NOISERATER: META-LEARNED NOISE VALUATION FOR DIFFUSION MODEL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training a diffusion model involves two sources of randomness for each data sample: the timestep and the Gaussian noise realization. The timestep has been studied extensively through scheduling and weighting, whereas the impact of the noise realization at a given timestep is still underexplored. In this work, we examine whether different noise instances are equally informative. We introduce *NoiseRater*, a network that scores an individual noise instance conditioned on the data sample and timestep. The rater is learned through bilevel optimization, where its scores reweight the diffusion loss in the inner loop and it is updated to reduce validation loss after the inner-loop updates. Using the trained rater to select training noise, we observe three properties of training noise. First, noise realizations at the same timestep are not equally useful, with the rater's top-scored noise improving performance over i.i.d. sampling and its bottom-scored noise degrading it. Second, this utility is contextual, depending jointly on the image, the class, and the timestep. Third, noise selection is complementary to timestep-level design, retaining most of its gain when combined with existing scheduling and weighting schemes. These findings establish instance-level noise valuation as a new axis for understanding and improving diffusion training. Code is available at `https://anonymous.4open.science/r/NoiseRater-8ECB`.

1 INTRODUCTION

Diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2020b) have emerged as a dominant paradigm for generative modeling (Polyak et al., 2024; Rombach et al., 2022; Watson et al., 2023). Recently, noise has increasingly become a central object of control at inference time. Methods focusing on test-time compute scaling (He et al., 2025; Kim et al., 2025; Ren et al., 2025) and noise trajectory manipulation (Ahn et al., 2024; Li et al., 2025; Singhal et al., 2025) demonstrate that the specific magnitude, structure, and evolution of noise critically shape the final generation quality. In contrast to this fine-grained control at inference time, standard training treats the choice of noise as largely indiscriminate.

During standard training, the diffusion objective relies on two independent sources of stochasticity for any given data sample: the noise level (timestep) and the noise realization itself. While the former has been extensively optimized through timestep scheduling and weighting (Choi et al., 2022; Esser et al., 2024; Hang et al., 2023; 2025; Karras et al., 2022; Kim et al., 2024; Raya et al., 2026; Sun and Shi, 2026), the latter remains largely overlooked. Once the timestep is fixed, the noise is drawn independently from a standard Gaussian, and every realization contributes equally to the loss. This implicitly assumes that all noise realizations are equally useful for learning, leaving fine-grained *instance-level* noise variation unexplored.

This raises a fundamental question: ***are all noise realizations equally useful for learning diffusion models?*** More specifically, even at the same timestep, different noise instances may carry varying levels of learning signal. Some may provide clear supervision for denoising, while others may be ambiguous, redundant, or less informative for optimization. If so, uniformly treating noise during training may lead to suboptimal results.

In this work, we explore the problem of instance-level noise valuation during training using meta-learning. Concretely, we introduce *NoiseRater*, a parametric function that assigns a score to each noise instance conditioned on the data sample and timestep. These scores are used to reweight the

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.

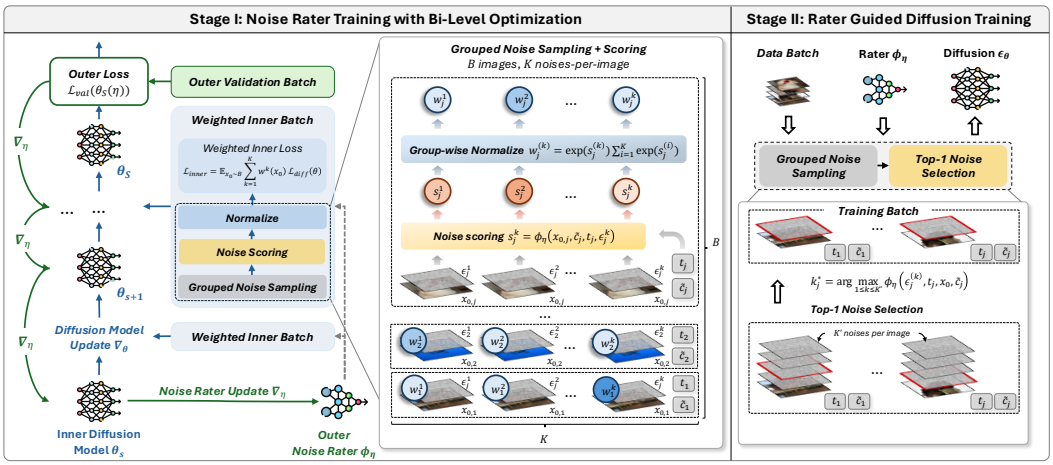

Figure 1: Illustration of our meta-learned noise valuation approach for diffusion training. We first train a noise rater using bilevel optimization, and then guide diffusion training with the rater's scores.

diffusion loss, allowing the model to focus on more informative noise while downweighting less useful ones. To train the NoiseRater, we formulate a bilevel optimization framework (Engstrom et al., 2025; Maclaurin et al., 2015): the diffusion model is trained using weighted noise samples in the inner loop, while the rater is optimized in the outer loop to minimize validation loss. This setup enables the rater to identify the contribution of distinct noise instances to generalization performance, rather than relying on heuristic criteria.

By integrating NoiseRater into the training pipeline, we conduct a comprehensive empirical study of how instance-level noise affects diffusion learning, yielding three key insights about the utility of noise realizations: 1) *Not equally useful.* Even for the same data sample at the same timestep, different Gaussian realizations carry different learning signals. Selecting the rater's top-scored candidate improves over the uniformly random choice made by standard training, and selecting its bottom-scored candidate falls below it at the same per-step cost. 2) *Context-dependent.* The utility of a noise realization is not an intrinsic property of the noise itself. Rather, it depends jointly on the data sample, the timestep, and the training stage: proxies computed from the noise alone, such as magnitude-based selection, capture only a fraction of the attainable gain, and the rater's advantage vanishes when its conditioning on timestep is removed. 3) *Complementary.* Instance-level noise selection addresses an axis largely distinct from timestep-level design: combined with established timestep weighting and scheduling schemes, it retains most of its standalone improvement. Our contributions are summarized as follows:

- We formalize instance-level noise valuation and introduce *NoiseRater*, a meta-learned rater that measures the utility of a noise instance and can be used as a noise selector to improve diffusion training.
- We show that noise realizations at the same timestep are not equally useful. Selecting the rater's top-scored candidate improves over i.i.d. sampling, and selecting its bottom-scored candidate falls below it.
- We show that noise utility is context-dependent. It depends jointly on the data sample and timestep rather than on the noise alone.
- We show that noise selection is complementary to timestep-level design, retaining most of its gain on top of existing weighting and scheduling schemes.

2 Preliminaries: Diffusion Models

Diffusion models define a generative process by learning to reverse a gradual corruption of data into noise. In the standard formulation, a forward noising process incrementally perturbs a clean data sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ into a sequence of increasingly noisy latent variables $\{\mathbf{x}_t\}_{t=1}^{T}$.

Forward process. The forward diffusion process is a Markov chain that adds Gaussian noise at each timestep $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\big(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}\big)$, $t = 1, \ldots, T$, where $\{\beta_t\}_{t=1}^T$ is a predefined variance schedule controlling the noise magnitude at each step. Intuitively, this process gradually destroys the structure in $\mathbf{x}_0$, eventually producing a nearly isotropic Gaussian distribution.

A key property of this construction is that the marginal distribution at any timestep t admits a closed-form expression $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\big)$, where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. This expression allows us to directly sample $\mathbf{x}_t$ from $\mathbf{x}_0$ without simulating the full Markov chain, which is crucial for efficient training.

Reverse process and training objective. The generative model is trained to invert this noising process. Instead of directly parameterizing the reverse transition $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, it is common to train a neural network $\epsilon_\theta(\mathbf{x}_t, t)$ to predict the noise that was added to produce $\mathbf{x}_t$. Specifically, given a noisy sample $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, the model is trained using a simple mean squared error (MSE) objective:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{t, \mathbf{x}_0, \epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2\right]. \tag{1}$$

Training proceeds by sampling triples $(\mathbf{x}_0, t, \epsilon)$, constructing $\mathbf{x}_t$ using the closed-form expression above, and minimizing the discrepancy between the true noise ϵ and the predicted noise $\epsilon_\theta(\mathbf{x}_t, t)$.

Conditional training and classifier-free guidance. In conditional diffusion models, the denoising network additionally takes as input a condition $c \sim p(c)$ (e.g., text prompt), and is trained to predict noise as $\epsilon_\theta(\mathbf{x}_t, t, c)$. Following classifier-free guidance (CFG) (Ho and Salimans, 2022), we randomly drop the condition during training with probability p_{drop}, replacing c with a null condition $\varnothing$. This yields a unified model that supports both conditional and unconditional predictions as $\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, c}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \tilde{c})\|^2\right]$, where $\tilde{c} \in \{c, \varnothing\}$ denotes the possibly dropped condition.

From DDPM to DDIM. While the above formulation follows the standard DDPM (Ho et al., 2020), this work implements the DDIM framework (Song et al., 2020a). DDIM provides a deterministic (or partially stochastic) non-Markovian sampling procedure that shares the same training objective but enables faster and more flexible generation. Importantly, our modifications operate entirely at the level of the training loss and are therefore directly compatible with DDIM without altering its sampling dynamics.

3 Method: Meta-Learned Noise Rater

A central observation about standard diffusion training is that all noise samples are treated equally when optimizing Equ. 1. We hypothesize that not all noise realizations contribute equally to learning: *some may be more informative, while others may be redundant or even detrimental*. To explore this, we introduce a parametric *NoiseRater* $\phi_\eta(\epsilon, t, \mathbf{x}_0, \tilde{c}) : \mathcal{E} \times [0,1] \times \mathcal{X} \times \mathcal{C} \to \mathbb{R}$, which assigns a scalar score to each noise instance conditioned on the clean image $\mathbf{x}_0$, the condition $\tilde{c}$, and the timestep t. We adapt meta-learning (Shu et al., 2019) to train the rater so that its scores reflect the relative importance of the training signal induced by each noise instance.

Our method has two stages (Figure 1). In the *rater training* stage, the rater scores each noise instance and these scores reweight the diffusion loss (Section 3.1); the rater itself is updated through a bilevel meta-objective defined on the resulting diffusion model (Section 3.2). In the *diffusion training* stage, we freeze the rater, sample a pool of candidate noises for each image–timestep pair, and use the rater's scores to select the best noise (Section 3.3).

3.1 Noise Weighting

Grouped Noise Sampling. To ensure that the noise rater focuses on differences between noise realizations rather than image content, we construct each minibatch B as groups of samples that share the same underlying clean image. Specifically, for each $\mathbf{x}_0$, $\tilde{c}$ and t, we draw K independent noises $\{\epsilon^{(k)}\}_{k=1}^K$ (e.g., $K = 4$), producing a set of noisy inputs $\{\mathbf{x}_t^{(k)}\}_{k=1}^K$.

We then apply a *group-wise normalization* to these K instances associated with the same $\mathbf{x}_0$. The rater assigns scores $\{\phi_\eta(\epsilon^{(k)}, t, \mathbf{x}_0, \tilde{c})\}_{k=1}^K$, which are normalized via a softmax within the group:

$$w^{(k)}(\mathbf{x}_0, \tilde{c}, t) = \frac{\exp(\phi_\eta(\epsilon^{(k)}, t, \mathbf{x}_0, \tilde{c}))}{\sum_{k'=1}^K \exp(\phi_\eta(\epsilon^{(k')}, t, \mathbf{x}_0, \tilde{c}))}. \tag{2}$$

This normalization enforces that weights are assigned based on the relative importance of noise realizations under a fixed condition $(\mathbf{x}_0, \tilde{c}, t)$, preventing the rater from exploiting variations in image content or conditioning across the batch B.

Rater-guided Training Objective. The standard diffusion loss is modified into a group-wise weighted objective:

$$\mathcal{L}_{\text{inner}}(\theta; \eta) = \mathbb{E}_{\mathbf{x}_0 \sim B}\left[\sum_{k=1}^K w^{(k)}(\mathbf{x}_0, \tilde{c}, t) \cdot \|\epsilon^{(k)} - \epsilon_\theta(\mathbf{x}_t^{(k)}, t, \tilde{c})\|^2\right], \tag{3}$$

where $\{(\epsilon^{(k)}, \mathbf{x}_t^{(k)})\}_{k=1}^K$ are independently sampled noise instances associated with $(\mathbf{x}_0, \tilde{c}, t)$, and $w^{(k)}(\mathbf{x}_0, \tilde{c}, t)$ are the corresponding group-normalized weights. This formulation reweights training signals according to the relative importance of noise realizations conditioned on $(\mathbf{x}_0, \tilde{c}, t)$, emphasizing informative noise while avoiding spurious correlations with image content or conditioning.

3.2 Bilevel Meta-Objective

We frame the learning of the noise rater as a bilevel optimization problem (Calian et al., 2025), where the diffusion model ϵ_θ and the rater ϕ_η are optimized at different levels.

Inner optimization (`model training`). For a fixed noise rater ϕ_η, the diffusion model parameters θ are updated by performing multiple gradient steps on the weighted loss:

$$\theta^{(s+1)} \leftarrow \theta^{(s)} - \alpha \nabla_\theta \mathcal{L}_{\text{inner}}(\theta^{(s)}; \eta), \quad s = 0, \ldots, S-1, \tag{4}$$

where $S \geq 1$ denotes the number of inner-loop updates. We write $\theta^{(S)}(\eta)$ for the resulting parameters. This corresponds to standard diffusion training, except that each training signal associated with $\mathbf{x}_0$ is reweighted according to the learned importance scores $\{w^{(k)}(\mathbf{x}_0, \tilde{c}, t)\}_{k=1}^K$.

Outer optimization (`rater training`). The noise rater parameters η are updated to improve the downstream performance of the diffusion model after the inner updates:

$$J_S(\eta) = \mathcal{L}_{\text{val}}(\theta^{(S)}(\eta)), \quad \eta^* = \arg\min_\eta J_S(\eta), \tag{5}$$

where $\mathcal{L}_{\text{val}}$ is computed on held-out data. J_S is what Algorithm 1 optimizes; the idealized objective $J(\eta) = \mathcal{L}_{\text{val}}(\theta^*(\eta))$ with $\theta^* = \arg\min_\theta \mathcal{L}_{\text{inner}}$ is analyzed in Thm. 3.1 and can be viewed as the $S \to \infty$ limit.

Conceptually, the rater ϕ_η learns to assign higher weights to noise samples that lead to better generalization of the diffusion model ϵ_θ after several training updates. This is closely related to meta-learning and data valuation approaches (Calian et al., 2025; Shu et al., 2019), but here the optimization operates over *noise realizations* rather than data points.

Meta-Gradient Computation. Optimizing J_S requires differentiating through the inner optimization process. By the chain rule,

$$\nabla_\eta J_S(\eta) = \frac{\partial \mathcal{L}_{\text{val}}}{\partial \theta}\Big|_{\theta^{(S)}(\eta)} \cdot \frac{\partial \theta^{(S)}(\eta)}{\partial \eta}, \tag{6}$$

where $\frac{\partial \theta^{(S)}(\eta)}{\partial \eta}$ captures how changes in the rater affect the inner diffusion model after S updates.

We compute it by differentiating through all S gradient steps using mixed-mode differentiation (Mixflow-MG (Kemaev et al., 2025)), which keeps the memory cost of the unrolled computation tractable.

Algorithm 1 Meta-Learning a Noise Rater for Diffusion Models

Inputs: Training dataset $\mathcal{D}_{\text{train}}$ and validation dataset $\mathcal{D}_{\text{val}}$; initialize noise rater parameters η (meta-optimizer M) and diffusion model parameters θ (optimizer D); number of outer steps N and inner steps S; batch size B; group size K.
for outer step $n = 1, \dots, N$ **do**
  for inner step $s = 1, \dots, S$ **do**
    Sample a grouped batch $\mathcal{B} = \{(\mathbf{x}_{0,j}, \tilde{c}_j, t_j, \{\epsilon_j^{(k)}\}_{k=1}^K)\}_{j=1}^B$ from $\mathcal{D}_{\text{train}}$
    For group j, evaluate the noise rater scores $s_j^{(k)} = \phi_\eta(\epsilon_j^{(k)}, t_j, \mathbf{x}_{0,j}, \tilde{c}_j)$ for $k = 1, \dots, K$
    Convert scores into normalized weights $w_j^{(k)} = \exp(s_j^{(k)}) / \sum_{i=1}^K \exp(s_j^{(i)})$
    Construct noisy inputs $\mathbf{x}_{t,j}^{(k)}$, and compute the weighted loss
      $\mathcal{L}_{\mathcal{B}}(\theta) = \mathbb{E}_j\big[\sum_{k=1}^K w_j^{(k)} \|\epsilon_j^{(k)} - \epsilon_\theta(\mathbf{x}_{t,j}^{(k)}, t_j, \tilde{c}_j)\|^2\big]$
    Update model parameters $\theta_s \leftarrow \theta_{s-1} - \alpha \nabla_\theta \mathcal{L}_{\mathcal{B}}(\theta)$ using optimizer D
  end for
  Sample a validation batch $\mathcal{B}_{\text{val}} \sim \mathcal{D}_{\text{val}}$
  Compute validation loss $\mathcal{L}_{\text{val}}(\theta_S)$ and update $\eta \leftarrow \eta - \beta \nabla_\eta \mathcal{L}_{\text{val}}(\theta_S)$ using meta-optimizer M
end for
return η

3.3 Post Meta-Training: Rater-Guided Noise Selection

After the meta-learning stage, we obtain a trained noise rater ϕ_η that captures the relative utility of noise instances for improving generalization. We then transition to a standard diffusion training phase in which the rater is *frozen*.

Top-1 noise selection. For each data sample $(\mathbf{x}_0, \tilde{c}, t)$, we sample a pool of K' candidate noise instances $\{\epsilon^{(k)}\}_{k=1}^{K'}$ and select the highest-scoring one:

$$k^* = \arg\max_{1 \le k \le K'} \phi_\eta(\epsilon^{(k)}, t, \mathbf{x}_0, \tilde{c}). \tag{7}$$

Only $\epsilon^{(k^*)}$ is passed to the diffusion model, so each training step costs one diffusion forward/backward pass plus K' rater evaluations. Selection depends only on the ordering of the raw scores; the group-wise softmax of Section 3.1 is used solely during meta-training.

Soft weighting as a surrogate for hard selection. The soft weighting of Section 3.1 is a differentiable surrogate for learning a ranking over candidate noises; Top-1 selection is a deployment policy that exploits this ranking. The two are not equivalent, and we validate hard selection empirically (Section 4). Note also that the selected noise no longer follows $\mathcal{N}(0, I)$ but a rater-induced distribution $q_\eta(\epsilon \mid \mathbf{x}_0, t, \tilde{c})$, so Stage II minimizes

$$\mathbb{E}_{\epsilon \sim q_\eta(\cdot \mid \mathbf{x}_0, t, \tilde{c})}\big[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \tilde{c})\|^2\big] \tag{8}$$

rather than Equ. 1. This shift is intentional and analogous to non-uniform timestep sampling: it biases the training signal toward more useful noise realizations, while the forward corruption and the sampling procedure remain unchanged.

Theorem 3.1 (Meta-gradient for noise-weighted diffusion training) *Let $\mathcal{L}_{\text{inner}}(\theta; \eta)$ be the inner objective and $\ell^{(k)}(\theta) = \|\epsilon^{(k)} - \epsilon_\theta(\mathbf{x}_t, t, \tilde{c})\|^2$ be the diffusion loss for the k-th noise instance. Assume that $\ell^{(k)}(\theta)$ is twice continuously differentiable in θ, and that $\mathcal{L}_{\text{inner}}(\theta; \eta)$ is μ-strongly convex in θ for some $\mu > 0$. Then the optimal inner solution $\theta^*(\eta)$ is locally unique and differentiable, and the outer objective $J(\eta) = \mathcal{L}_{\text{val}}(\theta^*(\eta))$ has gradient*

$$\nabla_\eta J(\eta) = -\Big(\nabla_\eta \nabla_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta)\Big)^\top \Big(\nabla_\theta^2 \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta)\Big)^{-1} \nabla_\theta \mathcal{L}_{\text{val}}(\theta^*(\eta)). \tag{9}$$

Interpretation. Thm. 3.1 characterizes an idealized bilevel problem in which the inner objective admits a locally unique optimum. These assumptions provide an interpretable implicit-gradient expression but do not generally hold for deep diffusion models. Our implementation instead optimizes a

finite-step unrolled meta-objective and differentiates directly through the inner optimization trajectory, without explicitly forming or inverting the inner Hessian.

Theorem 3.2 (Gradient alignment principle for noise weighting) *Consider a single inner update with learning rate $\alpha > 0$: $\theta^+(\eta) = \theta - \alpha \sum_{k=1}^{K} w^{(k)}(\eta) \nabla_\theta \ell^{(k)}(\theta)$, where $\ell^{(k)}(\theta)$ denotes the diffusion loss for the k-th noise instance.*

Assume that the validation loss $\mathcal{L}_{\text{val}}(\theta)$ has L-Lipschitz continuous gradients. Then

$$\mathcal{L}_{\text{val}}(\theta^+(\eta)) \leq \mathcal{L}_{\text{val}}(\theta) - \alpha \sum_{k=1}^{K} w^{(k)}(\eta) \left\langle \nabla_\theta \mathcal{L}_{\text{val}}(\theta), \nabla_\theta \ell^{(k)}(\theta) \right\rangle + \mathcal{O}(\alpha^2). \tag{10}$$

Consequently, up to second-order terms, minimizing the post-update validation loss encourages assigning larger weights to noise instances $\epsilon^{(k)}$ whose induced training gradients are better aligned with the validation gradient.

Interpretation. Thm. 3.2 reveals that noise weighting follows a *gradient alignment principle*. Each noise instance $\epsilon^{(k)}$ contributes in proportion to $\left\langle \nabla_\theta \mathcal{L}_{\text{val}}(\theta), \nabla_\theta \ell^{(k)}(\theta) \right\rangle$, which measures how well its induced training gradient aligns with the validation objective. Minimizing the post-update validation loss, therefore, encourages assigning larger weights to noise instances whose gradients are better aligned with improving generalization, while downweighting conflicting or uninformative ones. In effect, the noise rater learns a directionally optimized gradient estimator that amplifies useful learning signals and suppresses detrimental noise.

4 RESULTS

Throughout the experiments, we observe three properties of training noise: noise realizations are not equally useful (Section 4.2), their utility depends jointly on the data sample and the timestep (Section 4.3), and noise selection is complementary to timestep-level design (Section 4.4). Section 4.5 shows that this gain survives at matched compute: once the one-time rater training is amortized, noise selection also makes diffusion training more efficient.

4.1 EXPERIMENT SETUPS

Datasets and tasks. We evaluate three tasks on two datasets: (1) FFHQ (Karras et al., 2019) at 256×256 for unconditional generation; (2) IMAGENET (Deng et al., 2009) at 256×256 for unconditional generation and (3) class-conditional generation, where we apply CFG (Ho and Salimans, 2022) with scale 1.25 at sampling time.

Models. The diffusion model is DiT (Peebles and Xie, 2023) in the latent space of a pre-trained VAE (Rombach et al., 2022); we use DiT-S/2 for the main results and DiT-B/2 and DiT-L/2 in Appx. E. The rater is a dual-stream DiT-S/2 in the style of SD3 (Esser et al., 2024): the image latent and the noise are patchified into two token streams that share joint attention, the timestep and class are injected through AdaLN, and the noise-stream tokens are mean-pooled and projected to a scalar score (Appx. C.1, Fig. 2).

Training protocol and evaluation. (i) We train DiT-S/2 with standard i.i.d. Gaussian noise to a base checkpoint (400k steps for ImageNet, 200k for FFHQ); (ii) we train the rater for 2k steps on top of this frozen checkpoint, with a 10% random split of the training set as its outer-loop validation set; (iii) we resume diffusion training with rater-selected noise (k=4) for 80k steps on ImageNet, reporting FID-50K (Heusel et al., 2017) every 20k steps, and for 40k steps on FFHQ, reporting every 10k. Class-conditional ImageNet is sampled with CFG. Hyperparameters are in Appx. C.2.

4.2 NOISE REALIZATIONS ARE NOT EQUALLY USEFUL

Table 1 compares five noise selection strategies. Vanilla is standard training with a single i.i.d. noise; the other four draw $k = 4$ candidates per sample and keep one. Naive (min) and Naive (max) keep the

Table 1: Comparison of noise-selection strategies for DiT-S/2 on FFHQ and ImageNet (256×256). FID-50K reported along training; lower is better.

Dataset	Step	Vanilla	Naive (min)	Naive (max)	Rater (bottom)	Rater (top)
ImageNet, Uncond.	+20k	63.60	64.69	62.65	65.66	**61.88**
	+40k	62.93	64.50	62.00	65.51	**61.03**
	+60k	62.26	63.97	61.41	64.85	**60.37**
	+80k	61.69	63.31	60.82	64.43	**59.97**
ImageNet, Cond.	+20k	52.02	53.28	51.29	53.96	**50.71**
	+40k	51.31	53.02	50.47	53.70	**49.88**
	+60k	50.61	52.52	49.95	53.12	**49.11**
	+80k	50.12	51.82	49.29	52.67	**48.65**
FFHQ, Uncond.	+10k	11.98	13.14	**11.47**	13.22	11.54
	+20k	11.72	13.54	11.32	13.92	**11.28**
	+30k	11.57	13.66	11.18	14.36	**11.09**
	+40k	11.36	13.62	11.04	14.55	**10.93**

Table 2: Ablation of rater's input conditions. We progressively remove the timestep embedding, image input, and class label. FID-50K is reported every 20k steps. All raters are trained for 2k steps.

				ImageNet 256×256, FID-50K $\downarrow$			
Variant	Timestep	Image	Class	+20k	+40k	+60k	+80k
Full (ours)	✓	✓	✓	**61.88**	**61.03**	**60.37**	**59.97**
w/o timestep	✗	✓	✓	62.73	62.22	61.74	61.40
w/o image	✓	✗	✓	62.08	61.28	60.66	60.29
w/o image & class	✓	✗	✗	62.05	61.43	60.81	60.39
w/o image, timestep & class	✗	✗	✗	62.74	62.33	61.94	61.40

candidate with the smallest and largest norm, and Rater (bottom) and Rater (top) keep the candidate with the lowest and highest *NoiseRater* score. At every checkpoint, Rater (top) improves over Vanilla and Rater (bottom) falls below it, which shows that noise realizations at the same timestep are not equally useful and that the rater's score orders them by utility. Norm carries some signal, as Naive (max) helps and Naive (min) hurts, but the rater's gain over Vanilla is roughly twice that of Naive (max) on ImageNet.

4.3 Noise utility depends jointly on the data sample and the timestep

To test which context the rater relies on, we ablate its inputs and retrain it (1) without the timestep embedding, (2) without the image, (3) without both the image and the class label, and (4) with the noise alone (Table 2). Removing the timestep causes the largest degradation, from 59.97 to 61.40 at +80k, which matches the noise-only rater: the image and class label help only when the timestep is present. The noise-only rater retains a small gain over Vanilla (61.40 vs. 61.69), so what the rater learns beyond magnitude is almost entirely contextual. Removing the image or the class label alone degrades FID by a smaller but consistent margin. Together, these results show that noise utility depends jointly on the data sample and the timestep rather than on the noise alone.

4.4 Noise selection is complementary to timestep-level design

Table 3 combines NoiseRater (top-1 selection from $k = 4$ candidates) with three established timestep-level designs: Min-SNR loss weighting (Hang et al., 2023), P2 loss weighting (Choi et al., 2022), and the logit-normal timestep sampling of SD3 (Esser et al., 2024). Each scheme improves over vanilla training on its own, and adding NoiseRater improves each of them further at every checkpoint. The gain from noise selection is largely preserved across schemes: its FID reduction at +80k goes from -1.72 on vanilla training to between -1.36 and -1.51 on the three schemes, i.e., roughly 80–90% of the standalone gain is retained even on top of the strongest scheme. This indicates that instance-level noise selection addresses an axis of the training objective that timestep weighting and scheduling do not reach.

Table 3: Noise selection is complementary to timestep-level design. FID-50K for DiT-S/2 on ImageNet (256×256, unconditional); each pair differs only in adding NoiseRater top-1 selection (K=4) on top of the timestep-level scheme. Δ is the FID reduction from adding NoiseRater at +80k.

	ImageNet 256×256, FID-50K ↓				
Method	**+20k**	**+40k**	**+60k**	**+80k**	Δ_{+80k}
Vanilla	63.60	62.93	62.26	61.69	-1.72
+ NoiseRater	**61.88**	**61.03**	**60.37**	**59.97**	
Min-SNR	62.95	61.41	60.18	59.05	-1.51
+ NoiseRater	**61.17**	**59.60**	**58.42**	**57.54**	
P2	62.37	60.20	58.72	57.45	-1.40
+ NoiseRater	**60.51**	**58.46**	**57.06**	**56.05**	
Logit-normal	61.75	59.72	58.14	57.16	-1.36
+ NoiseRater	**60.38**	**58.29**	**56.77**	**55.80**	

Table 4: Effect of the candidate pool size k under equal steps and equal compute, for DiT-S/2 on ImageNet 256×256 resuming from the 400k checkpoint.

		ImageNet 256×256, FID-50K ↓			
	Strategy	+20k	+40k	+60k	+80k
Equal steps	k=2	62.50	61.68	61.07	60.38
	k=4	61.88	**61.03**	**60.37**	**59.97**
	k=8	**61.76**	61.74	61.41	60.97
	No Rater (k=1)	63.60	62.93	62.26	61.69
	Strategy	6.7 GPU-h	9.0 GPU-h	13.5 GPU-h	17.6 GPU-h
Equal compute	k=2	62.35	62.01	61.44	60.98
	k=4	61.88	**61.59**	**61.03**	**60.62**
	k=8	**61.86**	61.76	61.75	61.74
	No Rater (k=1)	63.25	62.90	62.23	61.69

4.5 The compute cost of noise selection.

The previous sections establish that *NoiseRater* selects noise of higher utility. However, deploying *NoiseRater* adds compute, including a one-time cost for rater training and, at every diffusion step, k rater evaluations for candidate scoring. This section examines the resulting trade-off along two axes, the candidate pool size k and the rater size, and reports each configuration under both equal training steps and equal compute. Equal-step results isolate the quality of the rater's noise valuation; equal-compute results ask whether that valuation still pays off once scoring overhead is charged. Equal compute counts the per-step scoring cost but not the one-time rater training (∼16 GPU-h, Appx. C.3), which is amortized across runs and, as Appx. E shows, across backbones.

Effect of k. Table 4 compares candidate pool sizes. A larger pool raises the chance of containing a high-utility noise but costs k rater evaluations per step. At equal steps, $k = 4$ is best throughout, and $k = 8$ is worse than $k = 4$ from +40k onward despite the larger pool. We hypothesize that top-1 selection from a large pool narrows the training noise distribution q_η (Eq. 8) too far from $\mathcal{N}(0, I)$, trading diversity of the training signal for per-sample utility. At equal compute, $k = 4$ remains the best trade-off, and $k = 8$ barely improves after 6.7 GPU-h.

Effect of rater size. Table 5 varies the rater's patch size p, hidden dimension h, and depth d. At equal steps, valuation quality grows with rater capacity: the largest rater (p=2, h=384, d=12) is best at every checkpoint, and halving h and d costs about 0.65 FID at +80k, whereas doubling p costs only 0.24. At equal compute, the picture changes because a larger p cuts the token count per stream by four and makes scoring much cheaper: the p=4, h=384, d=12 variant gives the best FID per GPU-hour, while the two shallow raters fall behind despite their lower cost. The most efficient rater

Table 5: Effect of the rater's size at equal training steps and compute. $p \uparrow$ indicates patch size, the larger the faster. $h \downarrow$ indicates hidden dimension, the smaller the faster. $d \downarrow$ indicates the transformer depth for the rater, the smaller the faster.

	$p \uparrow$	$h \downarrow$	$d \downarrow$	ImageNet 256×256, FID-50K ↓			
				+20k	+40k	+60k	+80k
Equal steps	2	384	12	**61.88**	**61.03**	**60.37**	**59.97**
	4	384	12	62.10	61.40	60.81	60.21
	2	192	6	61.97	61.50	61.02	60.64
	4	192	6	62.40	61.74	61.12	60.62
	No rater			63.60	62.93	62.26	61.69
				6.7 GPU-h	9.0 GPU-h	13.5 GPU-h	17.6 GPU-h
Equal compute	2	384	12	61.88	61.59	61.03	60.62
	4	384	12	61.83	**61.50**	**60.94**	**60.43**
	2	192	6	**61.81**	61.60	61.17	60.83
	4	192	6	62.08	61.74	61.13	60.67
	No rater			63.25	62.90	62.23	61.69

Table 6: Compute-matched comparison of the most efficient rater composed with timestep-level designs, for DiT-S/2 on ImageNet 256×256 resuming from the 400k checkpoint.

	Method	ImageNet 256×256, FID-50K ↓			
		6.7 GPU-h	9.0 GPU-h	13.5 GPU-h	17.6 GPU-h
Baseline	Vanilla	63.25	62.90	62.23	61.69
Timestep-level	Min-SNR	62.15	61.35	60.11	59.05
	P2	61.35	60.25	58.77	57.45
	Logit-normal	60.68	59.64	58.06	57.16
+ NoiseRater ($p = 4$,$h = 384$,$d = 12$)	Vanilla	61.83	61.50	60.94	60.43
	Min-SNR	60.86	60.11	58.98	58.19
	P2	60.06	59.13	57.68	56.61
	Logit-normal	**59.87**	**58.80**	**57.39**	**56.46**

is therefore not the smallest one but a full-capacity rater with coarser tokenization; we use it for the compute-matched comparison below.

Compute-matched comparison with timestep-level methods. Table 6 composes the $p = 4$ rater with the three timestep-level schemes of Section 4.4 under equal compute. Adding *NoiseRater* improves every scheme at every budget. The reductions at 17.6 GPU-h (-1.26 on vanilla, -0.86 Min-SNR, -0.84 P2, -0.70 logit-normal) are smaller than the equal-step gains in Table 3, as expected once scoring overhead is charged, but consistent in sign and comparable in relative size across schemes.

5 Conclusion

We asked whether all noise realizations are equally useful for training diffusion models, and introduced *NoiseRater*, a meta-learned rater that makes the utility of an individual noise instance measurable. Using it as a probe, we found that noise realizations at the same timestep are not equally useful, that their utility is a property of the noise in context rather than of the noise alone, and that selecting noise by utility is complementary to timestep-level weighting and scheduling. As a practical consequence, a frozen rater improves FID at matched compute and transfers across backbones. We hope these findings establish the choice of noise realization, alongside the choice of timestep, as an axis of diffusion training worth designing for.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks with required disclosure: implementing the Mixflow-MG (Kemaev et al., 2025) for keeping the memory cost of the unrolled computation tractable. We have not used generative AI tools to generate synthetic datasets, develop the theoretical framework, formulate mathematical claims, provide ingredients for or write the proofs, design the experimental methodology, interpret results, clean or reformat data; translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for the following tasks with recommended disclosure: editing the paper to improve readability, suggesting the structure of the abstract and introduction, drafting parts of the results and conclusion sections from our tables and notes, and checking the consistency between the text and the reported numbers across tables. We have reviewed all AI-assisted work. All experiments, numbers, and tables were produced by the authors without AI assistance; every interpretive claim that originated from an LLM was checked by the authors against the reported results before inclusion; and all AI-edited text was read and approved by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code for the noise rater, the bilevel meta-training, and the rater-guided diffusion training is available at the anonymous repository linked in the abstract. The rater architecture and initialization are specified in Appx. C.1, all meta-training hyperparameters in Appx. C.2 (Table 7), and compute requirements in Appx. C.3. The training and evaluation protocol, including checkpoints, step counts, and the FID-50K setup, is described in Section 4.1; the checkpoint-matching rule for larger backbones is given in Appx. E.

REFERENCES

D. Ahn, J. Kang, S. Lee, J. Min, M. Kim, W. Jang, H. Cho, S. Paul, S. Kim, E. Cha, et al. A noise is worth diffusion guidance. *arXiv preprint arXiv:2412.03895*, 2024.

S. Bechtle, A. Molchanov, Y. Chebotar, E. Grefenstette, L. Righetti, G. Sukhatme, and F. Meier. Meta learning via learned loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4161–4168. IEEE, 2021.

D. A. Calian, G. Farquhar, I. Kemaev, L. M. Zintgraf, M. Hessel, J. Shar, J. Oh, A. György, T. Schaul, J. Dean, et al. Datarater: Meta-learned dataset curation. *arXiv preprint arXiv:2505.17895*, 2025.

C. Chen, L. Yang, X. Yang, L. Chen, G. He, C. Wang, and Y. Li. Find: Fine-tuning initial noise distribution with policy optimization for diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6735–6744, 2024.

J. Choi, J. Lee, C. Shin, S. Kim, H. Kim, and S. Yoon. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11472–11481, 2022.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

L. Engstrom, A. Ilyas, B. Chen, A. Feldmann, W. Moses, and A. Madry. Optimizing ml training with metagradient descent. *arXiv preprint arXiv:2503.13751*, 2025.

P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

X. Guo, J. Liu, M. Cui, J. Li, H. Yang, and D. Huang. Initno: Boosting text-to-image diffusion models via initial noise optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9380–9389, 2024.

T. Hang, S. Gu, C. Li, J. Bao, D. Chen, H. Hu, X. Geng, and B. Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7441–7451, 2023.

T. Hang, S. Gu, J. Bao, F. Wei, D. Chen, X. Geng, and B. Guo. Improved noise schedule for diffusion training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4796–4806, 2025.

H. He, J. Liang, X. Wang, P. Wan, D. Zhang, K. Gai, and L. Pan. Scaling image and video generation via test-time evolutionary search. *arXiv preprint arXiv:2505.17618*, 2025.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR, 2018.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

K. Karunratanakul, K. Preechakul, E. Aksan, T. Beeler, S. Suwajanakorn, and S. Tang. Optimizing diffusion noise can serve as universal motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1334–1345, 2024.

I. Kemaev, D. A. Calian, L. M. Zintgraf, G. Farquhar, and H. Van Hasselt. Scalable meta-learning via mixed-mode differentiation. *arXiv preprint arXiv:2505.00793*, 2025.

J. Kim, T. Yoon, J. Hwang, and M. Sung. Inference-time scaling for flow models via stochastic generation and rollover budget forcing. *arXiv preprint arXiv:2503.19385*, 2025.

J.-Y. Kim, H. Go, S. Kwon, and H.-G. Kim. Denoising task difficulty-based curriculum for training diffusion models. *arXiv preprint arXiv:2403.10348*, 2024.

X. Li, M. Uehara, X. Su, G. Scalia, T. Biancalani, A. Regev, S. Levine, and S. Ji. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*, 2025.

D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

A. Polyak, A. Zohar, A. Brown, A. Tjandra, A. Sinha, A. Lee, A. Vyas, B. Shi, C.-Y. Ma, C.-Y. Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

G. Raya, B. Nguyen, G. Batzolis, Y. Takida, D. Stancevic, N. Murata, C.-H. Lai, Y. Mitsufuji, and L. Ambrogioni. Information-guided noise allocation for efficient diffusion training. *arXiv preprint arXiv:2602.18647*, 2026.

M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

Y. Ren, W. Gao, L. Ying, G. M. Rotskoff, and J. Han. Driftlite: Lightweight drift control for inference-time scaling of diffusion models. *arXiv preprint arXiv:2509.21655*, 2025.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019.

R. Singhal, Z. Horvitz, R. Teehan, M. Ren, Z. Yu, K. McKeown, and R. Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

N. Sun and L. Shi. Variance-aware adaptive weighting for diffusion model training. *arXiv preprint arXiv:2603.10391*, 2026.

Q. Sun, Z. Jiang, H. Zhao, and K. He. Is noise conditioning necessary for denoising generative models? *arXiv preprint arXiv:2502.13129*, 2025.

Z. Tang, J. Peng, J. Tang, M. Hong, F. Wang, and T.-H. Chang. Tuning-free alignment of diffusion models with direct noise optimization. In *ICML 2024 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2024.

Y. Wang, Y. He, and M. Tao. Evaluating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 37:19307–19352, 2024.

Z. Wang, G. Hu, and Q. Hu. Training noise-robust deep neural networks via meta-learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4524–4533, 2020.

J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Z. Zhou, S. Shao, L. Bai, S. Zhang, Z. Xu, B. Han, and Z. Xie. Golden noise for diffusion models: A learning framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17688–17697, 2025.

A Related Work

Noise Optimization at Inference Time. Recent work demonstrates that optimizing initial noise at inference time can significantly improve generation quality. One line of research directly optimizes the noise latent through backpropagation across denoising steps. These methods iteratively update the noise to maximize human-preference rewards (Tang et al., 2024), satisfy specific motion constraints (Karunratanakul et al., 2024), or align with internal attention scores (Guo et al., 2024). While effective and training-free, they incur massive computational overhead at inference due to repeated denoising loops. To amortize this cost, other approaches train auxiliary models to directly predict optimal noise. For instance, FIND (Chen et al., 2024) uses reinforcement learning to adjust the mean and variance of the initial Gaussian, while GoldenNoise (Zhou et al., 2025) trains a network to predict high-quality noise perturbations. Crucially, all treat noise as a test-time control variable. In contrast, we explore the untapped potential of evaluating and weighting noise during training.

Training-Time Loss Reweighting and Noise Selection. Various loss reweighting and noise scheduling strategies improve diffusion training efficiency. Most methods operate at the *timestep level*, adjusting the relative importance or sampling frequency of different noise magnitudes. For loss reweighting, P2 (Choi et al., 2022) prioritizes highly corrupted timesteps to learn global concepts, Min-SNR (Hang et al., 2023) balances inter-timestep optimization conflicts, and Sun and Shi (2026) dynamically adjusts weights based on the variance of loss distributions across log-SNR levels. Concurrently, noise scheduling approaches optimize the distribution of sampled timesteps: Hang et al. (2025) introduces a fixed schedule focusing on the $\log \text{SNR} \approx 0$ region. Kim et al. (2024) uses curriculum learning to progressively introduce harder timesteps, and Raya et al. (2026) proposes a dynamic, information-guided allocation based on entropy reduction. Wang et al. (2024) theoretically analyzes different total weighting, time, and noise schedules. In addition, EDM (Karras et al., 2022) provides a unifying perspective on diffusion design, showing that choices of noise parameterization, preconditioning, and loss scaling implicitly define effective weighting across noise levels. However, these methods treat all noise instances at a given timestep as equally informative. Recent work has also begun to question more fundamental assumptions about noise in diffusion models. For example, Sun et al. (2025) shows that explicit noise conditioning may not be strictly necessary, with models often maintaining competitive performance even without access to timestep information. While such work investigates whether noise information is required at all, it does not address how the variability of noise *instances* influences learning. In contrast, we assume standard conditioning and focus on a complementary question: *given noise, which specific realizations are most useful for training?*

Meta-Learning for Training Optimization. Meta-learning techniques have been explored for adaptive data weighting (Bechtle et al., 2021). Early methods (Jiang et al., 2018; Ren et al., 2018) used online meta-gradients to dynamically weight training examples. To improve scalability and generalization, subsequent work parameterized these weighting mechanisms using neural networks trained via bilevel optimization, evolving from dynamic loss-to-weight mappings (Shu et al., 2019; Wang et al., 2020) to large-scale dataset curation models (Calian et al., 2025). We introduce this bilevel optimization paradigm to diffusion models for the first time. Crucially, we shift the fundamental object of valuation: instead of rating the quality of training *data*, our meta-network is designed to rate the learning utility of the injected *noise* instances.

B Mathematical Analysis

B.1 Proof of Thm. 3.1

Define $F(\theta, \eta) := \nabla_\theta \mathcal{L}_{\text{inner}}(\theta; \eta)$. At the inner optimum $\theta^*(\eta)$, the first-order optimality condition gives $F(\theta^*(\eta), \eta) = \nabla_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta) = 0$. Since $\mathcal{L}_{\text{inner}}(\theta; \eta)$ is μ-strongly convex in θ, its Hessian $H(\eta) := \nabla^2_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta)$ is positive definite and hence invertible. Therefore, by the implicit function theorem, $\theta^*(\eta)$ is locally unique and differentiable with respect to η. Differentiating the optimality condition $\nabla_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta) = 0$ with respect to η yields $\nabla^2_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta) \frac{\partial \theta^*(\eta)}{\partial \eta} + \nabla_\eta \nabla_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta) = 0$. Solving for $\partial \theta^*(\eta)/\partial \eta$, we obtain $\frac{\partial \theta^*(\eta)}{\partial \eta} = -\left(\nabla^2_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta)\right)^{-1} \nabla_\eta \nabla_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta); \eta)$. Now define the outer objective $J(\eta) =$

$\mathcal{L}_{\text{val}}(\theta^*(\eta))$. By the chain rule, $\nabla_\eta J(\eta) = \left(\frac{\partial \theta^*(\eta)}{\partial \eta}\right)^\top \nabla_\theta \mathcal{L}_{\text{val}}(\theta^*(\eta))$. Substituting the expression above gives $\nabla_\eta J(\eta) = -\Big(\nabla_\eta \nabla_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta);\eta)\Big)^\top \Big(\nabla^2_\theta \mathcal{L}_{\text{inner}}(\theta^*(\eta);\eta)\Big)^{-1} \nabla_\theta \mathcal{L}_{\text{val}}(\theta^*(\eta))$. This proves the claimed meta-gradient formula.

B.2 Proof of Thm. 3.2

Let $g_k(\theta) := \nabla_\theta \ell^{(k)}(\theta)$ and $g_{\text{val}}(\theta) := \nabla_\theta \mathcal{L}_{\text{val}}(\theta)$. The single inner update is $\theta^+(\eta) = \theta - \alpha \sum_{k=1}^K w^{(k)}(\eta) g_k(\theta)$, hence $\theta^+(\eta) - \theta = -\alpha \sum_{k=1}^K w^{(k)}(\eta) g_k(\theta)$. Since $\mathcal{L}_{\text{val}}$ has L-Lipschitz continuous gradients, the descent lemma gives $\mathcal{L}_{\text{val}}(\theta^+) \le \mathcal{L}_{\text{val}}(\theta) + \langle g_{\text{val}}(\theta), \theta^+ - \theta\rangle + \frac{L}{2}\|\theta^+ - \theta\|^2$. Substituting the update direction, the linear term becomes $\langle g_{\text{val}}(\theta), \theta^+ - \theta\rangle = -\alpha \sum_{k=1}^K w^{(k)}(\eta) \langle g_{\text{val}}(\theta), g_k(\theta)\rangle$, and $\frac{L}{2}\|\theta^+ - \theta\|^2 = \frac{L\alpha^2}{2}\left\|\sum_{k=1}^K w^{(k)}(\eta) g_k(\theta)\right\|^2 = \mathcal{O}(\alpha^2)$. Therefore, $\mathcal{L}_{\text{val}}(\theta^+(\eta)) \le \mathcal{L}_{\text{val}}(\theta) - \alpha \sum_{k=1}^K w^{(k)}(\eta) \left\langle \nabla_\theta \mathcal{L}_{\text{val}}(\theta), \nabla_\theta \ell^{(k)}(\theta)\right\rangle + \mathcal{O}(\alpha^2)$. Thus, up to second-order terms in α, decreasing the validation loss favors larger weights on noise instances whose training gradients have larger positive alignment with the validation gradient.

C Experimental Details

C.1 Rater Architecture

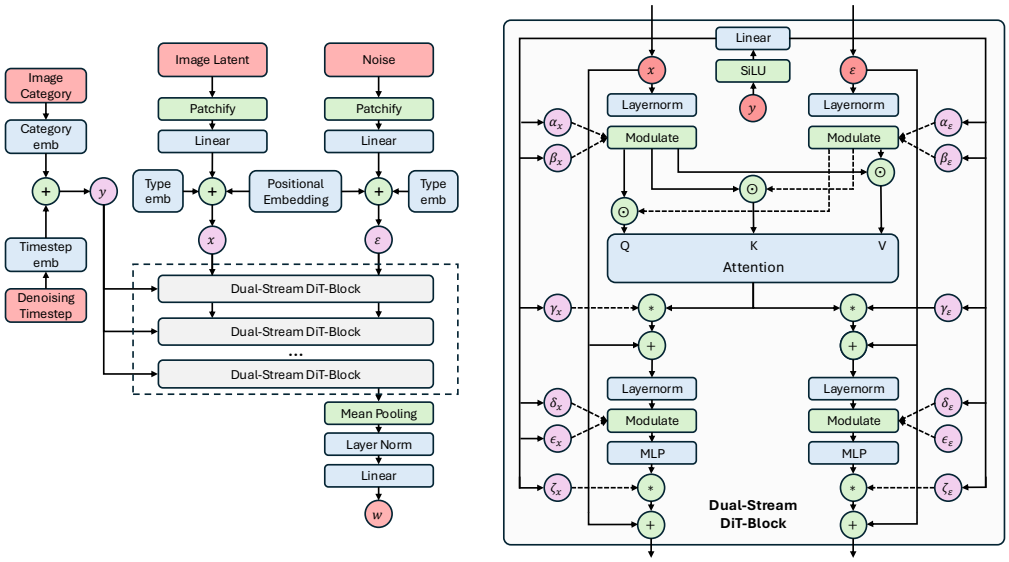

Figure 2: The illustration of the noise rater's architecture, which contains multiple dual-stream DiT blocks to incorporate several condition inputs.

Inputs and tokenization. The rater takes as input a clean image latent x_0, a noise sample ϵ, a diffusion timestep t, and a class label c. We operate in the latent space of the publicly available `stabilityai/sd-vae-ft-mse` VAE (Rombach et al., 2022), which encodes a 256×256 RGB image into a $4 \times 32 \times 32$ latent ($8\times$ spatial downsampling, 4 channels); the noise sample ϵ is drawn from $\mathcal{N}(0, I)$ with the same shape. Both x_0 and ϵ are patchified by two independent PatchEmbed projections (Conv2d with kernel size = stride = $p = 2$), producing two token sequences of length $T = (32/p)^2 = 256$, each in $\mathbb{R}^{T\times D}$ where $D = 384$ is the hidden size. Both streams share a single frozen 2D sin-cos positional embedding (added before the first block) and additionally receive learnable per-stream type embeddings that mark their stream identity, so the joint attention can distinguish image tokens from noise tokens.

Table 7: Hyperparameters for noise rater training.

Hyperparameter	Value
Candidate pool size	4
Outer (rater) learning rate	1×10^{-4}
Inner (diffusion) learning rate	5×10^{-2}
Inner steps per meta update	5
Meta refresh steps	4
Total meta-training steps	2,000
Inner model batch size	32
Validation batch size	128
Gradient clip norm	1.0

Conditioning. The conditioning vector $y \in \mathbb{R}^D$ is the sum of (i) a sinusoidal timestep embedding passed through a two-layer MLP and (ii) a learnable class embedding. The same y is broadcast to every block.

Dual-stream DiT block. Each block maintains two independent residual paths, one for the image stream and one for the noise stream. Each path has their own LayerNorm, MLP, and adaLN modulation parameters. Following DiT, each stream produces six modulation values per block; concretely, a single Linear layer projects y to $12D$ outputs and chunks them into the twelve required scalars per token dimension. Attention is the only sublayer shared across streams: at each block we form the concatenated sequence $[\hat{x}_0^{(\ell)}; \hat{\epsilon}^{(\ell)}] \in \mathbb{R}^{2T \times D}$ from the modulated normalized tokens of the two streams, run a single multi-head self-attention, and split the output back into per-stream halves before applying the per-stream gates and residuals. Joint attention lets noise tokens directly query image content (and vice versa) without requiring additional cross-attention parameters.

Output head. After the final block, the image-stream tokens are discarded; the noise-stream tokens are mean-pooled across the spatial dimension, layer-normalized, and projected by a single linear layer to a scalar score $w \in \mathbb{R}$.

Hyperparameters and initialization. We use the same per-block configuration as DiT-S/2: depth $= 12$, hidden size $D = 384$, 6 attention heads, MLP ratio $= 4$, patch size $p = 2$, operating on 32×32 latents (i.e., $T = 256$ tokens per stream). The total parameter count is approximately $\sim$60M, roughly twice that of DiT-S/2 because of the duplicated MLPs and modulation networks.

Unlike generative DiT models, we do *not* adopt adaLN-Zero (i.e., zero-initializing the adaLN projection and the output head); instead all linear layers use Xavier initialization. We found this necessary because the rater is trained via bilevel optimization: each rater update requires running an inner loop of diffusion updates whose loss is reweighted by the rater's scores. Under adaLN-Zero, the rater's score head is zero-initialized, so all candidate noises receive identical (zero) scores at initialization. After group-wise normalization, this collapses the inner-loop loss weights to a constant, yielding zero gradient signal back to the rater and stalling training before it begins. Standard Xavier initialization breaks this symmetry by giving the rater a non-trivial starting function, and yields stable training across all settings reported in the main paper.

C.2 Training Hyperparameters

Tab. 7 lists the hyperparameters used for noise rater training. The rater is trained on top of a frozen DiT-S/2 checkpoint via a meta-learning objective: an inner loop takes 5 gradient steps on the diffusion model with rater-weighted noise, and an outer loop updates the rater to minimize the resulting diffusion loss on a held-out validation split.

Table 8: Effect of applying the rater at different stages of DiT-S/2 training on ImageNet 256×256. We train a rater on top of the 100k, 400k, and 500k checkpoints, respectively, then continue diffusion training for an additional 80k steps. FID-50K is reported every 20k steps.

	100k		400k		500k	
Train steps	w/o rater	w/i rater	w/o rater	w/i rater	w/o rater	w/i rater
+20k	86.08	**85.09**	63.60	**61.87**	60.42	**60.13**
+40k	82.39	**81.33**	62.93	**61.03**	59.96	**59.54**
+60k	79.35	**78.57**	62.26	**60.37**	59.38	**58.98**
+80k	76.63	**76.28**	61.69	**59.95**	58.90	**58.68**

C.3 Computation Resources

Rater training. With the hyperparameters in Tab. 7, training the noise rater takes approximately 4 hours of wall-clock time on 4 NVIDIA H100 GPUs (16 CPU cores). This is a one-time cost that is amortized across all subsequent diffusion runs.

Diffusion training. All diffusion training is conducted on 4 NVIDIA H100 GPUs.

(i) DiT-S/2 main runs. Pre-training DiT-S/2 from scratch to 400k steps takes about 22 hours. Each 80k-step continuation run takes 4.3, 5.6, 7.0, and 9.0 hours for $k{=}1, 2, 4, 8$ respectively.

(ii) Larger-backbone generalization. (Appx. E) On DiT-B/2, the 80k continuation takes 5.4 hours without the rater and 6.7 hours with the rater at $k{=}2$. On DiT-L/2, it takes 9.2 hours without the rater and 10.4 hours with the rater at $k{=}2$.

D Rater of different training stage.

We explored whether the rater is equally beneficial when applied at different stages of diffusion training. We test this by training a rater on top of DiT-S/2 checkpoints at 100k, 400k, and 500k steps, and then continuing diffusion training for an additional 80k steps with $k{=}2$. As shown in Tab. 8, the rater consistently improves FID at every stage and every checkpoint, but the magnitude of the gain varies substantially: applied at 400k, the rater yields a 2.1%–2.8% relative FID reduction, whereas at 100k and 500k the gain shrinks to 0.4%–1.3% and 0.4%–0.7% respectively. We attribute this gap to hyperparameter sensitivity. All three runs reuse the same training recipe tuned on the 400k checkpoint; we did not re-tune for the early or late training regimes, where the diffusion model's noise-to-signal characteristics differ noticeably from those at 400k. We expect stage-specific tuning to recover most of the gap, and we leave a thorough hyperparameter sweep across training stages to future work. Even without such tuning, the rater never hurts and delivers consistent improvements across the full training trajectory.

E Generalization to larger backbones.

We test that the rater learns a notion of noise utility, conditioned on $(\mathbf{x}_0, t, \tilde{c})$, that does not depend on the specific backbone it was trained with. Our results show that the rater trained on small backbone is capable to select high utility noise for models with larger backbone.

Training stage matching. Since different backbones converge at different rates, naively reusing the same step count would compare backbones at very different points in their training trajectories. To match the rater at a suitable training stage, we instead select target checkpoints for DiT-B/2 and DiT-L/2 that match the *absolute training loss* of DiT-S/2 at 400k steps.

Concretely, let t denote the training step, and let $L_S(t)$, $L_B(t)$, and $L_L(t)$ denote the training loss curves of DiT-S/2, DiT-B/2, and DiT-L/2 respectively. We define the matched checkpoint for each larger backbone as

$$t_B^\star = \min\left\{\, t \,:\, \widetilde{L}_B(t') \le \widetilde{L}_S(400\text{k}) \;\text{ for all }\; t' \in [t, t+W] \,\right\}, \tag{11}$$

Table 9: Generalization to larger backbones. A rater trained on DiT-S/2 (at the 400k checkpoint) is applied off-the-shelf to train DiT-B/2 and DiT-L/2 on ImageNet 256×256, resuming from their 200k and 100k checkpoints, respectively. The rater is frozen and never retrained on the larger backbones.

			ImageNet 256×256, FID-50K ↓			
Backbone	Resume from	Strategy	+20k	+40k	+60k	+80k
DiT-B/2	200k	No Rater	53.54	51.78	50.15	48.65
		Rater (k=2)	**52.55**	**50.93**	**49.32**	**47.80**
DiT-L/2	100k	No Rater	48.09	43.60	40.13	36.95
		Rater (k=2)	**47.93**	**43.34**	**39.86**	**36.94**

and analogously for $t_L^\star$, where $\widetilde{L}(\cdot)$ denotes the training loss smoothed with a centered rolling window of size 500, and $W = 1000$ logged steps is a stability window that prevents single-step dips from triggering a spurious match. The smoothing and stability window together make the selection robust to the noisy loss curves typical of diffusion training.

Applying this procedure with $L_S(400\text{k})$ as the target yields $t_B^\star = 211$k for DiT-B/2 and $t_L^\star = 119$k for DiT-L/2. Intuitively, larger backbones reach the same loss level in fewer steps, which is consistent with prior observations on capacity-driven sample efficiency in diffusion training (Peebles and Xie, 2023). Thus, we take the rater trained on DiT-S/2 at the 400k checkpoint and apply it to train DiT-B/2 and DiT-L/2 from their 200k and 100k checkpoints (rounded it down to the nearest saved checkpoint), respectively, for an additional 80k steps with k=2.

Generalization performance. As shown in Tab. 9, the small-model rater consistently improves FID on both larger backbones at every checkpoint, with DiT-B/2 gaining 0.85 FID at +80k ($48.65 \rightarrow 47.80$) and DiT-L/2 matching the baseline at +80k while leading at every earlier checkpoint. This indicates that the rater captures a backbone-agnostic notion of noise quality, and that one cheap rater training run can be amortized across an entire family of diffusion models.

F Effect of the Number of Inner Steps S

The meta-objective J_S in Section 3.2 is a finite-step truncation of the idealized bilevel objective J analyzed in Thm. 3.1. The number of inner steps S therefore controls how closely the rater's training signal approximates the converged inner optimum, but also the cost of each meta-update, since the meta-gradient is obtained by differentiating through all S unrolled diffusion updates. To quantify this trade-off, we train raters with $S \in \{1, 3, 5, 7\}$, keeping every other hyperparameter in Tab. 7 fixed, and evaluate each rater under the same Stage-II protocol as the main experiments (DiT-S/2, ImageNet 256×256, unconditional, Top-1 selection with $K' = 4$, 80k continuation steps from the 400k checkpoint).

Table 10: Effect of the number of inner steps S on meta-training cost and downstream quality. Meta-training cost is measured in GPU-hours on 4×H100 for the full 2k meta-training steps. FID-50K is reported at +80k Stage-II steps. $S = 5$ is the setting used in all main-paper experiments.

S	Meta-training cost (GPU-h)	FID-50K ↓
1	3.87	60.01
3	9.75	60.19
5	15.82	**59.97**
7	21.49	61.41
No rater	–	61.69

Cost. Meta-training cost grows linearly in S, at roughly 3 GPU-hours per additional inner step, because each meta-update must run S diffusion forward/backward passes and then backpropagate through the resulting trajectory. Memory grows with S as well: mixed-mode differentiation (Kemaev et al., 2025) avoids materializing second-order terms, but the activations of all S inner steps must still be retained for the reverse pass.

Downstream quality. Downstream FID is comparatively insensitive to S in the range $1 \leq S \leq 5$: all three raters improve over vanilla training by 1.5–1.7 FID, and $S = 5$ is best by a small margin. Increasing to $S = 7$ degrades FID to 61.41, close to the no-rater baseline. We attribute this to the well-known instability of long unrolled meta-gradients (Maclaurin et al., 2015): as the unroll length grows, the meta-gradient becomes higher-variance and more poorly conditioned, and with the meta-learning rate held fixed the rater fails to converge to a useful ranking within the 2k-step budget. We did not re-tune the meta-learning rate for $S = 7$.