 0.11	0.60 $\pm$ 0.02	0.41 $\pm$ 0.02
	ReiNA (ours)	**7.91** $\pm$ 0.18	**3.71** $\pm$ 0.08	**0.68** $\pm$ 0.01	**0.49** $\pm$ 0.01

4.3 Cross-Domain Validation on StructSim

Unlike images, where denoising difficulty varies smoothly across samples, StructSim shows discrete structural heterogeneity. Simple conditions such as `WALL+NONE` involve few fields and no child blocks, while complex conditions such as `INLET+FLOW_DIRECTION` need nested children and extra numeric fields. A static noise schedule applies the same loss weight $w(t)$ to both. As a result, it wastes the learning capacity on simple distributions that are already well-learned, while giving too little attention to those hard ones. Due to the discrete difficulty gap, the unfocused noise allocation problem becomes even more severe.

Table 2 confirms our hypothesis. ReiNA improves all metrics, but the gain is concentrated on the Hard column, with $+0.239$ over DDPM cosine compared to $+0.079$ on OptAcc. This suggests that the residual-alignment and batch-adaptation mechanisms redirect capacity toward the structurally

Table 2: StructSim generation (shared backbone; only the forward noise controller differs). OPTACC: fraction of decoded spans whose root-option matches the conditioned decision; CHILDCONS: fraction whose child blocks exactly match the enabled-child set; FIELDF1: F1 over required numeric fields implied by the chosen option; HARD: CHILDCONS on `INLET+FLOW_DIRECTION` only. Best in **bold**. Mean ± std over 3 seeds.

Method	OptAcc↑	ChildCons↑	FieldF1↑	Hard↑
DDPM (cosine)	0.809 ± 0.016	0.738 ± 0.019	0.806 ± 0.015	0.614 ± 0.022
MuLAN-style	0.858 ± 0.012	0.821 ± 0.013	0.844 ± 0.014	0.731 ± 0.021
ReiNA	$\mathbf{0.888} \pm 0.007$	$\mathbf{0.877} \pm 0.009$	$\mathbf{0.889} \pm 0.010$	$\mathbf{0.853} \pm 0.012$

hardest conditions. This matches the batch-difficulty adaptation observed on CIFAR-10 as shown in Figure 4, and demonstrates that adaptive forward noising is not tied to a specific modality. The more heterogeneous the difficulty distribution, the greater the advantage over static schedules.

4.4 ABLATION STUDIES

Table 3 summarizes the ablations used to isolate the contribution of the boomerang reward (mean ± std over 3 seeds are reported in Appendix A.16.1). The cycle term tests whether the selected corruption remains recoverable, the residual-alignment term tests whether ReiNA truly focuses on under-learned noise levels, and the temporal term tests whether short-horizon residual improvement provides useful credit assignment for delayed learning effects. Additional state, action, and sensitivity ablations are described in Appendix A.16, and reward-weight sensitivity in Appendix A.13. Other key findings are verified on CelebA-HQ in Appendix A.19, and full comparison of training overhead can be found in Appendix A.18.

Table 3: Boomerang reward ablation on CIFAR-10.

Variant	FID↓	IS↑	Prec.↑	Recall↑
ReiNA (full)	6.49	9.78	0.65	0.57
w/o R_{cyc}	7.50	9.14	0.58	0.49
w/o R_{align}	7.16	9.33	0.60	0.51
w/o R_{temp}	7.31	9.18	0.59	0.50

4.5 MECHANISM ANALYSIS

The following mechanism analysis is conducted on CIFAR-10 to provide interpretable insight into ReiNA's learned policy behavior.

4.5.1 EVOLUTION OF NOISE SCHEDULE ACROSS TRAINING STAGES

Figure 3 shows that both schedule parameters evolve monotonically, with m_k rising from ≈ 0.4 to ≈ 3.5 and steepening the schedule. Meanwhile, n_k climbs past the static baseline to ≈ 0.83, shifting the inflection toward later diffusion times. The narrowing P10–P90 bands confirm that the schedule settles into a stable operating region after early exploration, which keeps the per-step loss weight $w(t)$ coherent and suppresses the mid-stage gradient oscillation induced by a static schedule. Consistent with this mechanism, the residual-alignment term encourages $w(t)$ to track noise levels with high exponentially averaged denoising error (Appendix A.2). The resulting $\sigma(t)$ curves (Eq. 7) share fixed endpoints and transition from a near-linear Early ramp to a late-rising Late profile that remains flatter at moderate t and concentrates $\sigma'(t)$ near $t \to 1$, progressively reshaping the implicit loss weighting $w(t) = 2\sigma'(t)/\sigma(t)^3$ in Eq. 12 to emphasize high-noise timesteps that remain under-learned (see Appendix A.15). This three-stage evolution is consistent with the phase-transition phenomenon in diffusion training reported by Sclocchi et al. (2025).

4.5.2 BATCH-ADAPTIVE NOISE ALLOCATION BY SAMPLE DIFFICULTY

We further examine whether ReiNA adapts its noising decisions to the difficulty of training batches. We stratify batches into easy, mixed, and hard groups according to their initial loss, and record the selected noise level σ_t throughout training.

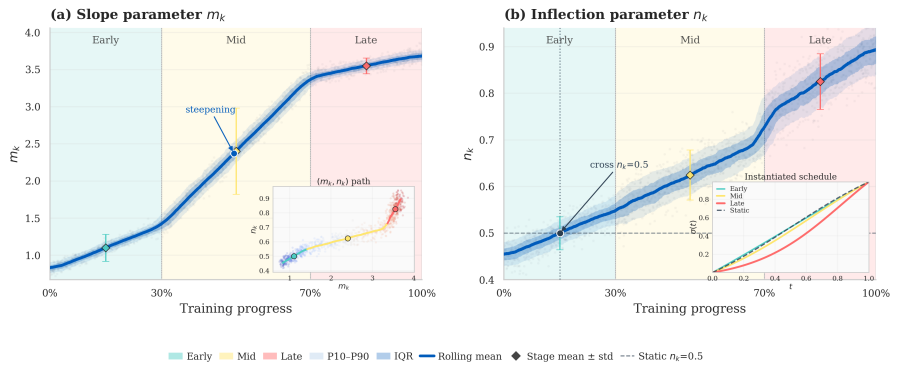


Figure 3: **Evolution of schedule parameters across training.** (a) Slope parameter m_k; inset traces the joint (m_k, n_k) trajectory, colored by stage. (b) Inflection parameter n_k; inset shows stage-wise $\sigma(t)$ from Eq. 7 versus a static reference. All curves share $\sigma(0) = \sigma_{\min}$ and $\sigma(1) = \sigma_{\max}$; Late shifts the rise toward later t (late-rising), rather than changing the terminal noise. Thick lines: rolling means; bands: IQR (dark) and P10–P90 (light); diamonds: stage means ± std.

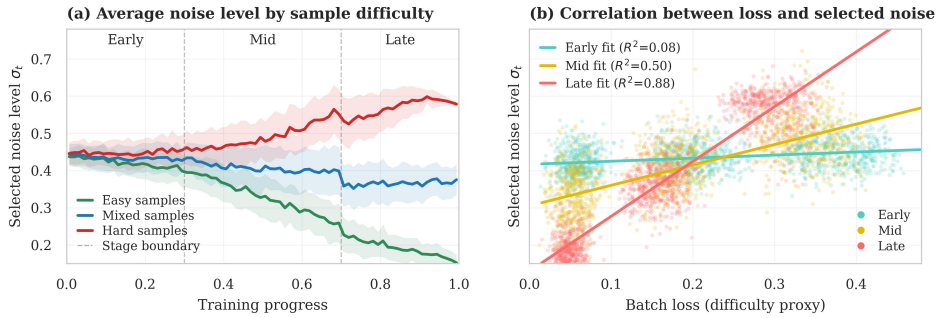


Figure 4: **Batch-adaptive noise allocation.** (a) Average selected noise level σ_t for easy, mixed, and hard batches across training. ReiNA assigns similar noise early on, but increasingly separates the groups as training progresses. (b) Loss-noise correlation by training stage. The rising R^2 from Early to Late shows that ReiNA progressively aligns stronger noise with harder batches.

As shown in Figure 4(a), the three groups receive similar noise levels in the early stage, when the model has not yet identified which samples are hard. As training progresses, their trajectories separate, with hard batches assigned higher noise, mixed batches remaining intermediate, and easy batches receiving lower noise. This separation is strongest in the late stage, suggesting that ReiNA increasingly allocates noise according to the residual difficulty of each batch.

Figure 4(b) confirms this trend through a loss-noise correlation analysis. The early-stage fit is weak and nearly flat ($R^2 = 0.07$), while the correlation strengthens in the mid stage ($R^2 = 0.52$) and becomes pronounced in the late stage ($R^2 = 0.87$). These results indicate that ReiNA aligns noising not only with the training stage, but also with the lagged batch difficulty signal, assigning stronger noise to harder batches while avoiding excessive corruption of easier ones.

5 Conclusion

We presented ReiNA, a framework that formulates adaptive forward noising as a residual-aligned training process for diffusion models. On three image datasets, ReiNA consistently improves over the static, the learned, and the adaptive baselines. Physical simulation results on StructSim further demonstrate the benefits of ReiNA on complex tasks. The ablation studies and the mechanistic analyses confirm the phase-aware and batch-adaptive behavior, supporting the view of forward noise scheduling as an adaptive control problem.

REFERENCES

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Grigory Bartosh, Dmitry Vetrov, and Christian A Naesseth. Neural flow diffusion models: Learnable forward process for improved diffusion modelling. *Advances in Neural Information Processing Systems*, 37:73952–73985, 2024.

Yuanjiang Cao, Quan Z Sheng, Julian McAuley, and Lina Yao. Reinforcement learning for generative ai: A survey. *arXiv preprint arXiv:2308.14328*, 2023.

Mohammad Hadi Dadizadeh Dargiry. Adaptive optimization of resource allocation in parallel processing of large language models using reinforcement learning algorithms. *Management Strategies and Engineering Sciences*, pp. 1–15, 2025.

Giorgio Franceschelli and Mirco Musolesi. Reinforcement learning for generative ai: State of the art, opportunities and open research challenges. *Journal of Artificial Intelligence Research*, 79: 417–446, 2024.

Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *International conference on machine learning*, pp. 1666–1675. PMLR, 2018.

Zigang Geng, Yibing Wang, Yeyao Ma, Chen Li, Yongming Rao, Shuyang Gu, Zhao Zhong, Qinglin Lu, Han Hu, Xiaosong Zhang, et al. X-omni: Reinforcement learning makes discrete autoregressive image generative models great again. *arXiv preprint arXiv:2507.22058*, 2025.

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.

Yikun Jiang, Huanyu Wang, Lei Xie, Hanbin Zhao, Hui Qian, John Lui, et al. D-llm: A token adaptive computing resource allocation strategy for large language models. *Advances in Neural Information Processing Systems*, 37:1725–1749, 2024.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yanyu Li. *Accelerating large scale generative ai: A comprehensive study*. PhD thesis, Northeastern University, 2024.

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. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Zichen Miao, Jiang Wang, Ze Wang, Zhengyuan Yang, Lijuan Wang, Qiang Qiu, and Zicheng Liu. Training diffusion models towards diverse image generation with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10844–10853, 2024.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

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.

Subham S Sahoo, Aaron Gokaslan, Chris De, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. *Advances in Neural Information Processing Systems*, 37:105730–105779, 2024.

Suzuki Sakurama and Zoe Brian. Making diffusion models practical: A survey on acceleration and optimization. 2025.

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

Antonio Sclocchi, Alessandro Favero, and Matthieu Wyart. A phase transition in diffusion models reveals the hierarchical nature of data. *Proceedings of the National Academy of Sciences*, 122(1): e2408799121, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Kai Wang, Mingjia Shi, Yukun Zhou, Zekai Li, Zhihang Yuan, Yuzhang Shang, Xiaojiang Peng, Hanwang Zhang, and Yang You. A closer look at time steps is worthy of triple speed-up for diffusion model training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12934–12944, 2025.

Chao Wu, Baoheng Li, Mingchen Gao, and Zhenyi Wang. From efficiency to adaptivity: A deeper look at adaptive reasoning in large language models. *arXiv preprint arXiv:2511.10788*, 2025.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Shihao Yuan, Yahui Liu, Yang Yue, Jingyuan Zhang, Wangmeng Zuo, Qi Wang, Fuzheng Zhang, and Guorui Zhou. Ar-grpo: Training autoregressive image generation models via reinforcement learning. *arXiv preprint arXiv:2508.06924*, 2025.

Beibei Zhang, Wenjing Liu, Xiang Chang, Changjing Shang, Qiang Shen, and Fei Chao. Optimizing time-step sampling probabilities in diffusion models for enhanced training efficiency. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2025. doi: 10.1109/IJCNN64981.2025.11227426.

Tianyi Zheng, Cong Geng, Peng-Tao Jiang, Ben Wan, Hao Zhang, Jinwei Chen, Jia Wang, and Bo Li. Non-uniform timestep sampling: Towards faster diffusion model training. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 7036–7045, 2024.

Tianyi Zheng, Jiayang Zou, Peng-Tao Jiang, Hao Zhang, Jinwei Chen, Jia Wang, and Bo Li. Bidirectional beta-tuned diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(1):359–373, 2026. doi: 10.1109/TPAMI.2025.3604039.

A Appendix

A.1 State Representation Details

At iteration k, the ReiNA state is computed from signals available after the previous RL iteration:

$$s_k = [k/K_{\max},\, \ell_k,\, g_k,\, f_k,\, h_k]^\top .$$

The progress term $k/K_{\max}$ is a normalized scalar indicating the model's position in the training lifecycle.

The loss descriptor ℓ_k is computed from a sliding window of recent losses $\mathbf{L}_{k-W:k-1}$. It contains summary statistics such as the mean, variance, and linear trend, where the trend is estimated by least-squares fitting. This term captures both recent performance and short-term stability.

The gradient descriptor is defined as $g_k = \left\| \nabla_\theta \mathcal{L}\big(\theta_{k-1}; x_t^{(k-1)}, \gamma_{k-1}(t)\big) \right\|_2$. It measures the magnitude of the previous diffusion update and serves as a proxy for local optimization difficulty.

The batch feature descriptor f_k is extracted from the clean batch $\boldsymbol{x}_0^{(k-1)}$ using a lightweight feature extractor, such as the first two convolutional blocks of ResNet-18 followed by global average pooling. Because the policy must commit to a_k before observing $x_0^{(k)}$, f_k serves as a one-step-lagged proxy for batch difficulty; under random shuffling, consecutive batches share similar distributional statistics, so f_k provides an informative signal about the forthcoming batch composition.

The action-history descriptor h_k embeds the past $\mathcal{M}$ selected actions through a learnable lookup table, allowing the policy to retain temporal context and avoid unstable oscillatory decisions.

A.2 Detailed Boomerang Reward

The boomerang reward supervises the ReiNA policy by measuring whether a forward noising action returns through the reverse denoising path as a useful learning signal. At iteration k:

$$\mathcal{R}_{\text{boom}}(k) = \sum_{i \in \mathcal{P}} \lambda_i \, R_i(k) \; - \; \sum_{j \in \mathcal{N}} \lambda_j \, R_j(k), \tag{13}$$

where $\mathcal{P} = \{\text{cyc}, \text{align}, \text{learn}, \text{temp}\}$ and $\mathcal{N} = \{\text{cost}, \text{smooth}\}$. The coefficients λ control relative preference.

Cycle consistency. This term asks whether the selected forward corruption can close the diffusion forward-reverse cycle. A lightweight preview denoiser D_ω predicts the clean sample from the current noisy sample x_t, the realized conditioning value $\gamma(t)$, and a compact batch embedding e:

$$R_{\text{cyc}} = -\|D_\omega(x_t, \gamma(t), e) - x^0\|_2^2.$$

A smaller reconstruction error yields a higher reward, encouraging the policy to choose noise levels that are informative but still recoverable by the current reverse model.

Residual-level alignment. We divide diffusion time or SNR into B bins and maintain an exponential moving average of the denoising residual e_b in each bin. The policy-induced loss weight $w_k(t) = 2\sigma_k'(t)/\sigma_k(t)^3$ is aggregated into the same bins as $p_b^{(k)}$. ReiNA rewards schedules that emphasize under-learned noise levels:

$$R_{\text{align}} = \left\langle \text{softmax}(p^{(k)}), \text{softmax}(e/\tau) \right\rangle.$$

This term directly connects the action to the main goal of ReiNA, which is reallocating training weight toward the noise levels that remain poorly learned.

Learnability boundary. Relying solely on alignment may favor noise levels that are too corrupted to learn. We therefore reward noise choices whose normalized residual lies near a target learnability level e_{tar}:

$$R_{\text{learn}} = -\left|\bar{e}_{\mathcal{B}_k}(k) - e_{\text{tar}}\right|.$$

This prevents the policy from collapsing to trivially easy low-noise regions or destructively hard high-noise regions.

Temporal return. To capture the delayed effect of a noising decision, we measure whether the residual of the action-emphasized bins decreases after a short horizon H:

$$R_{\text{temp}}(k) = \text{clip}\left(\bar{e}_{\mathcal{B}_k}(k) - \bar{e}_{\mathcal{B}_k}(k+H), -c_{\text{temp}}, c_{\text{temp}}\right).$$

The validation residuals are computed on a small fixed mini-set, making this term a low-cost proxy for whether the action has led to useful training progress.

Regularization. Two penalties keep the controller well-behaved. The noise cost discourages excessive average corruption,

$$R_{\text{cost}} = \bar{\sigma}_k, \qquad \bar{\sigma}_k = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \sigma_k(t),$$

and the smoothness cost penalizes abrupt policy changes,

$$R_{\text{smooth}} = \|a_k - a_{k-1}\|_2^2.$$

Together, they avoid unstable oscillatory schedules while preserving the adaptive behavior needed for residual alignment.

A.3 PPO Optimization Details

To attribute the delayed boomerang signal to preceding decisions, we define the cumulative return for RL iteration i within a PPO rollout of length C as:

$$G_i = \sum_{j=i}^{i+C-1} \gamma_{\text{RL}}^{j-i} \mathcal{R}_{\text{boom}}(j), \tag{14}$$

where $\gamma_{\text{RL}} = 0.99$ is the discount factor. The advantage is $\hat{A}_i = G_i - V_\xi(s_i)$, estimated via a learned value network V_ξ.

The ReiNA policy is optimized using PPO over the raw continuous action space. Let $a_i^{raw} = (u_i, v_i)$ denote the action sampled from the Gaussian policy at state s_i. The probability ratio between the current and previous policies is

$$r_i(\psi) = \frac{\pi_\psi(a_i^{raw} \mid s_i)}{\pi_{\psi_{\text{old}}}(a_i^{raw} \mid s_i)} = \frac{\mathcal{N}(a_i^{raw} \mid \mu_\psi(s_i), \Sigma_\psi(s_i))}{\mathcal{N}(a_i^{raw} \mid \mu_{\psi_{\text{old}}}(s_i), \Sigma_{\psi_{\text{old}}}(s_i))}. \tag{15}$$

Given the advantage estimate $\hat{A}_i$, PPO maximizes the clipped surrogate objective

$$\mathcal{L}^{\text{CLIP}}(\psi) = \mathbb{E}_i \left[\min \left(r_i(\psi)\hat{A}_i, \text{clip}\left(r_i(\psi), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}\right) \hat{A}_i \right) \right], \tag{16}$$

where $\epsilon_{\text{clip}} = 0.2$ controls the trust region size. The clipping term prevents excessively large policy updates by optimizing the lower bound between the unclipped and clipped objectives.

The value network is optimized with a squared regression loss against the trajectory-level return:

$$\mathcal{L}^{VF}(\xi) = \mathbb{E}_i \left[\left(V_\xi(s_i) - G_i \right)^2 \right]. \tag{17}$$

Together, $\mathcal{L}^{\text{CLIP}}$ and $\mathcal{L}^{VF}$ update the policy and value networks using discounted trajectories of the boomerang reward assigned through G_i.

A.4 ReiNA Hyperparameters

Table 4 lists the complete hyperparameter configuration used for all ReiNA experiments ("full" ablation variant).

A.5 Hardware and Software

All experiments are conducted on a single NVIDIA A800-SXM4-80GB GPU (Driver 570.86.10, CUDA 12.8) with PyTorch 2.11.0+cu128. Each method–dataset combination runs independently on one GPU.

A.6 Detailed Derivation of the Loss Function

The ReiNA controls a cumulative corruption curve, and the training objective used in Eq. 12 is the continuous-time limit of the same marginal noising family.

Table 4: ReiNA hyperparameter configuration.

Component	Setting
Policy network	
Action dimension	2 (slope m_k, inflection point n_k)
$\sigma_{\min}$, $\sigma_{\max}$	10^{-3}, $1-10^{-3}$ (fixed)
Output mapping	Eq. 7; (m_k, n_k) control shape only
Hidden / feature dim	128 / 64
Policy / value / preview LR	1×10^{-4} (Adam)
Warmup	5 000 diffusion steps
PPO	
Clipping threshold ϵ_{clip}	0.2
Update epochs per rollout	4
Update interval C	128 steps
GAE λ	0.95
Discount γ_{RL}	0.99
Boomerang reward weights	
λ_{cyc} (cycle consistency)	0.4
λ_{align} (residual alignment)	0.3
λ_{learn} (learnability boundary)	0.2
λ_{temp} (temporal return)	0.1
λ_{cost} (noise cost)	0.05
λ_{smooth} (action smoothness)	0.05
Residual-level tracking	
SNR / timestep bins B	10
Residual EMA ρ	0.95
Learnability target e_{tar}	0.4
Learnability range	$[0.25, 0.6]$
Temporal return	
Temporal horizon H	128 steps
Validation residual mini-set	2 048 samples
Clipping threshold c_{temp}	1.0
Reward normalization	
Component scaling	Running z-score
Final FID	Evaluation only, not a training reward
Preview features	
Batch embedding	Lightweight feature extractor
Optional semantic feature	CLIP ViT-B/32 for visual datasets

For a fixed discrete DDPM schedule, the forward transition is

$$q(x_i|x_{i-1}) = \mathcal{N}\left(x_i; \sqrt{1-\beta_i}x_{i-1}, \beta_i I\right),$$

and its marginal distribution is

$$q(x_i|x_0) = \mathcal{N}\left(x_i; \sqrt{\bar{\alpha}_i}x_0, (1-\bar{\alpha}_i)I\right), \quad \bar{\alpha}_i = \prod_{j=1}^{i}(1-\beta_j).$$

Thus the marginal process is fully determined by the cumulative signal coefficient $\bar{\alpha}_i$, while the one-step variances β_i can be recovered from adjacent cumulative values.

In ReiNA, the policy parameterizes a differentiable cumulative noise standard deviation $\sigma_k(t) \in [\sigma_{\min}, \sigma_{\max}]$ over normalized time $t \in [0,1]$, with $\sigma_{\min} > 0$, $\sigma_{\max} < 1$, and $\sigma_k'(t) \geq 0$. For the derivation below, we omit the iteration index k and write $\sigma(t)$. We define

$$\bar{\alpha}(t) = 1 - \sigma(t)^2.$$

Because $\sigma(t)$ is nondecreasing and bounded below one, $\bar{\alpha}(t)$ is nonincreasing and remains positive. The marginal noising distribution is therefore

$$q_\sigma(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{1-\sigma(t)^2}\, x_0, \sigma(t)^2 I\right),$$

or equivalently

$$x_t = \sqrt{1-\sigma(t)^2}\, x_0 + \sigma(t)\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

This is the same marginal form as DDPM with $1-\bar{\alpha}(t) = \sigma(t)^2$. If the continuous curve is evaluated on a discrete grid $0 = t_0 < t_1 < \cdots < t_T = 1$, the equivalent effective DDPM transition variance is

$$\beta_i^{\text{eff}} = 1 - \frac{\bar{\alpha}(t_i)}{\bar{\alpha}(t_{i-1})} = 1 - \frac{1-\sigma(t_i)^2}{1-\sigma(t_{i-1})^2}.$$

Since $\bar{\alpha}(t_i) \le \bar{\alpha}(t_{i-1})$ and $\bar{\alpha}(t_{i-1}) > 0$, this gives $\beta_i^{\text{eff}} \in [0, 1)$. Consequently, the policy-controlled curve defines a valid monotone forward process and can be viewed either as direct marginal noising or as a discretized DDPM schedule.

We next derive the loss weight. Define the signal-to-noise ratio

$$V(t) = \frac{\bar{\alpha}(t)}{1-\bar{\alpha}(t)} = \frac{1-\sigma(t)^2}{\sigma(t)^2}.$$

The corresponding log signal-to-noise ratio is

$$\gamma(t) = \log V(t) = \log \frac{1-\sigma(t)^2}{\sigma(t)^2}.$$

Differentiating with respect to normalized time gives

$$V'(t) = \frac{d}{dt}\left(\sigma(t)^{-2} - 1\right) = -\frac{2\sigma'(t)}{\sigma(t)^3}.$$

For continuous-time diffusion objectives written under a uniform sampling density over t, the local ELBO contribution is weighted by the amount of SNR removed per unit time, namely $-V'(t)$. Therefore the unnormalized weight induced by the ReiNA schedule is

$$w(t) = -V'(t) = \frac{2\sigma'(t)}{\sigma(t)^3}.$$

Using the standard x_0-prediction parameterization,

$$\hat{x}_0(x_t, \gamma(t)) = \frac{x_t - \sigma(t)\epsilon_\theta(x_t, \gamma(t))}{\sqrt{1-\sigma(t)^2}},$$

the weighted objective becomes

$$\mathcal{L}_{\text{new}}(\theta) = \frac{1}{2}\mathbb{E}_{x_0,t,\epsilon}\left[w(t)\,\|x_0 - \hat{x}_0(x_t, \gamma(t))\|_2^2\right].$$

Substituting $w(t) = 2\sigma'(t)/\sigma(t)^3$ recovers Eq. 12.

Finally, we note that the apparent singularity at small $\sigma(t)$ is controlled by the fixed lower endpoint $\sigma_{\min} > 0$. In implementation, we use the normalized effective weight

$$\tilde{w}(t) = \frac{\text{clip}(w(t), 0, w_{\max})}{\mathbb{E}_{t'\sim\mathcal{U}(0,1)}[\text{clip}(w(t'), 0, w_{\max})]},$$

estimated over the sampled timesteps in the minibatch. This normalization preserves the relative emphasis across noise levels while absorbing the global scale into the learning rate, and clipping prevents rare low-σ samples from dominating a minibatch update. The adaptive policy therefore changes the shape of the training weighting over t, not the validity of the underlying forward diffusion process.

A.7 Architecture Details

Absolute FID/IS values are higher than fully-tuned SoTA in the literature because we deliberately use a compact U-Net ($\approx$35.7M parameters at 32×32) with a reduced training budget, no EMA, and no classifier-free guidance, so that the outer-loop RL and the full ablation suite remain affordable.

CIFAR-10 (32×32). The U-Net has four resolution levels with channel multipliers $[1, 2, 2, 2]$ and base channels 128, incorporating self-attention at 16×16. Two ResNet blocks are used per level. Sinusoidal embeddings of the log signal-to-noise ratio are projected to each block. Training uses Adam (β_1=0.9, β_2=0.999), batch size 128, for 800 epochs ($\approx$313K steps).

CelebA-HQ / LSUN Bedroom (256×256). To accommodate higher resolution, we expand the U-Net to six levels with channel multipliers $[1, 1, 2, 2, 4, 4]$ (base channels 128), self-attention at 32×32 and 16×16, and two ResNet blocks per level. This yields $\approx$113M parameters (vs. 35.7M at 32×32). Training uses Adam with the same hyperparameters, batch size 64, and gradient accumulation of 2 for an effective batch of 128 when memory permits. CelebA-HQ trains for 500 epochs ($\approx$210K steps); LSUN Bedroom trains for 50 epochs ($\approx$2.37M steps).

A.8 Preview Denoiser D_ω Details

The preview denoiser D_ω is a lightweight 4-layer convolutional network (channels: $64 \to 128 \to 128 \to 3$) with log signal-to-noise ratio embedding via sinusoidal positional encoding and a compact batch feature vector concatenated to the bottleneck. It has $\approx$0.5 M parameters (vs. 35.7 M for the main U-Net).

D_ω is trained *jointly* with the diffusion model. At each step, D_ω receives the same noised batch x_t and the same conditioning value $\gamma(t)$, then predicts $\hat{x}_0$. Its loss $\|D_\omega(x_t, \gamma(t), e) - x_0\|^2$ is optimized with a separate Adam optimizer (lr $= 10^{-4}$); gradients do *not* flow back into the RL policy. As the main model improves, D_ω tracks the improving data distribution, keeping the cycle-consistency reward calibrated throughout training.

A.9 Optional Semantic Features Across Resolutions

The preview denoiser can optionally use a frozen semantic feature extractor such as CLIP (Radford et al., 2021). For CIFAR-10 (32×32), images are resized to 224×224 via bilinear interpolation before feature extraction. Since low-resolution CLIP features can be noisy, the default reward relies on cycle consistency and residual-level statistics rather than a standalone CLIP similarity reward. For CelebA-HQ and LSUN Bedroom (256×256), images are center-cropped and resized to 224×224 with minimal information loss, making semantic embeddings more reliable as auxiliary preview features. This design keeps the main reward diffusion-native while still allowing high-level features to improve the calibration of D_ω on visual datasets.

A.10 Evaluation Protocol

- **Metric library:** `torch-fidelity` v0.4.0 (FID, IS, Precision, Recall).
- **Generated samples:** 10 000 per evaluation.
- **Reference sets:** CIFAR-10 test split (10 000 images); CelebA-HQ test split (3 000 images); LSUN Bedroom validation set (300 images, with pre-computed Inception statistics on 50K training images for FID).
- **Inception network:** InceptionV3 with `torch-fidelity` default weights.
- **Sampling:** 1000-step ancestral sampling for all methods (DDPM, VDM, NFDM, MuLAN, ReiNA). ReiNA uses the frozen final policy schedule to define $\gamma_{\text{gen}}(t)$.
- **Seeds:** $\{0, 1, 2\}$
- **Sample batch size:** 16 (for GPU memory management during evaluation; does not affect metric computation).
- **Precision/Recall:** k-nearest-neighbor manifold estimation (k=3) per `torch-fidelity` defaults.

A.11 Normalized Training Residual Convergence

See Figure 5.

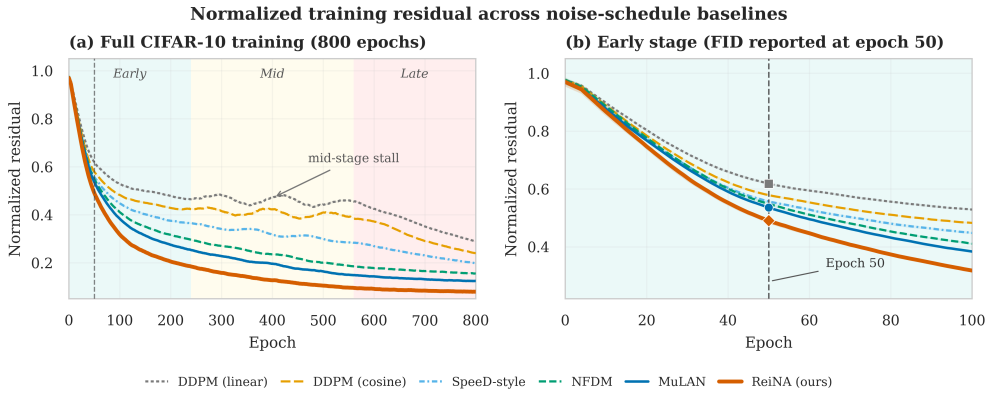


Figure 5: **Normalized training residual across noise-schedule baselines.** (a) Full CIFAR-10 training over 800 epochs. (b) Early-stage residual curves with the epoch-50 evaluation point marked. ReiNA reduces residual error faster than static and learned schedule baselines, and shows less mid-stage stalling.

A.12 Heuristic Adaptive Schedulers

We designed two diagnostic heuristic schedulers to test whether simple reactive rules can achieve adaptive scheduling:

- **Grad-React:** increases noise intensity proportional to the gradient norm when it exceeds a running-average threshold.
- **Loss-Trig:** increases noise when the training loss rises over a sliding window.

These methods are not from prior literature and serve as diagnostic baselines. Their original (aggressive) settings led to severe instability (FID > 100). We additionally report "tuned" variants with milder thresholds and blending with the base $\sigma(t)$:

Table 5 summarizes the results.

Table 5: Heuristic baselines on CIFAR-10.

Variant	FID↓	IS↑
Grad-React (original)	146.82	4.97
Grad-React (tuned)	13.47 ± 0.39	8.39 ± 0.07
Loss-Trig (original)	257.14	3.18
Loss-Trig (tuned)	15.29 ± 0.48	8.19 ± 0.08

Even after tuning, heuristic methods remain far below learned schedules, confirming that principled optimization (RL or gradient-based) is necessary for effective adaptive scheduling.

A.13 Reward Weight Sensitivity

We vary the main λ coefficients individually (setting each to 0, $0.5\times$, or $2\times$ default) while keeping others fixed, training for 200 epochs. Results are shown in Table 6.

The sensitivity is highest for λ_{cyc} and λ_{align}, because they respectively define recoverability and residual targeting. λ_{learn} and λ_{temp} should mainly affect stability and convergence speed, while λ_{cost} controls the average selected noise level.

Table 6: λ sensitivity for the boomerang reward (FID↓, mean $\pm$ std over 3 seeds, 200 epochs).

Weight	$0\times$	$0.5\times$	$2\times$
λ_{cyc}	8.89 ± 0.22	7.51 ± 0.14	7.86 ± 0.17
λ_{align}	8.28 ± 0.18	7.39 ± 0.13	7.61 ± 0.16
λ_{learn}	7.84 ± 0.18	7.26 ± 0.13	7.52 ± 0.15
λ_{temp}	7.67 ± 0.15	7.29 ± 0.13	7.21 ± 0.12
λ_{cost}	7.36 ± 0.16	7.35 ± 0.15	7.09 ± 0.12
Default (200 ep)		7.19 ± 0.13	

A.14 Batch-Adaptive Analysis Details

Stratification protocol. We compute the initial training loss $\ell_0(x)$ for each batch at its first encounter and partition all batches into tertiles (easy/mixed/hard) using global thresholds. These thresholds are fixed across stages; batches are *not* re-stratified per stage.

Non-linear fit. In addition to the linear R^2 reported in the main text, we fit a quadratic model $\sigma = a\ell_0^2 + b\ell_0 + c$. Per-stage results are shown in Table 7.

Table 7: Linear vs. quadratic fit of the loss–noise relationship by training stage.

Stage	R^2_{linear}	$R^2_{\text{quadratic}}$
Early	0.07	0.11
Mid	0.52	0.57
Late	0.87	0.92

The small gap between linear and quadratic R^2 suggests the loss–noise relationship is well-approximated by a linear model, particularly in the late stage.

A.15 Alignment Between Loss Weighting and Training Dynamics

Figure 6 examines whether ReiNA's adaptive loss weighting evolves consistently with training dynamics. Panel (a) shows the joint distribution of the loss weight $w(t)$ over diffusion timestep t and training progress. Early training assigns low and relatively broad weights, mid training expands the active weighting range, and late training shifts toward larger weights, especially at high-noise timesteps. This progression suggests that ReiNA increasingly emphasizes challenging high-variance regions as the model matures.

Panel (b) further compares the stage-specific profiles of $w(t)$ with a static cosine schedule. The Early-stage curve remains nearly flat and low, the Mid-stage curve introduces moderate decay, and the Late-stage curve becomes steep and power-law-like, particularly for $t > 0.1$. Notably, the Late-stage profile approaches and slightly exceeds the static cosine baseline at high-noise timesteps, indicating that ReiNA adaptively strengthens loss weighting where stronger regularization is most useful. The consistent shift across both panels shows that ReiNA's weighting strategy is not arbitrary, but systematically aligns the optimization objective with the evolving generative learning process.

A.16 Full Ablation and Sensitivity Analysis

A.16.1 Reward Component Ablation

Rows marked with $*$ are summarized in the main text (Table 3); remaining rows are reported here for completeness.

The full ablation separates the two essential claims of ReiNA. R_{cyc} tests whether noising remains recoverable, while R_{align} tests whether the learned weight profile actually follows under-learned noise levels. R_{temp} adds a low-cost delayed residual signal for credit assignment, and the remaining terms regularize the controller into a stable operating region.

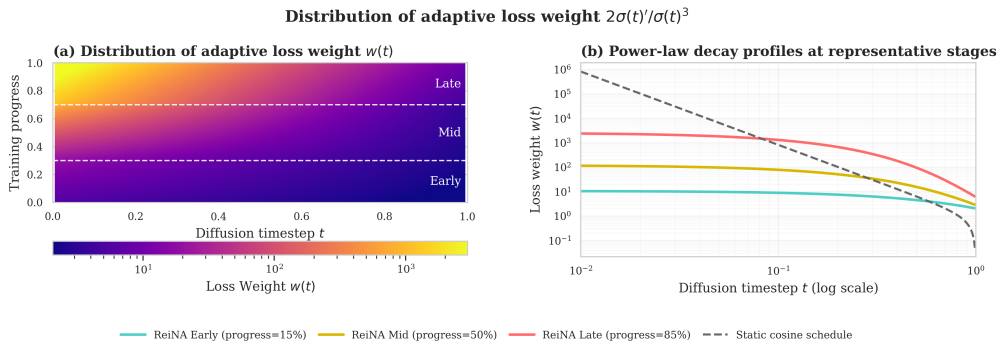


Figure 6: **Alignment between adaptive loss weighting and training dynamics.** (a) Distribution of the adaptive loss weight $w(t)$ across diffusion timestep t and training progress. (b) Stage-specific decay profiles of $w(t)$ compared with a static cosine schedule. ReiNA progressively shifts from low, broad weighting to stronger high-noise emphasis in the Late stage.

Table 8: Full reward component ablation on CIFAR-10 (mean $\pm$ std over 3 seeds).

Variant	FID↓	IS↑	Prec.↑	Recall↑	ΔFID
ReiNA (full)*	6.49 ± 0.07	9.78 ± 0.03	0.65 ± 0.02	0.57 ± 0.02	–
w/o R_{cyc}*	7.50 ± 0.16	9.14 ± 0.06	0.58 ± 0.02	0.49 ± 0.01	+1.01
w/o R_{align}*	7.16 ± 0.12	9.33 ± 0.04	0.60 ± 0.01	0.51 ± 0.02	+0.67
w/o R_{temp}*	7.31 ± 0.15	9.18 ± 0.06	0.59 ± 0.02	0.50 ± 0.03	+0.82
w/o R_{learn}	6.90 ± 0.10	9.52 ± 0.04	0.62 ± 0.01	0.53 ± 0.01	+0.41
w/o R_{cost}	6.76 ± 0.11	9.65 ± 0.05	0.63 ± 0.03	0.54 ± 0.02	+0.27
w/o R_{smooth}	6.61 ± 0.08	9.72 ± 0.03	0.64 ± 0.02	0.55 ± 0.01	+0.12

A.16.2 State Feature Ablation

Table 9 reports the impact of removing individual state components. Removing the loss descriptor ℓ_k causes the largest degradation, followed by the batch feature f_k, confirming that both optimization and data signals are essential for informed scheduling decisions.

Table 9: State feature ablation (full reward, CIFAR-10; mean $\pm$ std over 3 seeds).

State config	FID↓	IS↑	Prec.	Recall
Full state	6.49 ± 0.07	9.78 ± 0.03	0.65 ± 0.02	0.57 ± 0.02
w/o loss ℓ_k	7.04 ± 0.15	9.30 ± 0.06	0.59 ± 0.01	0.50 ± 0.03
w/o gradient g_k	6.73 ± 0.10	9.62 ± 0.03	0.64 ± 0.01	0.56 ± 0.01
w/o batch feat. f_k	6.96 ± 0.12	9.38 ± 0.04	0.60 ± 0.03	0.51 ± 0.01
w/o history h_k	6.78 ± 0.13	9.54 ± 0.05	0.61 ± 0.01	0.53 ± 0.03
Progress only	7.46 ± 0.17	9.04 ± 0.06	0.57 ± 0.03	0.48 ± 0.01

A.16.3 Action Space Ablation

Table 10 compares the full 2D action (slope + inflection point) with 1D variants. Both dimensions contribute, with the inflection-point-only variant suffering the largest FID gap, indicating that controlling where the schedule concentrates is more limited without slope adjustment.

A.16.4 Checkpoint Interval Sensitivity

Table 11 reports sensitivity to the PPO update interval C. The default $C{=}128$ achieves the best FID; shorter intervals ($C{=}64$) increase policy update noise, while longer intervals ($C{=}256, 512$) delay credit assignment and degrade performance.

Table 10: Action space ablation (full reward + full state; mean ± std over 3 seeds).

Action config	FID↓	IS↑
2D: slope + inflection point (default)	6.49 ± 0.07	9.78 ± 0.03
1D: slope only	6.93 ± 0.12	9.40 ± 0.04
1D: inflection point only	7.20 ± 0.16	9.18 ± 0.06

Table 11: Checkpoint interval C sensitivity (mean ± std over 3 seeds).

C	FID↓	IS↑
64	6.69 ± 0.13	9.59 ± 0.05
128 (default)	6.49 ± 0.07	9.78 ± 0.03
256	6.60 ± 0.09	9.67 ± 0.03
512	6.86 ± 0.15	9.46 ± 0.04

A.17 VDM Objective Comparison

We evaluate VDM under both its native variational lower bound (VLB) objective and the unified MSE-based x_0-prediction objective shared by all other baselines. Under VLB, VDM improves to FID 11.84 (from 12.63 under MSE), confirming that objective mismatch was partly responsible for its poor absolute performance. Nevertheless, VDM remains substantially behind schedule-learning methods (NFDM, MuLAN, ReiNA), likely because its single-parameter monotone schedule is far less expressive than the policy-controlled sigmoid family used by ReiNA. SpeeD-style timestep reweighting improves marginally over DDPM-linear (FID 8.91 vs. 10.35), consistent with its design focus on sampling efficiency rather than schedule shape.

A.18 Training Overhead

Table 12 reports per-step wall-clock time and peak GPU memory for each method on both resolutions.

Table 12: Training overhead per method on CIFAR-10 (32×32) and CelebA-HQ (256×256). All timings measured on a single NVIDIA A800-SXM4-80GB with identical batch size per dataset.

	CIFAR-10 (32×32)			CelebA-HQ (256×256)		
Method	sec/step	GPU (GB)	×DDPM	sec/step	GPU (GB)	×DDPM
DDPM	0.12	5.2	1.00×	0.52	21.4	1.00×
NFDM	0.13	5.4	1.08×	0.54	22.1	1.04×
MuLAN	0.15	5.8	1.25×	0.58	23.2	1.12×
SpeeD	0.12	5.2	1.00×	0.52	21.4	1.00×
ReiNA	0.18	7.5	1.50×	0.68	28.6	1.31×

The overhead arises from the RL policy and value networks (≈0.8M parameters), the preview denoiser D_ω (≈0.5M), and a frozen CLIP forward pass. Because these components have fixed cost independent of image resolution, the relative overhead decreases from 1.50× (CIFAR-10) to 1.31× (CelebA-HQ) as the U-Net forward–backward pass dominates.

A.19 High-Resolution Ablation Verification

To verify that key ablation findings transfer from CIFAR-10 to higher resolution, we use the same boomerang reward ablation protocol on CelebA-HQ (256×256, 3 seeds, 500 epochs).

The reward-component ranking transfers to high resolution: removing R_{cyc} degrades FID the most (+1.28), followed by R_{temp} (+1.06) and R_{align} (+0.91), indicating that recoverability and delayed residual feedback remain the most critical factors when visual structure is richer and the controller's relative overhead is lower. The three regularization-oriented terms induce smaller but consistent

Table 13: Boomerang reward ablation on CelebA-HQ 256×256 (mean $\pm$ std over 3 seeds).

Variant	FID↓	IS↑	Prec.↑	Recall↑
ReiNA (full)	9.94 ± 0.16	3.46 ± 0.05	0.69 ± 0.02	0.52 ± 0.02
w/o R_{cyc}	11.22 ± 0.23	3.08 ± 0.08	0.62 ± 0.01	0.43 ± 0.03
w/o R_{align}	10.85 ± 0.19	3.22 ± 0.06	0.64 ± 0.03	0.46 ± 0.01
w/o R_{temp}	11.00 ± 0.22	3.14 ± 0.08	0.63 ± 0.01	0.44 ± 0.03
w/o R_{learn}	10.36 ± 0.17	3.31 ± 0.05	0.65 ± 0.03	0.48 ± 0.01
w/o R_{cost}	10.20 ± 0.19	3.35 ± 0.07	0.66 ± 0.01	0.49 ± 0.03
w/o R_{smooth}	10.05 ± 0.16	3.42 ± 0.05	0.68 ± 0.02	0.51 ± 0.01

degradations (R_{learn} $+0.42$, R_{cost} $+0.26$, R_{smooth} $+0.11$), matching the ordering observed on CIFAR-10 and showing that the full reward remains beneficial even at high resolution.

A.20 StructSim Dataset

StructSim is a materials configuration corpus constructed from supervised ANSYS CFX files written in a script language for physical simulation, paired with DOML schemas. Rather than generating an entire setup file, we extract physically coherent boundary subtrees of type `MASS_AND_MOMENTUM`, which encode inlet/outlet/wall mass–momentum settings commonly used in CFD case configuration.

Sample definition. A block is a root script node together with all descendant nodes enabled by a structural decision. Each training example contains: (i) a pruned DOML subtree as the generation condition; (ii) context such as `BoundaryType`; and (iii) the target script span. Unselected DOML branches are removed from both input and target so that disabled children (e.g., unused `FLOW_DIRECTION`) never appear as supervision.

Scale and heterogeneity. The pilot corpus contains 899 `MASS_AND_MOMENTUM` samples, of which $\approx$397 enable a flow-direction child and $\approx$502 do not. Condition keys mix boundary role and child enablement, e.g., `WALL+NONE`, `OUTLET+NONE`, and `INLET+FLOW_DIRECTION`. Option vocabularies follow CFX priors (static pressure, no-slip wall, normal-speed / total-pressure inlets). Shallow fields are packed into an 18-D vector (option one-hots, standardized numerics, child options) for diffusion training and decoded back to nested `KEYWORD`/`END` text.

Continuous relaxation and codec. StructSim places discrete options and optional child blocks behind a fixed codec so that training stays inside the same continuous Gaussian process used for images. The codec encodes each block as an 18-dimensional vector whose coordinates store option one-hots, standardized numeric fields, and shallow child indicators. During training, ReiNA and all schedule baselines construct noisy latents from this vector by the same rule as Eq. 8, and the policy continues to control only the monotone schedule $\sigma_k(t)$ together with the induced loss weight $w(t) = 2\sigma_k'(t)/\sigma_k(t)^3$. After sampling, the codec maps the denoised vector back to nested `KEYWORD`/`END` text by selecting the nearest option, restoring numeric fields, and emitting child blocks only when the reconstructed coordinates indicate that they are enabled. Structural metrics in Table 2 are evaluated exclusively after this decoding step. In this design, discreteness is confined to the encode and decode maps, while adaptive noising operates on continuous coordinates. The StructSim results therefore speak to transfer of residual-aligned Gaussian scheduling under heterogeneous structural difficulty, not to a native discrete or absorbing forward process, which is left outside the present study.

Evaluation protocol. We hold out a fixed test split and compare noise controllers under an identical 1D U-Net backbone, codec, and boundary prior. Metrics are computed after decoding: OptAcc (root option match under the queried condition), ChildCons (enabled-child set agrees with generated subtree presence), FieldF1 (required numeric fields), and Hard (ChildCons on `INLET+FLOW_DIRECTION` only). For ReiNA, the temporal return uses these structural residual scores in place of image residuals, while the cycle, alignment, learnability, and regularization terms follow the main method.

A.21 Additional Qualitative Results

Figure 7 provides additional LSUN Bedroom comparisons under matched initial noise, and Figure 8 shows uncurated CelebA-HQ samples generated by ReiNA.



Figure 7: **Qualitative comparison on LSUN Bedroom.** Each triple is generated from the same initial noise x_T, using random seed selection. Left: DDPM (linear). Middle: DDPM (cosine). Right: ReiNA (ours). ReiNA yields cleaner layout, more stable lighting, and fewer texture artifacts than the two static schedules, consistent with the FID ranking in Table 1.



Figure 8: **Qualitative results on CelebA-HQ.** Samples generated by ReiNA on CelebA-HQ (256×256). ReiNA produces photorealistic faces with coherent identity and lighting, consistent with the FID ranking in Table 1.

A.22 LLM Usage

We used large language models for light copy-editing and grammar checking. All scientific content, experimental design, and claims were written and verified by the authors.