Noise Hypernetworks: Amortizing Test-Time Compute in Diffusion Models

Luca Eyring[1,2,3] **Shyamgopal Karthik**[1,2,3,4] **Alexey Dosovitskiy**[5]
Nataniel Ruiz[6*] **Zeynep Akata**[1,2,3*]

[1]Technical University of Munich [2]Munich Center of Machine Learning
[3]Helmholtz Munich [4]University of Tübingen [5]Inceptive [6]Google
luca.eyring@tum.de

Abstract

The new paradigm of test-time scaling has yielded remarkable breakthroughs in Large Language Models (LLMs) (e.g. reasoning models) and in generative vision models, allowing models to allocate additional computation during inference to effectively tackle increasingly complex problems. Despite the improvements of this approach, an important limitation emerges: the substantial increase in computation time makes the process slow and impractical for many applications. Given the success of this paradigm and its growing usage, we seek to preserve its benefits while eschewing the inference overhead. In this work we propose one solution to the critical problem of integrating test-time scaling knowledge into a model during post-training. Specifically, we replace reward guided test-time noise optimization in diffusion models with a Noise Hypernetwork that modulates initial input noise. We propose a theoretically grounded framework for learning this reward-tilted distribution for distilled generators, through a tractable noise-space objective that maintains fidelity to the base model while optimizing for desired characteristics. We show that our approach recovers a substantial portion of the quality gains from explicit test-time optimization at a fraction of the computational cost. Code is available at https://github.com/ExplainableML/HyperNoise.

1 Introduction

Recently, inference-time scaling has made remarkable breakthroughs in Large Language Models [28, 40, 84] and generative vision models, enabling models to spend more computation during inference to solve complex problems effectively. Drawing from the success and growing usage of test-time compute in LLMs, several methods have attempted to apply similar ideas in the context of diffusion models for generation [6, 22, 59, 66, 67, 80, 88, 90, 92, 93, 97]. The goal of this process is to spend additional compute during inference to obtain generations that better reflect desired output properties.

Diffusion model test-time techniques that optimize the initial noise or intermediate steps of the diffusion process, often guided by feedback from pre-trained reward models [48, 54, 101, 102, 103, 107], have demonstrated significant promise in improving critical attributes of the generated outputs, such as prompt following, aesthetics, quality and composition [11, 22, 44, 59, 66, 67, 97]. These methods generally fall into two broad categories: gradient-based optimization, which typically requires substantial GPU memory for backpropagation through the full model [6, 22, 46, 67, 97], and gradient-free optimization, which often necessitates a very large number of function evaluations (NFEs), sometimes thousands, of the computationally expensive denoising network [44,

*Equal Supervision

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

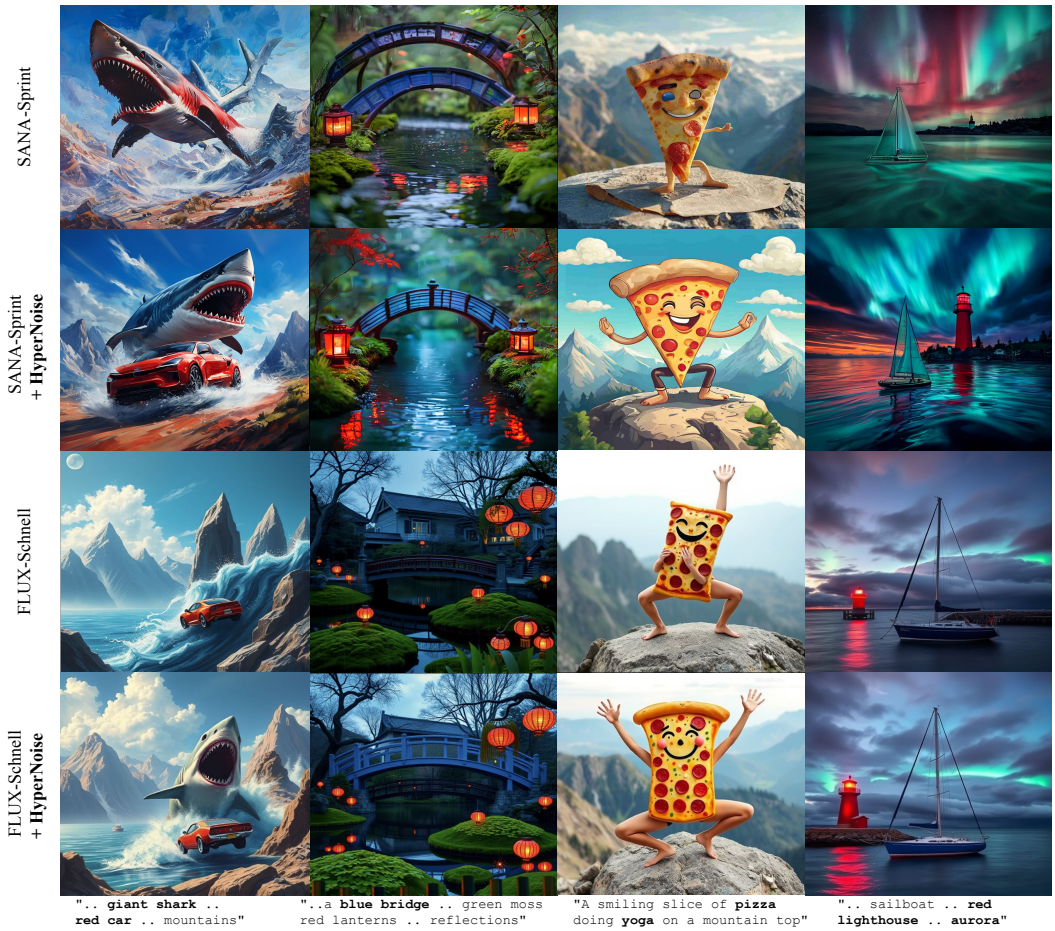

Figure 1: The same initial random noise is used for the base generation and the initialization of the noise hypernetwork. **HyperNoise** significantly improves upon the initially generated image with respect to both prompt faithfulness and aesthetic quality for both SANA-Sprint and FLUX-Schnell.

59, 93]. While both strategies can effectively boost output quality, they introduce considerable latency (exceeding 10 minutes for one generation), severely limiting their practical utility, particularly for real-time applications. This is an instantiation of a global problem of test-time scaling methods that we seek to tackle in this work. The core hypothesis of our work is whether *it is possible to capture a portion of test-time scaling benefits by integrating this knowledge into a neural network during training time?*

To address this, one might consider directly fine-tuning the diffusion model using reward signals [12, 14, 18, 53, 74, 89, 91, 108] or with Direct Preference Optimization (DPO) [34, 45, 52, 77, 98]. The objective here can be formulated as learning a *tilted distribution* (Equation 3), which upweights samples with high reward while maintaining fidelity to a base model's distribution. These methods are usually expensive to train due to the need for backpropagation through the sampling process. Instead, one might consider directly fine-tuning a step-distilled generative model to learn this target distribution. However, this approach typically involves a KL regularization to the base model that is intractable for distilled models. An imbalance or poor estimation of this can lead to the model "reward-hacking", superficially maximizing the reward metric while significantly deviating from the desired underlying data distribution, thus not achieving the genuine desired improvements.

In this work, we propose a different path to realize the benefits of the target tilted distribution (Equation 3), particularly for step-distilled generative models. Our core hypothesis is that instead of modifying the parameters of the base generator, we can achieve the desired output distribution by learning to predict an optimal initial noise distribution. We first show that such an optimal *tilted noise distribution* $p_0^\star$ exists (characterized by Equation 5). When samples from this $p_0^\star$ are passed through the frozen generator, they naturally produce outputs that are distributed according to the target data-space tilted distribution. To learn this tilted noise distribution, we introduce a lightweight

network, f_ϕ, that transforms standard Gaussian noise into a modulated, improved noise latent. The crucial advantage of this approach lies in its optimization objective. In particular, the regularization term, a KL divergence between the modulated noise distribution and the standard Gaussian prior, is defined entirely in the noise space. We show that this noise-space KL divergence can be made tractable and effectively approximated by an L_2 penalty on the magnitude of the noise modification.

This lightweight network forms the core of our approach, which we term **Noise Hyper**networks. It functions akin to a hypernetwork [2, 30, 31, 39, 63, 72, 95, 105] as rather than generating the final image, it produces a specific, optimized starting latent for the main frozen generative model. This effectively guides the output of the base model without any changes to its parameters. Broadly, a hypernetwork is an auxiliary model trained to generate crucial inputs or parameters of a primary model. Our f_ϕ embodies this concept by learning to predict the optimized initial noise as input to the frozen generator. Consequently, *our proposed approach is effectively training a Noise Hypernetwork to perform the task of test-time noise optimization*, by learning to directly output an optimized noise latent in a single step sidestepping the need for expensive, iterative test-time optimization.

Our practical implementation of the noise hypernetwork utilizes Low-Rank Adaptation (LoRA), ensuring it remains parameter-efficient and adds negligible computational cost during inference. We apply our method to text-to-image generation, conducting evaluations with an illustrative "redness" reward task to demonstrate core mechanics, as well as complex alignments using sophisticated human-preference reward models. We demonstrate the efficacy of our approach by applying it to distilled diffusion models SD-Turbo [83], SANA-Sprint [13], and FLUX-Schnell. Overall, our experiments show that we can recover a substantial portion of the quality gains from explicit test-time optimization at a fraction of the computational inference cost. In summary, our contributions are:

1. We introduce HyperNoise, a novel framework that learns to predict an optimized initial noise for a fixed distilled generator, effectively moving test-time noise optimization benefits and computational costs into a one-time post-training stage.
2. We propose the first theoretically grounded framework for learning the reward tilted distribution of distilled generators, through a tractable noise-space objective that maintains fidelity to the base model while optimizing for desired characteristics.
3. We demonstrate through extensive experiments significant enhancements in generation quality for state-of-the-art distilled models with minimal added inference latency, making high-quality, reward-aligned generation practical for fast generators.

2 Background

Preliminaries. Recent generative models are based on a time-dependent formulation between a standard Gaussian distribution $\mathbf{x}_0 \sim p_0 = \mathcal{N}(0, \mathbf{I})$ and a data distribution $\mathbf{x}_1 \sim p_{data}$. These models define an interpolation between the initial noise and the data distribution, such that

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \mathbf{x}_1, \tag{1}$$

where α_t is a decreasing and σ_t is an increasing function of $t \in [0, 1]$. Score-based diffusion [33, 43, 47, 85, 86] and flow matching [3, 55, 56] models share the observation that the process $\mathbf{x}_t$ can be sampled dynamically using a stochastic or ordinary differential equation (SDE or ODE). The neural networks parameterizing these ODEs/SDEs are trained to learn the underlying dynamics, typically by predicting the score of the perturbed data distribution or the conditional vector field. Generating a sample then involves simulating this learned differential equation starting from $\mathbf{x}_0 \sim p_0$.

Step-Distilled Models. The iterative simulation of such ODEs/SDEs often requires numerous steps, leading to slow sample generation. To address this latency, distillation techniques have emerged as a powerful approach. The objective is to train a "student" model that emulates the behavior of a pre-trained "teacher" model (multi-step ODE/SDE simulation) but achieves this with drastically fewer, or even a single, evaluation step(s). Prominent distillation methods such as Adversarial Diffusion Distillation [83], Consistency Models [58, 87], or Flow Maps [9, 10] have enabled the development of highly efficient few-step generative models, like SD/SDXL-Turbo [83] and SANA-Sprint [13]. In this work, we denote such a distilled generator by g_θ. The significantly reduced number of sampling steps in these distilled models makes them more amenable to various optimization techniques and practical for real-time applications, which is why they are the focus of our work.

Test-Time Noise Optimization. Test-time optimization techniques aim to improve pre-trained generative models on a per-sample basis at inference. One prominent gradient-based strategy is

test-time noise optimization [6, 29, 46, 67, 90, 97]. Given a pre-trained generator g_θ (which could be a multi-step diffusion or flow matching model), this approach optimizes the initial noise $\mathbf{x}_0$ for each generation instance. The objective is to find an improved $\mathbf{x}_0^\star$ that maximizes a given reward $r(g_\theta(\mathbf{x}_0))$, often subject to regularization and can be formulated as

$$\mathbf{x}_0^\star = \arg\max_{\mathbf{x}_0}(r(g_\theta(\mathbf{x}_0)) - \mathrm{Reg}(\mathbf{x}_0)), \tag{2}$$

where $\mathrm{Reg}(\mathbf{x}_0)$ is a regularization term designed to keep $\mathbf{x}_0^\star$ within a high-density region of the prior noise distribution p_0, thus ensuring the generated sample $g_\theta(\mathbf{x}_0^\star)$ remains plausible. ReNO [22] adapted this framework for distilled generators g_θ, enabling more efficient test-time optimization compared to full diffusion models. However, this per-sample optimization still incurs significant computational costs at inference, involving multiple forward and backward passes, and increased GPU memory. This inherent latency and computational burden motivate the exploration of methods that can imbue models with desired properties without per-instance test-time optimization.

Reward-based Fine-tuning and the Tilted Distribution. To circumvent the per-sample inference costs associated with test-time optimization, an alternative paradigm is to directly fine-tune the generative model g_θ to align with a reward function. We consider the pre-trained base distilled diffusion model g_θ, which transforms an initial noise sample $\mathbf{x}_0$ into an output sample $\mathbf{x} = g_\theta(\mathbf{x}_0)$. The distribution of these generated output samples is the pushforward of p_0 by g_θ, which we denote as $p^{\text{base}} = (g_\theta)_\sharp p_0$. Given g_θ and a differentiable reward function $r(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ that quantifies the preference of samples $\mathbf{x}$, our objective is to learn the so called *tilted distribution*

$$p^\star(\mathbf{x}) \propto p^{\text{base}}(\mathbf{x}) \exp(r(\mathbf{x})). \tag{3}$$

This target distribution is defined to upweight samples with high rewards under $r(\mathbf{x})$ while staying close to the original $p^{\text{base}}(\mathbf{x})$. We would like to learn $p^\star(\mathbf{x})$ by minimizing the KL divergence $D_{\text{KL}}(p^\phi \| p^\star)$. Here, p^ϕ is the distribution generated by modifying the base process using trainable parameters ϕ. e.g. ϕ could correspond to a fine-tuned version of θ. This objective can be decomposed such that

$$\min_\phi D_{\text{KL}}(p^\phi \| p^\star) = \min_\phi D_{\text{KL}}(p^\phi \| p^{\text{base}}) - \mathbb{E}_{\mathbf{x}\sim p^\phi}[r(\mathbf{x})], \tag{4}$$

where we omit the normalization constant of $p^\star(\mathbf{x})$ which is constant w.r.t. ϕ (see Appendix A.2). This objective encourages the learned model p^ϕ to generate high-reward samples while regularizing its deviation from the original base distribution p^{base}.

Challenges in Direct Reward Fine-tuning of Distilled Models. Directly optimizing Equation 4 by fine-tuning the parameters of a distilled, e.g. one-step, g_θ poses significant challenges. The term $D_{\text{KL}}(p^\phi \| p^{\text{base}})$ requires evaluating the densities of p^ϕ and p^{base}. For typical neural network generators, these densities involve Jacobian determinants through the change-of-variable formula, which are often intractable or computationally prohibitive to compute for high-dimensional data [70]. Previously, a line of work has analyzed fine-tuning Diffusion [89, 91] and Flow matching [18] models based on Equation 4 through the lens of Stochastic Optimal Control. However, this formulation relies on dynamical generative models (SDEs) and its application to distilled models is not straightforward, as these often lack the explicit continuous-time dynamical structure (e.g., an underlying SDE or ODE) that these fine-tuning techniques leverage.

3 Noise Hypernetworks

Given the challenges in directly fine-tuning g_θ, we introduce Noise Hypernetworks (**HyperNoise**), a novel theoretically grounded approach to learn $p^\star$ for distilled generative models. The core idea is to learn a new distribution for the initial noise, p_0^ϕ, such that samples $\hat{\mathbf{x}}_0 \sim p_0^\phi$, when passed through the fixed generator g_θ, produce outputs $\mathbf{x} = g_\theta(\hat{\mathbf{x}}_0)$ that are effectively drawn from the target tilted distribution $p^\star(\mathbf{x})$ (Equation 3). Instead of modifying the parameters θ of the base generator, we keep g_θ fixed. This requires p_0^ϕ to approximate an optimal modulated noise distribution, $p_0^\star$. This *tilted noise distribution*, which precisely steers g_θ to $p^\star$, can be characterized by (Appendix A.3)

$$p_0^\star(\mathbf{x}_0) \propto p_0(\mathbf{x}_0) \exp(r(g_\theta(\mathbf{x}_0))). \tag{5}$$

To realize the modulated noise distribution p_0^ϕ, we parameterize it using a learnable noise hypernetwork f_ϕ (with parameters ϕ). This network defines a transformation T_ϕ that maps initial noise

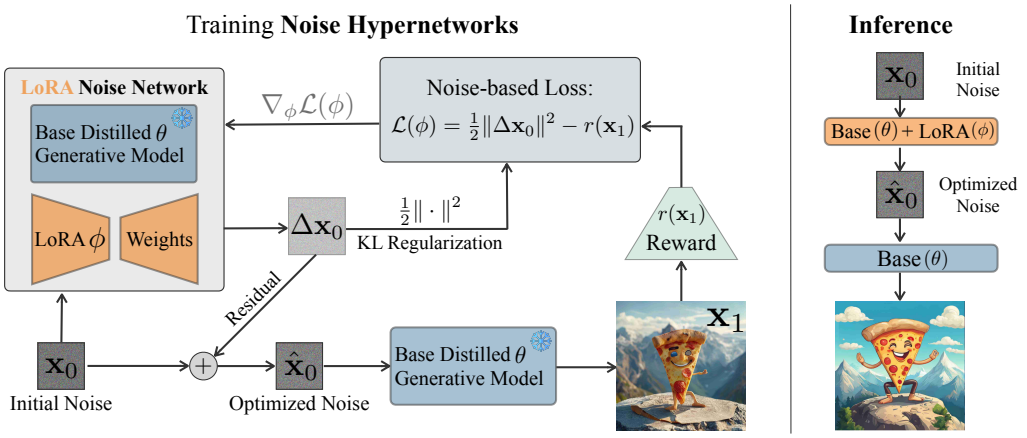

Figure 2: Illustration of our proposed HyperNoise approach. During training, the LoRA parameters are trained to predict improved noises and are optimized by reward maximization subject to KL regularization. During inference, the noise hypernetwork directly predicts the improved noise initialization which is used for the final generation.

samples $\mathbf{x}_0 \sim p_0$ to modulated samples $\hat{\mathbf{x}}_0$ via a residual formulation such that

$$\hat{\mathbf{x}}_0 = T_\phi(\mathbf{x}_0) \coloneqq \mathbf{x}_0 + f_\phi(\mathbf{x}_0). \tag{6}$$

The distribution of these modulated samples, p_0^ϕ, is thus the pushforward of p_0 by T_ϕ, i.e., $p_0^\phi = (T_\phi)_\sharp p_0$. We propose to train the parameters ϕ of the noise modulation network f_ϕ by minimizing the KL divergence $D_{\mathrm{KL}}(p_0^\phi \| p_0^\star)$. This can be shown to be equivalent to minimizing the loss function

$$\mathcal{L}_{\mathrm{noise}}(\phi) = D_{\mathrm{KL}}(p_0^\phi \| p_0) - \mathbb{E}_{\hat{\mathbf{x}}_0 \sim p_0^\phi}[r(g_\theta(\hat{\mathbf{x}}_0))]. \tag{7}$$

Analogously to Equation 4, this objective encourages p_0^ϕ (and thus f_ϕ) to produce initial noise samples $\hat{\mathbf{x}}_0$ that effectively steer the fixed generator g_θ towards high-reward outputs $\mathbf{x}$. The KL term $D_{\mathrm{KL}}(p_0^\phi \| p_0)$ regularizes this steering by ensuring p_0^ϕ remains close to the original noise distribution p_0. Next, we show that in contrast to Equation 4, $\mathcal{L}_{\mathrm{noise}}$ can be made tractable.

3.1 KL Divergence in Noise Space

The resulting KL divergence term $D_{\mathrm{KL}}(p_0^\phi \| p_0)$ is derived in detail in Appendix A. The derivation involves the change of variables formula, simplification of Gaussian log-PDF terms, and an application of Stein's Lemma. This leads to the following expression for the KL divergence:

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\mathbf{x}_0 \sim p_0}[\tfrac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 + \mathrm{Tr}(J_{f_\phi}(\mathbf{x}_0)) - \log|\det(I + J_{f_\phi}(\mathbf{x}_0))|], \tag{8}$$

where $J_{f_\phi}(\mathbf{x}_0)$ is the Jacobian of f_ϕ with respect to $\mathbf{x}_0$. Let $\mathcal{E}(A) \coloneqq \mathrm{Tr}(A) - \log|\det(I + A)|$. Then Equation 8 can be rewritten as $D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\mathbf{x}_0 \sim p_0}[\tfrac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 + \mathcal{E}(J_{f_\phi}(\mathbf{x}_0))]$. To simplify this expression, we analyze the error term $\mathcal{E}(J_{f_\phi}(\mathbf{x}_0))$. The following Theorem provides a bound on this term under a Lipschitz assumption on f_ϕ.

Theorem 1 (Bound on Log-Determinant Approximation Error)**.** *Let $A = J_{f_\phi}(\mathbf{x}_0)$ be the $d \times d$ Jacobian matrix of $f_\phi(\mathbf{x}_0)$. Assume f_ϕ is L-Lipschitz continuous, such that its Lipschitz constant $L < 1$. This implies that the spectral radius $\rho(A) \leq L < 1$. Then, the error term $\mathcal{E}(A) = Tr(A) - \log|\det(I + A)|$ is bounded by*

$$|\mathcal{E}(A)| \leq d(-\log(1 - L) - L). \tag{9}$$

See Appendix A.4 for the full proof. Theorem 7 shows that if the Lipschitz constant L of f_ϕ is sufficiently small (specifically, $L < 1$), the error term $|\mathcal{E}(A)|$ is bounded. For small L, $-\log(1-L) - L \approx L^2/2$, making the bound approximately $dL^2/2$. Thus, the expected error $\mathbb{E}_{\mathbf{x}_0 \sim p_0}[\mathcal{E}(J_{f_\phi}(\mathbf{x}_0))]$ becomes negligible if L is kept small. Under this condition, we can approximate the KL divergence with

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) \approx \mathbb{E}_{\mathbf{x}_0 \sim p_0}[\tfrac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2]. \tag{10}$$

This approximation simplifies the KL divergence term in our objective to a computationally tractable L_2 penalty on the magnitude of the noise modification $f_\phi(\mathbf{x}_0)$. Substituting it into our initial noise modulation objective (Equation 7), we arrive at the final loss to minimize

$$\mathcal{L}_{\text{noise}}(\phi) = \mathbb{E}_{\mathbf{x}_0 \sim p_0}[\tfrac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 - r(g_\theta(\mathbf{x}_0 + f_\phi(\mathbf{x}_0)))]. \tag{11}$$

Connection to test-time noise optimization. Our proposed method addresses the same fundamental goal as Noise Optimization (Equation 2) of steering generation towards high-reward outputs while maintaining fidelity to the base distribution. However, instead of performing iterative optimization for each sample at inference time, we amortizes this optimization into a one-time post-training process. By learning the noise modulation network f_ϕ, we effectively pre-computes a general policy for transforming any initial noise $\mathbf{x}_0$. Consequently, steered generation with HyperNoise remains highly efficient at inference, requiring only a single forward pass through f_ϕ and then g_θ.

Theoretical Justification via Data Processing Inequality. The KL divergence term $D_{\text{KL}}(p_0^\phi \| p_0)$ in our objective (Equation 7) provides a principled way to regularize the output distribution in data space. The Data Processing Inequality (DPI) [15] states that for any function, such as our fixed generator g_θ, the KL divergence between its output distributions is upper-bounded by the KL divergence between its input distributions. In our context, where $p_0^\phi = (T_\phi)_\sharp p_0$ is the distribution of modulated noise $\hat{\mathbf{x}}_0 = T_\phi(\mathbf{x}_0)$ and $p^{\text{base}} = (g_\theta)_\sharp p_0$ is the base output distribution, the DPI implies

$$D_{\text{KL}}(p_0^\phi \| p_0) \geq D_{\text{KL}}((g_\theta)_\sharp p_0^\phi \| (g_\theta)_\sharp p_0). \tag{12}$$

Thus, by minimizing $D_{\text{KL}}(p_0^\phi \| p_0)$ in the noise space, we effectively minimizes an upper bound on the KL divergence between the steered output distribution $(g_\theta)_\sharp p_0^\phi$ and the original base distribution p^{base}. This offers a theoretically grounded mechanism for controlling the deviation of the generated data distribution, complementing the empirical reward maximization, even when direct computation of data-space KL divergences (as in Equation 4) is intractable.

3.2 Effective Implementation

To implement Noise Hypernetworks efficiently and ensure stable training, we adopt several key strategies for the noise modulation network f_ϕ and the training process, summarized in Algorithm 1. Note that our training algorithm (Equation 11) does not require target data samples from $p^\star$, p^{base}, nor p_{data}. It only requires: (1) base noise samples $\mathbf{x}_0 \sim p_0$, (2) the fixed generator g_θ, and (3) the reward function $r(\cdot)$. For conditional $g_\theta(\cdot|c)$, it additionally requires the conditions c.

Lightweight Noise Hypernetwork with LoRA. The noise modulation network f_ϕ is instantiated by reusing the architecture of the pre-trained generator g_θ and making it trainable via Low-Rank Adaptation (LoRA) [35]. The original g_θ weights are frozen, and only the LoRA adapter parameters in f_ϕ are learned. This approach is parameter-efficient, reducing memory and computational overhead as we only need to keep g_θ in memory once. It also allows f_ϕ to inherit useful inductive biases from g_θ's architecture. For conditional models $g_\theta(\cdot|c)$, $f_\phi(\mathbf{x}_0|c)$ can similarly leverage learned conditional representations by applying LoRA to conditioning pathways, e.g. the learned text-conditioning of a text-to-image model.

Algorithm 1 HyperNoise

Input: g_θ (distilled generative Model), r (reward fn), Optional $\mathcal{C} = \{c_i\}_{i=1}^N$ (condition dataset)
Initialize Noise Hypernetwork $f_\phi(\cdot) = \mathbf{0}$ through LoRA weights ϕ applied on top of g_θ
while training **do**
  Sample noise $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$, $\mathrm{c} = \varnothing$
  if $\mathcal{C}$ **then**
    Sample condition $\mathrm{c} \sim \mathcal{C}$
  Predict modulated noise $\Delta\mathbf{x}_0 = f_\phi(\mathbf{x}_0, \mathrm{c})$
  Generate $\mathbf{x}_1 = g_\theta(\mathbf{x}_0 + \Delta\mathbf{x}_0, \mathrm{c})$
  Compute Loss $\mathcal{L}_{\text{noise}}(\phi) = \frac{1}{2}\|\Delta\mathbf{x}_0\|^2 - r(\mathbf{x}_1)$
  Gradient step on $\nabla_\phi \mathcal{L}_{\text{noise}}(\phi)$
return Noise Hypernetwork LoRA weights ϕ

Initialization. We propose to initialize f_ϕ such that its output $f_\phi(\cdot) = \mathbf{0}$. This is crucial for training stability and supports the validity of the L_2 approximation for $D_{\text{KL}}(p_0^\phi \| p_0)$ (Equation 10) from the start of training. Specifically, we modify the final layer of f_ϕ to output only the LoRA-generated perturbation, which are initialized to output $\mathbf{0}$ (this is achieved by setting the second LoRA matrix, often denoted B, to zero), *without* using any frozen base weights. This ensures that at initialization $f_\phi(\cdot) = \mathbf{0}$ such that effectively $\hat{\mathbf{x}}_0 = \mathbf{x}_0 + f_\phi(\mathbf{x}_0) = \mathbf{x}_0$, making $p_0^\phi = p_0$.

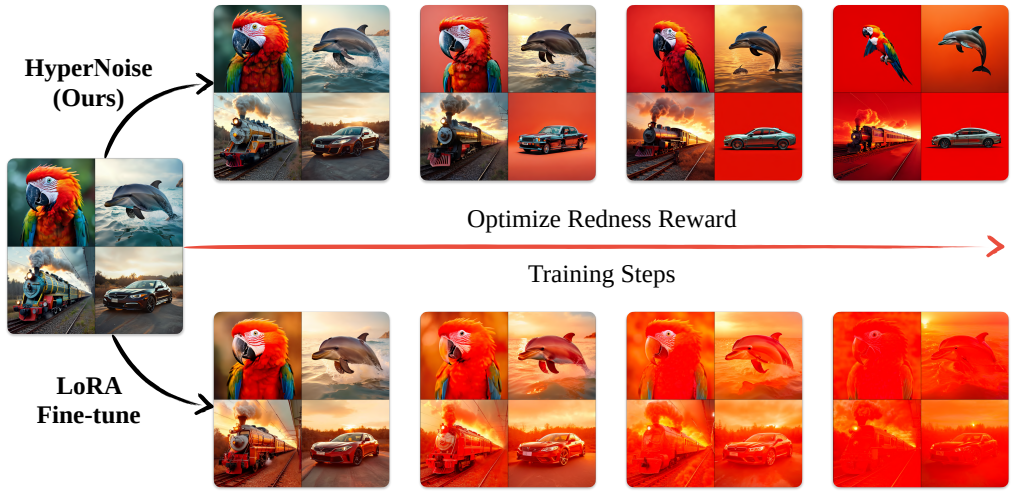

Figure 3: An illustrative example of optimizing for learning the tilted distribution with an image redness reward. We show direct LoRA fine-tuning of SANA-Sprint [13] in comparison to training a noise hypernetwork with our proposed objective. Notably, when training with our objective, the model optimizes the desired reward while staying considerably closer to p^{base}, as showcased by the model not diverging from the image manifold, unlike in direct LoRA fine-tuning.

4 Experiments

Our experimental evaluation is designed to assess the efficacy of our objective for the popular setting of text-to-image (T2I) models. We benchmark the noise hypernetwork against established methods, primarily direct LoRA fine-tuning of the base generative model [74], and investigate its capacity to match or recover the performance gains typically associated with test-time scaling techniques like ReNO [22], but through a post-training approach. To clearly delineate these comparisons, we structure our experiments as follows: We first present an illustrative experiment employing a *"redness reward"*. This controlled setting is designed to demonstrate the advantages of our training objective, particularly its ability to optimize for a target reward while mitigating divergence from the base model's learned data manifold p^{base}. Subsequently, we extend our evaluation to more complex and practical scenarios, focusing on aligning generative models with *human-preference reward models*.

4.1 Redness Reward

We begin our evaluation with the goal of learning the tilted distribution (Equation 3) given a redness reward. This metric helps showcase the potential underlying issue of directly fine-tuning the generation model g_ϕ (a fine-tuned variant of the base model g_θ). For this experiment, the redness reward $r(\mathbf{x})$ is defined as the difference between the red channel intensity and the average of the green and blue channel intensities: $r(\mathbf{x}) = \mathbf{x}^0 - \frac{1}{2}(\mathbf{x}^1 + \mathbf{x}^2)$, where $\mathbf{x}^i$ denotes the i-th color channel of the generated image $\mathbf{x}$ and is used to train the recent SANA-Sprint [13] model, for full details see Appendix B.1.

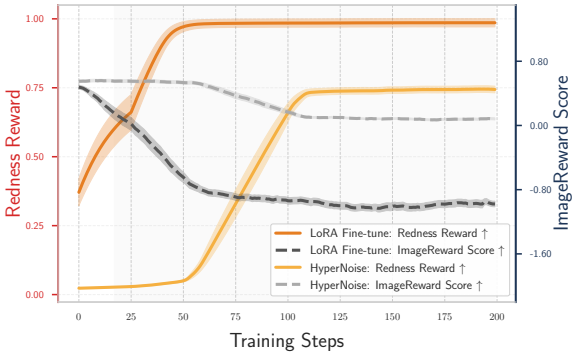

Figure 4: Trade-off between the redness reward objective and an image quality metric, ImageReward, for direct fine-tuning and Noise Hypernetworks. As opposed to direct fine-tuning, our proposed method optimizes the redness objective while not significantly dropping image quality as indicated by the ImageReward score.

The primary concern with directly fine-tuning g_ϕ to maximize a reward is the risk of significant deviation from the original data distribution p^{base}. This deviation can lead to a high

Table 1: **Quantitative Results on GenEval**. Our Noise Hypernetwork combined with (1) SD-Turbo [83], (2) SANA-Sprint 0.6B [13], and Flux-Schnell consistently improving results while maintaining few-step denoising, fast inference, and minimal memory overhead. Results from best-of-n sampling [44], ReNO [22], and prompt optimization [4, 61] are greyed out to provide a reference upper-bound in terms of applying optimization at inference. Prompt optimization † additionally requires a significant amount of calls to an LLM, either locally or through an API.

Model	Params (B)	Time (s) ↓	Mean ↑	Single↑	Two↑	Counting↑	Colors↑	Position↑	Attribution↑
SD v2.1 [79]	0.8	1.9	0.50	0.98	0.51	0.44	0.85	0.07	0.17
SDXL [73]	2.6	6.9	0.55	0.98	0.74	0.39	0.85	0.15	0.23
DPO-SDXL [98]	2.6	6.9	0.59	0.99	0.84	0.49	0.87	0.13	0.24
Hyper-SDXL [78]	2.6	0.3	0.56	1.00	0.76	0.43	0.87	0.10	0.21
Flux-dev	12.0	23.0	0.68	0.99	0.85	0.74	0.79	0.21	0.48
SD3-Medium [21]	2.0	4.4	0.70	1.00	0.90	0.72	0.87	0.31	0.66
SD-Turbo [83]	0.8	0.2	0.49	0.99	0.51	0.38	0.85	0.07	0.14
+ HyperNoise	1.1	0.3	0.57	0.99	0.65	0.50	0.89	0.14	0.22
+ Prompt Optimization [4, 61]	0.8†	95.0†	0.59	0.99	0.76	0.53	0.88	0.10	0.28
+ Best-of-N [44]	0.8	10.0	0.60	1.00	0.78	0.55	0.88	0.10	0.29
+ ReNO [22]	0.8	20.0	0.63	1.00	0.84	0.60	0.90	0.11	0.36
SANA-Sprint [13]	0.6	0.2	0.70	1.00	0.80	0.64	0.86	0.41	0.51
+ HyperNoise	0.9	0.3	0.75	1.00	0.88	0.71	0.85	0.51	0.55
+ Prompt Optimization [4, 61]	0.6†	95.0†	0.75	0.99	0.91	0.82	0.89	0.36	0.56
+ Best-of-N [44]	0.6	15.0	0.79	0.99	0.92	0.72	0.91	0.53	0.65
+ ReNO [22]	0.6	30.0	0.81	0.99	0.93	0.74	0.92	0.60	0.67
FLUX-Schnell (4-step)	12.0	0.7	0.68	0.99	0.88	0.66	0.78	0.27	0.48
+ HyperNoise	13.0	0.9	0.72	0.99	0.93	0.67	0.83	0.30	0.59
+ ReNO [22]	12.0	40.0	0.76	0.99	0.94	0.70	0.86	0.39	0.65

$D_{\mathrm{KL}}(p^{\phi} \| p^{\text{base}})$, where p^{ϕ} is the distribution induced by the fine-tuned model g_{ϕ}. Such a divergence often manifests as a degradation in overall image quality or a loss of diversity, even if the target reward (e.g. redness) is achieved. Figure 4 quantitatively illustrates this trade-off by plotting the redness reward against a general image quality metric (ImageReward), comparing our Noise Hypernetwork approach with LoRA fine-tuning, while Figure 3 visually corroborates these results.

4.2 Human-preference Reward Models

Implementation Details. We conduct our primary experiments on aligning text-to-image models with human preferences using SD-Turbo [83], SANA-Sprint [13] and FLUX-Schnell. Notably, SANA-Sprint and FLUX-Schnell exhibit strong prompt-following capabilities competitive with proprietary models, making them robust base models for our evaluations. For the reward signal $r(\cdot)$ essential to our objective (Equation 11) and for the direct fine-tuning baseline, we utilize the exact same composition of reward models proposed in ReNO [22] consisting of ImageReward [103], HPSv2.1 [101], Pickscore [48], and a CLIP-score. For the noise hypernetwork, we use a LoRA [35] module on the base distilled model with the proposed initialization as described in Section 3.2. Training for the noise hypernetwork is performed using ~70k prompts from Pick-a-Picv2 [48], T2I-Compbench train set [37], and Attribute Binding (ABC-6K) [25] prompts. Our evaluations of the trained models are performed on GenEval [26], ensuring that the training and evaluation prompts do not have any overlap, measuring the generalization of the noise hypernetwork to unseen prompts. We mainly compare HyperNoise with three different test-time techniques: Best-of-N sampling [44, 59], ReNO [22], and LLM-based prompt optimization [4, 61]. As detailed in Table 1, all of these incur significantly increased computational costs at test-time, ranging from 33× to 300× slower inference compared to HyperNoise, making them impractical for large-scale deployment where efficiency is paramount. Full experimental details are provided in Appendix B.2.

Quantitative Results. We present our main quantitative results on the GenEval benchmark in Table 1. Our Noise Hypernetwork training scheme consistently yields significant performance gains across all model scales while maintaining near-baseline inference costs. When applied to SD-Turbo, our method nearly recovers most of the improvements from inference-time noise optimization, achieving an overall GenEval performance of 0.57 that even surpasses SDXL (which has 2× more parameters and 25× NFEs), clearly highlighting the benefits from our noise hypernetwork training. With SANA-Sprint, we observe consistent improvements (0.75 vs 0.70) over the base model, achieving the same performance as LLM-based prompt optimization while being 300× faster, and recovering about half of the performance gains achieved by ReNO with minimal GPU memory overhead. Notably, we observe

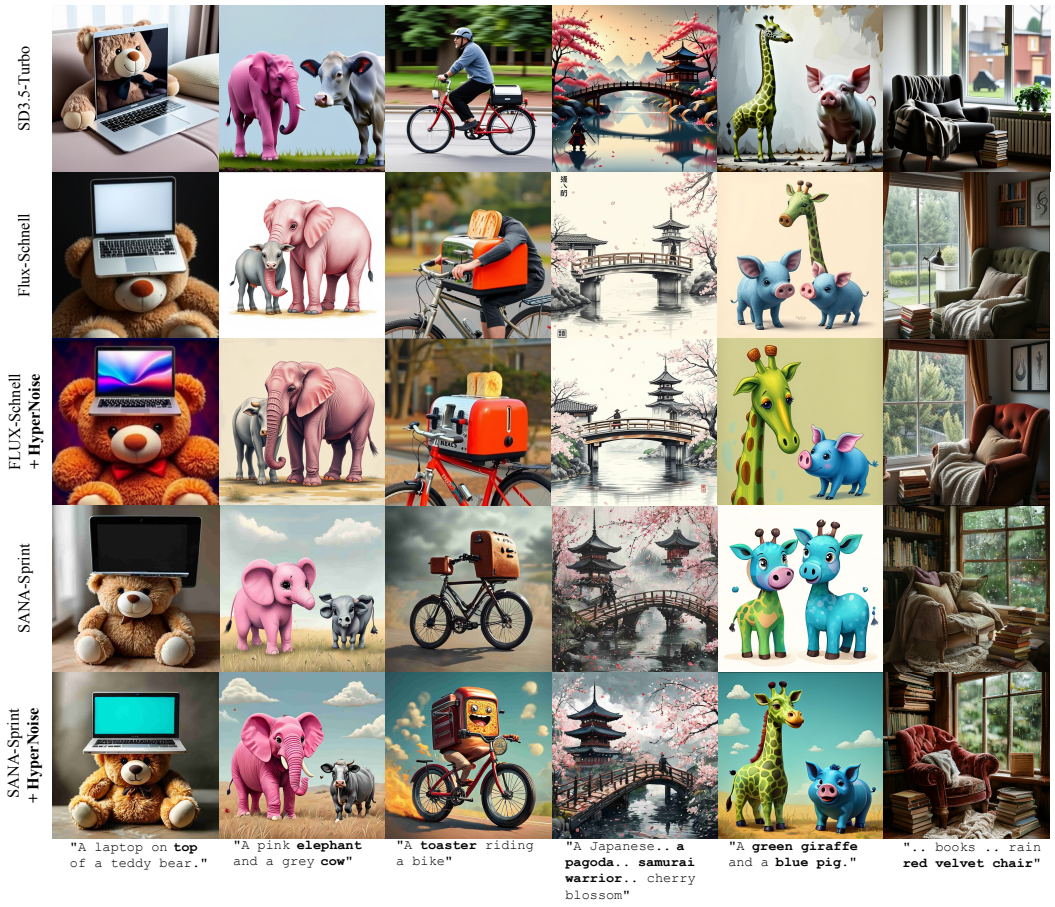

Figure 5: Qualitative comparison our proposed noise hypernetwork with popular distilled models such as Flux-Schnell, SD3.5-Turbo, SANA-Sprint for 4-step generation. Both SANA-Sprint and FLUX-Schnell share the initial noise for the base and HyperNoise generation.

similar trends for the larger 12B parameter FLUX-Schnell, where we again recover substantial performance gains (0.71 vs 0.68) while maintaining the efficiency advantages that make our approach practical for real-world deployment. The consistent efficiency gains across model scales demonstrate that our approach successfully amortizes the optimization cost during training, enabling high-quality generation without the prohibitive test-time computational overhead of alternative methods.

Superiority over Direct Fine-tuning and Multi-Step Generalization. In Tab. 8, we show the generalization of our training on multi-step inference despite being trained only with one-step generation. We obtain consistent improvements over SANA-Sprint for one, two, and four step generation. Notably, our model with one-step generation noticeably outperforms SANA-Sprint with 4 steps. We also illustrate how direct fine-tuning of the base model with the *same* reward objective can lead to significantly worse results, highlighting the necessity of preventing "reward-hacking" in a principled fashion. We visualize this in Appendix C.4, where we observe similar patterns as previous works for reward-hacking [14, 53, 91].

Table 2: Mean GenEval results for SANA-Sprint highlighting generalization across inference timesteps of our Noise Hypernetwork and failure of direct LoRA fine-tuning.

SANA-Sprint [13]	**NFEs**	**GenEval Mean↑**
One-step	1	0.70
+ LoRA fine-tune [14, 74, 103]	1	0.67
+ HyperNoise	2	**0.75**
Two-step	2	0.72
+ LoRA fine-tune [14, 74, 103]	2	0.66
+ HyperNoise	3	**0.76**
Four-step	4	0.73
+ LoRA fine-tune [14, 74, 103]	4	0.62
+ HyperNoise	5	**0.77**

Qualitative Results. We illustrate examples of generated images in Fig. 5 showing our method applied to both SANA-Sprint and FLUX-Schnell, alongside comparisons to SD3.5-Turbo. Our noise hypernetwork demonstrates consistent improvements across

both base models. For SANA-Sprint, the improvements are substantial: we observe both correction of generation artifacts and significantly enhanced prompt following for complex compositional requests. When applied to the already high-quality FLUX-Schnell, our method still provides noticeable improvements in detail quality and prompt adherence, demonstrating that our approach can enhance even strong base models while maintaining the efficiency advantages essential for practical deployment.

5 Related Work

Test-Time Scaling. The paradigm of test-time scaling has yielded remarkable breakthroughs, with models allocating additional computation during inference to solve increasingly complex problems. In language models, this has manifested through process reward models [60, 84, 109] and reinforcement learning from verifiable rewards [49, 64], leading to systems like o1 [40] and DeepSeek-R1 [28]. Beyond scaling denoising steps in diffusion models, test-time techniques improve generation quality by finding better initial noise or refining intermediate states during inference, often guided by pre-trained reward models. These methods fall into two categories: search-based approaches [44, 59, 92, 93] that evaluate multiple candidates, and optimization-based approaches [6, 19, 29, 46, 67, 90, 97] that iteratively refine noise or latents through gradient descent. Although both strategies achieve significant quality improvements, they introduce substantial computational overhead, with generation times frequently exceeding several minutes per image.

Aligning Diffusion Models with Rewards. Reward models [48, 101, 102, 103, 107] have been effectively used to directly fine-tune diffusion models using reinforcement learning [8, 12, 17, 24, 108] or direct reward fine-tuning [14, 18, 41, 50, 53, 74, 75, 103]. Alternatively, Direct Preference Optimization (DPO) [34, 45, 52, 77, 98] learns from paired comparisons rather than absolute rewards. A particular instance of reward fine-tuning [18, 89, 91] analyzes learning the reward-tilted distribution through stochastic optimal control. Uehara et al. [91] fine-tune continuous-time diffusion models by jointly optimizing both the drift term and initial noise distribution, but their SDE-based formulation requires continuous-time dynamics and backpropagation through the full sampling process, making it computationally expensive and inapplicable to step-distilled models. For distilled models, concurrent work [42, 62, 68] has explored preference tuning, though without the theoretical foundation for sampling from the target-tilted distribution that our approach provides. Wagenmaker et al. [96] apply similar noise-space optimization principles to diffusion policies in robotic control, demonstrating efficient adaptation while preserving pretrained capabilities across diverse domains.

Hypernetworks. Auxiliary models [30] that predict parameters of task-specific models have been used for vision [2, 31] and language tasks [39, 63, 72]. For generative models, they have been used to generate weights through diffusion [20, 99] and to speed up personalization [2, 81]. NoiseRefine [1] and Golden Noise [110] train hypernetworks to predict initial noise to replace classifier-free guidance or find reliable generations by selecting 'ground-truth' noise pairs as supervision, as opposed to the end-to-end training in our framework. Work on diffusion priors [5, 16, 23, 27, 65, 82] also adapts the noise distribution, but these approaches modify the training process rather than enabling post-hoc adaptation of pre-trained models. Concurrently, Venkatraman et al. [94] explore sampling from reward-tilted distributions for arbitrary generators, but our work demonstrates this approach at scale with comprehensive evaluation across multiple model architectures and unseen prompt distributions.

6 Conclusion

In this work we provide fresh perspective for post-training diffusion models through the introduction of HyperNoise, a noise modulation strategy. Our principled training objective coupled with the efficient training scheme is able to achieve a meaningful improvements in performance across multiple models while avoiding 'reward-hacking'. We hope that our efficient and effective solution for aligning diffusion models with downstream objectives finds use across a wide variety of domains and use cases, especially in cases where test-time optimization would be prohibitively expensive.

Limitations. Preference-tuning diffusion models heavily relies on strong pre-trained base models and meaningful reward signals. While constant improvements are made to develop better pre-trained base models, specific focus should be devoted to improving reward models that can give meaningful feedback on a variety of aspects that are important for high-quality generation.

Acknowledgements

This work was partially funded by the ERC (853489 - DEXIM) and the Alfried Krupp von Bohlen und Halbach Foundation, which we thank for their generous support. Shyamgopal Karthik thanks the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for support. Luca Eyring would like to thank the European Laboratory for Learning and Intelligent Systems (ELLIS) PhD program for support.

References

[1] Donghoon Ahn, Jiwon Kang, Sanghyun Lee, Jaewon Min, Minjae Kim, Wooseok Jang, Hyoungwon Cho, Sayak Paul, SeonHwa Kim, Eunju Cha, et al. A noise is worth diffusion guidance. *arXiv preprint arXiv:2412.03895*, 2024.

[2] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *CVPR*, 2022.

[3] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR*, 2023.

[4] Kumar Ashutosh, Yossi Gandelsman, Xinlei Chen, Ishan Misra, and Rohit Girdhar. Llms can see and hear without any training, 2025. URL https://arxiv.org/abs/2501.18096.

[5] Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. Neural flow diffusion models: Learnable forward process for improved diffusion modelling, 2025. URL https://arxiv.org/abs/2404.12940.

[6] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. In *ICML*, 2024.

[7] Samarth Bhatia and Felix Dangel. Lowering pytorch's memory consumption for selective differentiation. 2024.

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

[9] Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025. URL https://arxiv.org/abs/2505.18825.

[10] Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2025. URL https://arxiv.org/abs/2406.07507.

[11] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. In *SIGGRAPH*, 2023.

[12] Chaofeng Chen, Annan Wang, Haoning Wu, Liang Liao, Wenxiu Sun, Qiong Yan, and Weisi Lin. Enhancing diffusion models with text-encoder reinforcement learning. In *ECCV*, 2024.

[13] Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Enze Xie, and Song Han. Sana-sprint: One-step diffusion with continuous-time consistency distillation. *arXiv preprint arXiv:2503.09641*, 2025.

[14] Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *ICLR*, 2024.

[15] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0471241954.

[16] Cecilia Curreli, Dominik Muhle, Abhishek Saroha, Zhenzhang Ye, Riccardo Marin, and Daniel Cremers. Nonisotropic gaussian diffusion for realistic 3d human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[17] Fei Deng, Qifei Wang, Wei Wei, Matthias Grundmann, and Tingbo Hou. Prdp: Proximal reward difference prediction for large-scale reward finetuning of diffusion models. In *CVPR*, 2024.

[18] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

[19] Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models, 2017. URL https://arxiv.org/abs/1711.05772.

[20] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *ICCV*, 2023.

[21] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

[22] Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. In *NeurIPS*, 2024.

[23] Luca Eyring, Dominik Klein, Théo Uscidda, Giovanni Palla, Niki Kilbertus, Zeynep Akata, and Fabian J Theis. Unbalancedness in neural monge maps improves unpaired domain translation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2UnCj3jeao.

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

[25] Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. In *ICLR*, 2023.

[26] Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023.

[27] Sang gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior, 2022. URL https://arxiv.org/abs/2106.06406.

[28] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[29] Xiefan Guo, Jinlin Liu, Miaomiao Cui, Jiankai Li, Hongyu Yang, and Di Huang. Initno: Boosting text-to-image diffusion models via initial noise optimization. In *CVPR*, 2024.

[30] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[31] Eric Hedlin, Munawar Hayat, Fatih Porikli, Kwang Moo Yi, and Shweta Mahajan. Hypernet fields: Efficiently training hypernetworks without ground truth by learning weight trajectories. *arXiv preprint arXiv:2412.17040*, 2024.

[32] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning, 2022.

[33] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[34] Jiwoo Hong, Sayak Paul, Noah Lee, Kashif Rasul, James Thorne, and Jongheon Jeong. Margin-aware preference optimization for aligning diffusion models without reference. *arXiv preprint arXiv:2406.06424*, 2024.

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

[36] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

[37] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. In *NeurIPS*, 2023.

[38] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

[39] Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. Hint: hypernetwork instruction tuning for efficient zero-& few-shot generalisation. *arXiv preprint arXiv:2212.10315*, 2022.

[40] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[41] Rohit Jena, Ali Taghibakhshi, Sahil Jain, Gerald Shen, Nima Tajbakhsh, and Arash Vahdat. Elucidating optimal reward-diversity tradeoffs in text-to-image diffusion models. *arXiv preprint arXiv:2409.06493*, 2024.

[42] Zhiwei Jia, Yuesong Nan, Huixi Zhao, and Gengdai Liu. Reward fine-tuning two-step diffusion models via learning differentiable latent-space surrogate reward. *arXiv preprint arXiv:2411.15247*, 2024.

[43] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

[44] Shyamgopal Karthik, Karsten Roth, Massimiliano Mancini, and Zeynep Akata. If at first you don't succeed, try, try again: Faithful diffusion-based text-to-image generation by selection. *arXiv preprint arXiv:2305.13308*, 2023.

[45] Shyamgopal Karthik, Huseyin Coskun, Zeynep Akata, Sergey Tulyakov, Jian Ren, and Anil Kag. Scalable ranked preference optimization for text-to-image generation. *arXiv preprint arXiv:2410.18013*, 2024.

[46] Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. In *CVPR*, 2024.

[47] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *NeurIPS*, 2021.

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

[49] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[50] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

[51] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, 2022.

[52] Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, and Kazuki Kozuka. Aligning diffusion models by optimizing human utility. *arXiv preprint arXiv:2404.04465*, 2024.

[53] Yanyu Li, Xian Liu, Anil Kag, Ju Hu, Yerlan Idelbayev, Dhritiman Sagar, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Textcraftor: Your text encoder can be image quality controller. In *CVPR*, 2024.

[54] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. *arXiv preprint arXiv:2404.01291*, 2024.

[55] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

[56] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[57] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[58] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[59] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[60] Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.

[61] Oscar Mañas, Pietro Astolfi, Melissa Hall, Candace Ross, Jack Urbanek, Adina Williams, Aishwarya Agrawal, Adriana Romero-Soriano, and Michal Drozdzal. Improving text-to-image consistency via automatic prompt optimization, 2024. URL https://arxiv.org/abs/2403.17804.

[62] Zichen Miao, Zhengyuan Yang, Kevin Lin, Ze Wang, Zicheng Liu, Lijuan Wang, and Qiang Qiu. Tuning timestep-distilled diffusion model using pairwise sample optimization. *arXiv preprint arXiv:2410.03190*, 2024.

[63] Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

[64] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[65] Beatrix Miranda Ginn Nielsen, Anders Christensen, Andrea Dittadi, and Ole Winther. Diffenc: Variational diffusion with a learned encoder. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8nxy1bQWTG.

[66] Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas Bryan. Ditto-2: Distilled diffusion inference-time t-optimization for music generation. *arXiv preprint arXiv:2405.20289*, 2024.

[67] Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J. Bryan. Ditto: Diffusion inference-time t-optimization for music generation, 2024. URL https://arxiv.org/abs/2401.12179.

[68] Owen Oertell, Jonathan D Chang, Yiyi Zhang, Kianté Brantley, and Wen Sun. Rl for consistency models: Faster reward guided text-to-image generation. *arXiv preprint arXiv:2404.03673*, 2024.

[69] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[70] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2021. URL https://arxiv.org/abs/1912.02762.

[71] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for compositional text-to-image synthesis. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[72] Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. Hypertuning: Toward adapting large language models without back-propagation. In *ICML*, pages 27854–27875, 2023.

[73] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

[74] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

[75] Mihir Prabhudesai, Russell Mendonca, Zheyang Qin, Katerina Fragkiadaki, and Deepak Pathak. Video diffusion alignment via reward gradients. *arXiv preprint arXiv:2407.08737*, 2024.

[76] 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 *ICML*, 2021.

[77] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*, 2023.

[78] Yuxi Ren, Xin Xia, Yanzuo Lu, Jiacheng Zhang, Jie Wu, Pan Xie, Xing Wang, and Xuefeng Xiao. Hyper-sd: Trajectory segmented consistency model for efficient image synthesis, 2024.

[79] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[80] Litu Rout, Yujia Chen, Nataniel Ruiz, Abhishek Kumar, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Rb-modulation: Training-free personalization of diffusion models using stochastic optimal control. *arXiv preprint arXiv:2405.17401*, 2024.

[81] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Wei Wei, Tingbo Hou, Yael Pritch, Neal Wadhwa, Michael Rubinstein, and Kfir Aberman. Hyperdreambooth: Hypernetworks for fast personalization of text-to-image models. In *CVPR*, 2024.

[82] Subham Sekhar Sahoo, Aaron Gokaslan, Christopher De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=loMa99A4p8.

[83] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[84] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[85] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[86] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[87] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023.

[88] Aravindan Sundaram, Ujjayan Pal, Abhimanyu Chauhan, Aishwarya Agarwal, and Srikrishna Karanam. Cocono: Attention contrast-and-complete for initial noise optimization in text-to-image synthesis. *arXiv preprint arXiv:2411.16783*, 2024.

[89] Wenpin Tang. Fine-tuning of diffusion models via stochastic control: entropy regularization and beyond, 2024. URL https://arxiv.org/abs/2403.06279.

[90] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. *arXiv preprint arXiv:2405.18881*, 2024.

[91] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control, 2024. URL https://arxiv.org/abs/2402.15194.

[92] Masatoshi Uehara, Xingyu Su, Yulai Zhao, Xiner Li, Aviv Regev, Shuiwang Ji, Sergey Levine, and Tommaso Biancalani. Reward-guided iterative refinement in diffusion models at test-time with applications to protein and dna design, 2025. URL https://arxiv.org/abs/2502.14944.

[93] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review, 2025. URL https://arxiv.org/abs/2501.09685.

[94] Siddarth Venkatraman, Mohsin Hasan, Minsu Kim, Luca Scimeca, Marcin Sendera, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. *arXiv preprint arXiv:2502.06999*, 2025.

[95] Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[96] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

[97] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance. In *ICCV*, 2023.

[98] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *CVPR*, 2024.

[99] Kai Wang, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

[100] Zijie J Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv preprint arXiv:2210.14896*, 2022.

[101] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[102] Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Better aligning text-to-image models with human preference. In *ICCV*, 2023.

[103] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *NeurIPS*, 2023.

[104] Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. Long-clip: Unlocking the long-text capability of clip. In *European Conference on Computer Vision*, pages 310–325. Springer, 2024.

[105] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*, 2018.

[106] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[107] Sixian Zhang, Bohan Wang, Junqiang Wu, Yan Li, Tingting Gao, Di Zhang, and Zhongyuan Wang. Learning multi-dimensional human preference for text-to-image generation. In *CVPR*, 2024.

[108] Yinan Zhang, Eric Tzeng, Yilun Du, and Dmitry Kislyuk. Large-scale reinforcement learning for diffusion models. In *ECCV*, 2024.

[109] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

[110] Zikai Zhou, Shitong Shao, Lichen Bai, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. *arXiv preprint arXiv:2411.09502*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract clearly demarcates the aspirational goals for the field along with the results targeted by the paper and the theory/results achieve the conrete claims made in the abstract

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, there is a clear discussion of limitations of the both the theory and experiments in the paper.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Yes, both the main paper and the appendix contains a clear discussion of the assumptions made to achieve the theoretical results shown in the paper.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, the main paper already provides the most crucial details required to reproduce the results in the paper. The supplementary material provides complete clarity. Additionally, we utilize only open-source models and data for our experiments, enhancing reproducing further.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, we released the entire codebase, training scripts and data to reproduce our results under https://github.com/ExplainableML/HyperNoise.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, complete experimental details for the training and test are provided in the supplementary material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the most significant evaluations (e.g. Tab. 1), results are averaged over 4 seeds as is standard practice for the GenEval benchmark.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, our main experiments were performed on an academic compute budget of 6H100 GPUs for 3 days, which is clearly stated in the experiment section.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: All the authors have read the NeurIPS Code of Ethics and agree that all aspects of our research conforms with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Image generation models have become widespread over the past 3 years and we believe that their overall societal impact is relatively well-understood at this point. We do not foresee major additional societal impact specifically from our work. We discuss these details in the appendix.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our method aims to improve the alignment of existing open-source methods. To the extent that they are publicly available without major safeguards, we beleieve that there are no other major additional safeguards that are required for our work.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have appropriately cited existing assets.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We shall provide proper documentation for our released models and code and the time of the public release.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No experiments with crowdsourcing or experiments with human subjects were done in the main paper.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No IRB approvals or equivalent was required for the experiments in our paper.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: No non-standard use of LLMs beyond typographical and grammatical edits of the paper.

Appendix

The Appendix is organized as follows:

- Section A provides all of our theoretical derivations.
- Section B outlines the implementation details.
- Section C presents further quantitative and qualitative analysis.

A Theoretical Derivations

This section provides rigorous derivations for the reward-tilted noise distribution and our tractable training objective. We include a temperature parameter $\alpha > 0$ for completeness, though the main paper uses $\alpha = 1$.

A.1 Setup and Standing Assumptions

Let $p_0(\mathbf{x}_0)$ denote the standard Gaussian density on $\mathbb{R}^d$:

$$p_0(\mathbf{x}_0) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\|\mathbf{x}_0\|^2\right). \tag{13}$$

Let $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ be the pre-trained distilled generator and $r : \mathbb{R}^d \to \mathbb{R}$ be the reward function.

Standing Assumptions. Throughout this section, we assume:

1. The generator $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is measurable.
2. The reward function $r : \mathbb{R}^d \to \mathbb{R}$ is measurable and $\mathbb{E}_{\mathbf{x}_0 \sim p_0}[e^{r(g_\theta(\mathbf{x}_0))/\alpha}] < \infty$ for our chosen temperature $\alpha > 0$.
3. For any $\mathbf{x} \in \text{Range}(g_\theta)$, the preimage set $g_\theta^{-1}(\{\mathbf{x}\})$ has a well-defined measure structure.

These assumptions are mild and realistic for neural network generators.

Pushforward Measure and Base Distribution. The base generator density $p^{\text{base}}(\mathbf{x})$ is the density of the pushforward measure $(g_\theta)_\sharp P_0$, where P_0 is the probability measure corresponding to $p_0(\mathbf{x}_0)$. Formally, $(g_\theta)_\sharp P_0$ is defined such that for any Borel set $A \subset \mathbb{R}^d$:

$$((g_\theta)_\sharp P_0)(A) = P_0(g_\theta^{-1}(A)) = \int_{g_\theta^{-1}(A)} p_0(\mathbf{x}_0) d\mathbf{x}_0. \tag{14}$$

Under our standing assumptions, the density $p^{\text{base}}(\mathbf{x})$ can be written using the Dirac delta as:

$$p^{\text{base}}(\mathbf{x}) = \int_{\mathbb{R}^d} \delta(\mathbf{x} - g_\theta(\mathbf{x}_0)) p_0(\mathbf{x}_0) d\mathbf{x}_0. \tag{15}$$

Note that in the main text, with slight abuse of notation, we write $(g_\theta)_\sharp p_0$ instead of $(g_\theta)_\sharp P_0$.

KL Divergence. The Kullback-Leibler (KL) divergence between two probability densities $q(\mathbf{v})$ and $p(\mathbf{v})$ is defined as:

$$D_{\text{KL}}(q\|p) := \int_{\mathbb{R}^d} q(\mathbf{v}) \log \frac{q(\mathbf{v})}{p(\mathbf{v})} d\mathbf{v}, \tag{16}$$

provided the integral exists and is finite.

A.2 The Reward-Tilted Output Distribution

The primary goal is to align the generator with the reward function $r(\mathbf{x})$ by targeting a *reward-tilted output distribution* $p^\star(\mathbf{x})$ that upweights high-reward samples while maintaining similarity to the base distribution.

Definition 1 (Reward-Tilted Output Distribution)**.** The target reward-tilted output density $p^\star(\mathbf{x})$ is defined by upweighting samples from the base generator density $p^{\text{base}}(\mathbf{x})$ according to the reward $r(\mathbf{x})$:

$$p^\star(\mathbf{x}) := \frac{1}{Z^\star} p^{\text{base}}(\mathbf{x}) \exp\left(\frac{r(\mathbf{x})}{\alpha}\right), \tag{17}$$

where $Z^\star$ is the normalization constant ensuring $p^\star(\mathbf{x})$ integrates to one:

$$Z^\star := \int_{\mathbb{R}^d} p^{\text{base}}(\mathbf{x}) \exp\left(\frac{r(\mathbf{x})}{\alpha}\right) d\mathbf{x}. \tag{18}$$

Under our standing assumptions, we have $Z^\star < \infty$. We denote $P^\star$ as the probability measure corresponding to $p^\star$.

Interpretation. The temperature parameter $\alpha > 0$ controls the strength of the reward signal:

- When $\alpha \to \infty$, we have $p^\star(\mathbf{x}) \to p^{\text{base}}(\mathbf{x})$ (no reward influence)
- When $\alpha \to 0$, the distribution concentrates on high-reward regions
- $\alpha = 1$ provides a natural balance between reward optimization and staying close to the base distribution

Objective for Fine-Tuning Generator Parameters. If we aim to fine-tune the generator parameters from θ to ϕ, leading to a new output density $p^\phi(\mathbf{x})$ (when input is from $p_0(\mathbf{x}_0)$), a principled approach is to minimize the KL divergence $D_{\text{KL}}(p^\phi \| p^\star)$.

Proposition 2 (KL Objective for Generator Fine-tuning)**.** *Minimizing $D_{\text{KL}}(p^\phi \| p^\star)$ with respect to the generator parameters ϕ is equivalent to minimizing:*

$$J_{gen}(\phi) = D_{\text{KL}}(p^\phi \| p^{base}) - \frac{1}{\alpha} \mathbb{E}_{\mathbf{x} \sim p^\phi}[r(\mathbf{x})]. \tag{19}$$

Proof. Using the definition of $p^\star(\mathbf{x})$ from Equation (17):

$$\begin{aligned}
D_{\text{KL}}(p^\phi \| p^\star) &= \int_{\mathbb{R}^d} p^\phi(\mathbf{x}) \log \frac{p^\phi(\mathbf{x})}{p^\star(\mathbf{x})} d\mathbf{x} \\
&= \int_{\mathbb{R}^d} p^\phi(\mathbf{x}) \log \frac{p^\phi(\mathbf{x}) Z^\star}{p^{\text{base}}(\mathbf{x}) \exp\left(\frac{r(\mathbf{x})}{\alpha}\right)} d\mathbf{x} \\
&= \int_{\mathbb{R}^d} p^\phi(\mathbf{x}) \left(\log \frac{p^\phi(\mathbf{x})}{p^{\text{base}}(\mathbf{x})} - \frac{r(\mathbf{x})}{\alpha} + \log Z^\star \right) d\mathbf{x} \\
&= D_{\text{KL}}(p^\phi \| p^{\text{base}}) - \frac{1}{\alpha} \mathbb{E}_{\mathbf{x} \sim p^\phi}[r(\mathbf{x})] + \log Z^\star.
\end{aligned} \tag{20}$$

Since $\log Z^\star$ is constant with respect to ϕ, minimizing $D_{\text{KL}}(p^\phi \| p^\star)$ is equivalent to minimizing $J_{\text{gen}}(\phi)$. □

Challenges with Direct Generator Fine-tuning. While Proposition 2 provides a theoretically sound objective, directly optimizing it for distilled models poses significant challenges:

1. **Intractable KL term:** Computing $D_{\text{KL}}(p^\phi \| p^{\text{base}})$ requires evaluating densities of high-dimensional neural network generators, which involves intractable Jacobian determinants
2. **No continuous-time structure:** Unlike full diffusion models, distilled generators often lack explicit SDE/ODE structure that would enable techniques from stochastic optimal control

3. **Reward hacking:** Without proper regularization, optimization can lead to adversarial exploitation of the reward model, generating unrealistic samples that achieve high reward scores

These challenges motivate our alternative approach of modifying the input noise distribution while keeping the generator fixed, which we develop in the next section.

A.3 The Reward-Tilted Noise Distribution

An alternative to modifying the generator g_θ is to modify the input noise density $p_0(\mathbf{x}_0)$ while keeping g_θ fixed. We seek an optimal *tilted noise density* $p_0^\star(\mathbf{x}_0)$ such that its pushforward through g_θ results in the target output density $p^\star(\mathbf{x})$.

Normalization Constant in Noise Space. First, we show that the normalization constant $Z^\star$ from Equation (18) can be expressed as an integral over the noise space. Using Equation (15) for $p^{\text{base}}(\mathbf{x})$ in the definition of $Z^\star$:

$$\begin{aligned}
Z^\star &= \int_{\mathbb{R}^d} \exp\left(\frac{r(\mathbf{x})}{\alpha}\right)\left(\int_{\mathbb{R}^d} \delta(\mathbf{x}-g_\theta(\mathbf{x}_0'))p_0(\mathbf{x}_0')d\mathbf{x}_0'\right)d\mathbf{x} \\
&= \int_{\mathbb{R}^d}\left(\int_{\mathbb{R}^d} \exp\left(\frac{r(\mathbf{x})}{\alpha}\right)\delta(\mathbf{x}-g_\theta(\mathbf{x}_0'))d\mathbf{x}\right)p_0(\mathbf{x}_0')d\mathbf{x}_0' \quad \text{(Fubini's theorem)} \\
&= \int_{\mathbb{R}^d} \exp\left(\frac{r(g_\theta(\mathbf{x}_0'))}{\alpha}\right)p_0(\mathbf{x}_0')d\mathbf{x}_0' \quad \text{(sifting property of Dirac delta)} \\
&= \int_{\mathbb{R}^d} \exp\left(\frac{r(g_\theta(\mathbf{x}_0))}{\alpha}\right)p_0(\mathbf{x}_0)d\mathbf{x}_0. \qquad (21)
\end{aligned}$$

Definition 2 (Tilted Noise Distribution)**.** The *tilted noise density* $p_0^\star(\mathbf{x}_0)$ is defined as:

$$p_0^\star(\mathbf{x}_0) := \frac{1}{Z^\star}p_0(\mathbf{x}_0)\exp\left(\frac{r(g_\theta(\mathbf{x}_0))}{\alpha}\right), \qquad (22)$$

where $Z^\star$ is the normalization constant from Equation (18), which by Equation (21) can be computed in noise space.

Theorem 3 (Properties of the Tilted Noise Distribution)**.** *Let $p_0^\star(\mathbf{x}_0)$ be the tilted noise density defined in Definition 2 and $P_0^\star$ be the corresponding probability measure. Under our standing assumptions:*

1. ***Pushforward Identity:*** *The density of the pushforward measure $(g_\theta)_\sharp P_0^\star$ is $p^\star(\mathbf{x})$.*
2. ***KL Projection:*** *The density $p_0^\star(\mathbf{x}_0)$ uniquely minimizes $D_{\mathrm{KL}}(q_0\|p_0)$ among all noise densities $q_0(\mathbf{x}_0)$ such that the density of $(g_\theta)_\sharp Q_0$ (where Q_0 is the measure for q_0) equals $p^\star(\mathbf{x})$.*

Proof. **Part 1: Pushforward Identity.** We need to show that $(g_\theta)_\sharp P_0^\star$ has density $p^\star(\mathbf{x})$. For any bounded measurable set $A \subset \mathbb{R}^d$, we have:

$$\begin{aligned}
((g_\theta)_\sharp P_0^\star)(A) = P_0^\star(g_\theta^{-1}(A)) &= \int_{g_\theta^{-1}(A)} p_0^\star(\mathbf{x}_0)d\mathbf{x}_0 \\
&= \int_{g_\theta^{-1}(A)} \frac{1}{Z^\star}p_0(\mathbf{x}_0)\exp\left(\frac{r(g_\theta(\mathbf{x}_0))}{\alpha}\right)d\mathbf{x}_0. \qquad (23)
\end{aligned}$$

To evaluate this integral, we use the fundamental property of pushforward measures. For any measurable function $h : \mathbb{R}^d \to \mathbb{R}$:

$$\int_{\mathbb{R}^d} h(\mathbf{x})((g_\theta)_\sharp P_0)(d\mathbf{x}) = \int_{\mathbb{R}^d} h(g_\theta(\mathbf{x}_0))P_0(d\mathbf{x}_0). \qquad (24)$$

Applying this with $h(\mathbf{x}) = \mathbf{1}_A(\mathbf{x})\exp\left(\frac{r(\mathbf{x})}{\alpha}\right)$:

$$\int_{g_\theta^{-1}(A)} \exp\left(\frac{r(g_\theta(\mathbf{x}_0))}{\alpha}\right)p_0(\mathbf{x}_0)d\mathbf{x}_0 = \int_A \exp\left(\frac{r(\mathbf{x})}{\alpha}\right)p^{\text{base}}(\mathbf{x})d\mathbf{x}. \qquad (25)$$

Substituting back into Equation (23):

$$\begin{aligned}((g_\theta)_\sharp P_0^\star)(A) &= \frac{1}{Z^\star}\int_A \exp\left(\frac{r(\mathbf{x})}{\alpha}\right) p^{\text{base}}(\mathbf{x})d\mathbf{x} \\ &= \int_A \frac{1}{Z^\star} p^{\text{base}}(\mathbf{x}) \exp\left(\frac{r(\mathbf{x})}{\alpha}\right) d\mathbf{x} \\ &= \int_A p^\star(\mathbf{x})d\mathbf{x}. \end{aligned} \tag{26}$$

Since this holds for all measurable sets A, the pushforward $(g_\theta)_\sharp P_0^\star$ has density $p^\star(\mathbf{x})$.

Part 2: KL Projection Characterization. Consider the constrained optimization problem:

$$\min_{q_0} D_{\text{KL}}(q_0 \| p_0) \quad \text{subject to} \quad (g_\theta)_\sharp Q_0 \text{ has density } p^\star. \tag{27}$$

We use the method of Lagrange multipliers. Introduce a multiplier function $\lambda : \mathbb{R}^d \to \mathbb{R}$ and consider the functional:

$$\mathcal{L}(q_0, \lambda) = \int_{\mathbb{R}^d} q_0(\mathbf{x}_0) \log \frac{q_0(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} d\mathbf{x}_0 + \int_{\mathbb{R}^d} \lambda(\mathbf{x}) \left(p^\star(\mathbf{x}) - \rho_{q_0}(\mathbf{x})\right) d\mathbf{x}, \tag{28}$$

where $\rho_{q_0}(\mathbf{x})$ is the density of $(g_\theta)_\sharp Q_0$.

For the constraint term, we can write:

$$\int_{\mathbb{R}^d} \lambda(\mathbf{x}) \rho_{q_0}(\mathbf{x}) d\mathbf{x} = \int_{\mathbb{R}^d} \lambda(g_\theta(\mathbf{x}_0)) q_0(\mathbf{x}_0) d\mathbf{x}_0, \tag{29}$$

using the change of variables formula for pushforward measures.

Therefore:

$$\mathcal{L}(q_0, \lambda) = \int_{\mathbb{R}^d} q_0(\mathbf{x}_0) \left(\log \frac{q_0(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} - \lambda(g_\theta(\mathbf{x}_0))\right) d\mathbf{x}_0 + \int_{\mathbb{R}^d} \lambda(\mathbf{x}) p^\star(\mathbf{x}) d\mathbf{x}. \tag{30}$$

Taking the functional derivative with respect to $q_0(\mathbf{x}_0)$ and setting to zero:

$$\frac{\delta \mathcal{L}}{\delta q_0(\mathbf{x}_0)} = \log \frac{q_0(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} + 1 - \lambda(g_\theta(\mathbf{x}_0)) = 0. \tag{31}$$

This yields:

$$q_0(\mathbf{x}_0) = p_0(\mathbf{x}_0) \exp[\lambda(g_\theta(\mathbf{x}_0)) - 1]. \tag{32}$$

To satisfy the constraint, we need the density of $(g_\theta)_\sharp Q_0$ to equal $p^\star(\mathbf{x})$. Using Part 1 in reverse, this happens when:

$$q_0(\mathbf{x}_0) = \frac{1}{Z^\star} p_0(\mathbf{x}_0) \exp\left(\frac{r(g_\theta(\mathbf{x}_0))}{\alpha}\right). \tag{33}$$

Comparing with the optimality condition, we need:

$$\lambda(g_\theta(\mathbf{x}_0)) - 1 = \frac{r(g_\theta(\mathbf{x}_0))}{\alpha} - \log Z^\star. \tag{34}$$

Setting $\lambda(\mathbf{x}) = \frac{r(\mathbf{x})}{\alpha} - \log Z^\star + 1$, we obtain $q_0 = p_0^\star$.

Uniqueness follows from the strict convexity of the KL divergence in its first argument. □

Objective for Learning the Tilted Noise Distribution. To learn a parameterized noise density $p_0^\phi(\mathbf{x}_0)$ that approximates $p_0^\star(\mathbf{x}_0)$, we minimize $D_{\text{KL}}(p_0^\phi \| p_0^\star)$.

Proposition 4 (KL Objective for Learning Tilted Noise Density)**.** *Minimizing $D_{\text{KL}}(p_0^\phi \| p_0^\star)$ with respect to ϕ is equivalent to minimizing:*

$$J_{\textit{noise}}(\phi) = D_{\text{KL}}(p_0^\phi \| p_0) - \frac{1}{\alpha}\mathbb{E}_{\mathbf{x}_0 \sim p_0^\phi}[r(g_\theta(\mathbf{x}_0))]. \tag{35}$$

Proof. Using the definition of $p_0^\star(\mathbf{x}_0)$ from Equation (22):

$$\begin{aligned} D_{\mathrm{KL}}(p_0^\phi \| p_0^\star) &= \int_{\mathbb{R}^d} p_0^\phi(\mathbf{x}_0) \log \frac{p_0^\phi(\mathbf{x}_0)}{p_0^\star(\mathbf{x}_0)} d\mathbf{x}_0 \\ &= \int_{\mathbb{R}^d} p_0^\phi(\mathbf{x}_0) \log \frac{p_0^\phi(\mathbf{x}_0) Z^\star}{p_0(\mathbf{x}_0) \exp\left(\frac{r(g_\theta(\mathbf{x}_0))}{\alpha}\right)} d\mathbf{x}_0 \\ &= \int_{\mathbb{R}^d} p_0^\phi(\mathbf{x}_0) \left(\log \frac{p_0^\phi(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} - \frac{r(g_\theta(\mathbf{x}_0))}{\alpha} + \log Z^\star \right) d\mathbf{x}_0 \\ &= D_{\mathrm{KL}}(p_0^\phi \| p_0) - \frac{1}{\alpha} \mathbb{E}_{\mathbf{x}_0 \sim p_0^\phi}[r(g_\theta(\mathbf{x}_0))] + \log Z^\star. \end{aligned} \tag{36}$$

Since $\log Z^\star$ is constant with respect to ϕ, minimizing $D_{\mathrm{KL}}(p_0^\phi \| p_0^\star)$ is equivalent to minimizing $J_{\text{noise}}(\phi)$. □

A.3.1 Connection to Stochastic Optimal Control

We now show how our result connects to the sophisticated stochastic optimal control framework of Uehara et al. [91] for fine-tuning continuous-time diffusion models, demonstrating that their approach naturally reduces to our simpler result for one-step generators.

Continuous-Time Framework. Uehara et al. [91] consider the entropy-regularized control problem:

$$\max_{u,\nu} \mathbb{E}_{P^{u,\nu}}[r(\mathbf{x}_T)] - \alpha \mathbb{E}_{P^{u,\nu}} \left[\int_0^T \frac{\|u(t,\mathbf{x}_t)\|^2}{2\sigma^2(t)} dt + \log \frac{\nu(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} \right] \tag{37}$$

where $P^{u,\nu}$ is the path measure induced by the SDE with drift $f(t,\mathbf{x}) + u(t,\mathbf{x})$ and ν the initial distribution to optimize.

Reduction to One-Step Generators. For a one-step generator $\mathbf{x} = g_\theta(\mathbf{x}_0)$, the stochastic process degenerates:

- The evolution is deterministic: $\mathbf{x}_T = g_\theta(\mathbf{x}_0)$
- No drift control is needed: optimal $u \equiv 0$
- Only the initial distribution ν requires optimization

The objective reduces to:

$$\max_{\nu} \mathbb{E}_{\mathbf{x}_0 \sim \nu}[r(g_\theta(\mathbf{x}_0))] - \alpha \cdot D_{\mathrm{KL}}(\nu \| p_0) \tag{38}$$

Optimal Initial Distribution. According to their Corollary 2, the optimal initial distribution is:

$$\nu^*(\mathbf{x}_0) = \frac{\exp(v_0^*(\mathbf{x}_0)/\alpha) \cdot p_0(\mathbf{x}_0)}{Z^\star} \tag{39}$$

where $v_0^*(\mathbf{x}_0)$ is the value function at time $t = 0$.

Value Function for Deterministic Generators. From their Lemma 1 (Feynman-Kac formulation), the value function satisfies:

$$\exp\big(v_0^*(\mathbf{x}_0)/\alpha\big) = \mathbb{E}_{P^{0,\nu}} \left[\exp\left(\frac{r(\mathbf{x}_T)}{\alpha} \right) \middle| \mathbf{x}_0 \right] \tag{40}$$

For the deterministic generator g_θ:

$$\begin{aligned} \mathbb{E}[\exp(r(\mathbf{x}_T)/\alpha)|\mathbf{x}_0] &= \mathbb{E}[\exp(r(g_\theta(\mathbf{x}_0))/\alpha)|\mathbf{x}_0] \\ &= \exp(r(g_\theta(\mathbf{x}_0))/\alpha) \quad \text{(deterministic given } \mathbf{x}_0\text{)} \end{aligned} \tag{41}$$

Therefore: $v_0^*(\mathbf{x}_0) = r(g_\theta(\mathbf{x}_0))$.

Final Result and Validation. Substituting back into the optimal distribution formula:

$$\nu^*(\mathbf{x}_0) = \frac{\exp(r(g_\theta(\mathbf{x}_0))/\alpha) \cdot p_0(\mathbf{x}_0)}{Z^*} \tag{42}$$

where $Z^* = \int \exp(r(g_\theta(\mathbf{x}_0))/\alpha) \cdot p_0(\mathbf{x}_0) d\mathbf{x}_0$.

This is precisely our $p_0^\star$ in Definition 2. This alignment between the two frameworks is significant, as it confirms that:

1. Our direct variational approach and the general stochastic control theory yield the same optimal noise distribution.
2. This equivalence arises because for one-step generators, the continuous-time framework naturally collapses to our setting, with their value function $v_0^\star$ simplifying to the composed reward $r \circ g_\theta$.

This connection not only validates our result but also situates it as an important special case within the broader theory of stochastic control.

A.4 Tractable KL Divergence for Noise Modification

We derive a tractable expression for $D_{\mathrm{KL}}(p_0^\phi \| p_0)$ where p_0^ϕ is the density of modified noise $\hat{\mathbf{x}}_0 = T_\phi(\mathbf{x}_0)$ with $T_\phi(\mathbf{x}_0) = \mathbf{x}_0 + f_\phi(\mathbf{x}_0)$. This derivation involves the change of variables formula, simplification of Gaussian log-PDF terms, and an application of Stein's Lemma.

Setup and Minimal Assumptions. Let $T_\phi : \mathbb{R}^d \to \mathbb{R}^d$ be the residual transformation:

$$T_\phi(\mathbf{x}_0) = \mathbf{x}_0 + f_\phi(\mathbf{x}_0) \tag{43}$$

where $f_\phi : \mathbb{R}^d \to \mathbb{R}^d$ is a learned perturbation function with Jacobian $J_{f_\phi}(\mathbf{x}_0) = \frac{\partial f_\phi(\mathbf{x}_0)}{\partial \mathbf{x}_0^T}$.

Assumption 1 (Regularity Conditions)**.** We assume:

1. f_ϕ is continuously differentiable
2. T_ϕ is a global diffeomorphism (invertible with continuous derivatives)
3. f_ϕ satisfies the regularity conditions for Stein's lemma: $\mathbb{E}[\|f_\phi(\mathbf{x}_0)\|^2] < \infty$ and $\mathbb{E}[\|\mathbf{x}_0\| \|f_\phi(\mathbf{x}_0)\|] < \infty$ for $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, I)$

Sufficient Condition for Global Diffeomorphism. While Assumption 1 requires T_ϕ to be a global diffeomorphism, we provide a practical sufficient condition:

Lemma 5 (Lipschitz Condition for Invertibility)**.** *If f_ϕ is L-Lipschitz continuous with $L < 1$, then T_ϕ is a global diffeomorphism.*

Proof. Bi-Lipschitz bounds: for any $\mathbf{x}_0, \mathbf{x}_0'$,

$$\|T_\phi(\mathbf{x}_0) - T_\phi(\mathbf{x}_0')\| \le \|\mathbf{x}_0 - \mathbf{x}_0'\| + \|f_\phi(\mathbf{x}_0) - f_\phi(\mathbf{x}_0')\| \le (1 + L) \|\mathbf{x}_0 - \mathbf{x}_0'\|, \tag{44}$$

$$\|T_\phi(\mathbf{x}_0) - T_\phi(\mathbf{x}_0')\| \ge \|\mathbf{x}_0 - \mathbf{x}_0'\| - \|f_\phi(\mathbf{x}_0) - f_\phi(\mathbf{x}_0')\| \ge (1 - L) \|\mathbf{x}_0 - \mathbf{x}_0'\|. \tag{45}$$

Hence T_ϕ is injective. For surjectivity, fix any target $\mathbf{y}$ and define $G_\mathbf{y}(\mathbf{z}) = \mathbf{y} - f_\phi(\mathbf{z})$, a contraction with constant $L < 1$. By Banach's fixed-point theorem there exists a unique $\mathbf{z}^\star$ with $\mathbf{z}^\star = G_\mathbf{y}(\mathbf{z}^\star)$, i.e., $T_\phi(\mathbf{z}^\star) = \mathbf{y}$. Finally, $J_{T_\phi}(\mathbf{x}_0) = I + J_{f_\phi}(\mathbf{x}_0)$ is invertible for all $\mathbf{x}_0$ (its smallest singular value is at least $1 - L > 0$), and the inverse is C^1 by the inverse function theorem. Thus T_ϕ is a global C^1 diffeomorphism. □

KL Divergence via Change of Variables. Under Assumption 1, we can apply the change of variables formula. The KL divergence is:

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\hat{\mathbf{x}}_0 \sim p_0^\phi} \left[\log \frac{p_0^\phi(\hat{\mathbf{x}}_0)}{p_0(\hat{\mathbf{x}}_0)} \right] \tag{46}$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim p_0} \left[\log \frac{p_0^\phi(T_\phi(\mathbf{x}_0))}{p_0(T_\phi(\mathbf{x}_0))} \right] \tag{47}$$

By the change of variables formula:

$$p_0^\phi(T_\phi(\mathbf{x}_0)) = p_0(\mathbf{x}_0)|\det(J_{T_\phi}(\mathbf{x}_0))|^{-1} \tag{48}$$

Since $J_{T_\phi}(\mathbf{x}_0) = I + J_{f_\phi}(\mathbf{x}_0)$, substituting into Equation (47):

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\mathbf{x}_0 \sim p_0}\left[\log p_0(\mathbf{x}_0) - \log p_0(T_\phi(\mathbf{x}_0)) - \log|\det(I + J_{f_\phi}(\mathbf{x}_0))|\right] \tag{49}$$

Specialization to Gaussian Base Distribution. For $p_0(\mathbf{x}_0) = \mathcal{N}(\mathbf{0}, I)$, the log-density difference simplifies:

$$\log p_0(\mathbf{x}_0) - \log p_0(T_\phi(\mathbf{x}_0)) = -\frac{1}{2}\|\mathbf{x}_0\|^2 + \frac{1}{2}\|T_\phi(\mathbf{x}_0)\|^2 \tag{50}$$

$$= -\frac{1}{2}\|\mathbf{x}_0\|^2 + \frac{1}{2}\|\mathbf{x}_0 + f_\phi(\mathbf{x}_0)\|^2 \tag{51}$$

$$= \mathbf{x}_0^T f_\phi(\mathbf{x}_0) + \frac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 \tag{52}$$

Substituting Equation (52) into Equation (49):

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, I)}\left[\mathbf{x}_0^T f_\phi(\mathbf{x}_0) + \frac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 - \log|\det(I + J_{f_\phi}(\mathbf{x}_0))|\right] \tag{53}$$

Application of Stein's Lemma. Under the regularity conditions in Assumption 1, Stein's lemma applies:

Lemma 6 (Stein's Lemma for Vector Fields)**.** *Let $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, I)$ and $h : \mathbb{R}^d \to \mathbb{R}^d$ satisfy $\mathbb{E}[\|h(\mathbf{x})\|^2] < \infty$ and $\mathbb{E}[\|\mathbf{x}\|\|h(\mathbf{x})\|] < \infty$. Then:*

$$\mathbb{E}[\mathbf{x}^T h(\mathbf{x})] = \mathbb{E}[Tr(J_h(\mathbf{x}))] \tag{54}$$

Applying Lemma 6 to Equation (53), we obtain:

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\mathbf{x}_0 \sim p_0}\left[\frac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 + \mathrm{Tr}(J_{f_\phi}(\mathbf{x}_0)) - \log|\det(I + J_{f_\phi}(\mathbf{x}_0))|\right] \tag{55}$$

This is exactly the expression referenced in the main text.

Log-Determinant Approximation Analysis. Let $\mathcal{E}(A) \coloneqq \mathrm{Tr}(A) - \log|\det(I + A)|$. Then Equation (55) can be rewritten as:

$$D_{\mathrm{KL}}(p_0^\phi \| p_0) = \mathbb{E}_{\mathbf{x}_0 \sim p_0}\left[\frac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2 + \mathcal{E}(J_{f_\phi}(\mathbf{x}_0))\right] \tag{56}$$

To simplify this expression, we analyze the error term $\mathcal{E}(J_{f_\phi}(\mathbf{x}_0))$. The following theorem provides a bound on this term under a Lipschitz assumption on f_ϕ.

Theorem 7 (Bound on Log-Determinant Approximation Error)**.** *Let $A = J_{f_\phi}(\mathbf{x}_0)$ be the $d \times d$ Jacobian matrix of $f_\phi(\mathbf{x}_0)$. Assume f_ϕ is L-Lipschitz continuous, such that its Lipschitz constant $L < 1$. This implies that the spectral radius $\rho(A) \leq L < 1$. Then, the error term $\mathcal{E}(A) = Tr(A) - \log|\det(I + A)|$ is bounded by:*

$$|\mathcal{E}(A)| \leq d(-\log(1 - L) - L) \tag{57}$$

Proof. Since f_ϕ is L-Lipschitz, the spectral norm of its Jacobian satisfies $\|A\|_2 \leq L$. This implies the spectral radius $\rho(A) \leq \|A\|_2 \leq L < 1$, ensuring all eigenvalues $\lambda_i(A)$ satisfy $|\lambda_i(A)| < 1$.

Since $1 + \lambda_i(A) > 0$ for all i, we have $\det(I + A) > 0$, so $\log|\det(I + A)| = \log\det(I + A)$.

For $\rho(A) < 1$, the matrix logarithm series converges:

$$\log\det(I + A) = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k}\mathrm{Tr}(A^k) \tag{58}$$

Therefore:

$$\mathcal{E}(A) = \text{Tr}(A) - \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} \text{Tr}(A^k) \tag{59}$$

$$= \sum_{k=2}^{\infty} \frac{(-1)^k}{k} \text{Tr}(A^k) \tag{60}$$

Taking absolute values and using $|\text{Tr}(A^k)| \leq d \cdot \rho(A)^k \leq d \cdot L^k$:

$$|\mathcal{E}(A)| \leq \sum_{k=2}^{\infty} \frac{d \cdot L^k}{k} \tag{61}$$

$$= d \left(\sum_{k=1}^{\infty} \frac{L^k}{k} - L \right) \tag{62}$$

$$= d(-\log(1-L) - L) \tag{63}$$

□

Practical Approximation and Final Objective. Theorem 7 shows that if the Lipschitz constant L of f_ϕ is sufficiently small (specifically, $L < 1$), the error term $|\mathcal{E}(A)|$ is bounded. For small L, $-\log(1-L) - L \approx L^2/2$, making the bound approximately $dL^2/2$. Thus, the expected error $\mathbb{E}_{\mathbf{x}_0 \sim p_0}[\mathcal{E}(J_{f_\phi}(\mathbf{x}_0))]$ becomes negligible if L is kept small. Under this condition, we can approximate the KL divergence with:

$$D_{\text{KL}}(p_0^\phi \| p_0) \approx \mathbb{E}_{\mathbf{x}_0 \sim p_0} \left[\frac{1}{2} \| f_\phi(\mathbf{x}_0) \|^2 \right] \tag{64}$$

This approximation simplifies the KL divergence term in our objective to a computationally tractable L_2 penalty on the magnitude of the noise modification $f_\phi(\mathbf{x}_0)$.

Integration with Main Objective. Combining our approximation with Proposition 4, and substituting Equation (64) into our initial noise modulation objective, we arrive at the final loss to minimize:

$$\mathcal{L}_{\text{noise}}(\phi) = \mathbb{E}_{\mathbf{x}_0 \sim p_0} \left[\frac{1}{2} \| f_\phi(\mathbf{x}_0) \|^2 - \frac{1}{\alpha} r\big(g_\theta(\mathbf{x}_0 + f_\phi(\mathbf{x}_0))\big) \right] \tag{65}$$

This objective balances reward maximization against the KL regularization term, providing a principled and computationally tractable approach to learning the reward-tilted noise distribution.

Practical Implementation Considerations. The validity of our approximation depends on maintaining small Lipschitz constants. In practice, this is supported by:

1. **Initialization**: Setting $f_\phi(\cdot) \equiv \mathbf{0}$ ensures $\mathcal{E}(A) = 0$ initially
2. **Regularization**: The term $\frac{1}{2}\|f_\phi(\mathbf{x}_0)\|^2$ naturally penalizes large perturbations, helping maintain small eigenvalues of J_{f_ϕ}

While we do not explicitly enforce $L < 1$ during training, these practical measures help maintain f_ϕ in a regime where our approximation remains accurate throughout the optimization process.

B Experimental and Implementation Details

In this Section we report the details for all of our experimental results. We mainly use the SANA-Sprint 0.6B [13] model, and train it using one-step generation. Additionally, we use the default guidance scale of 4.5 for all experiments. After training, we evaluate our models using different amounts of NFEs with one forward pass of Noise Hypernetwork beforehand.

LoRA parameterization We parameterize our noise hypernetwork f_ϕ with LoRA weights on top of the base distilled generative model. We found this to be important mainly to reuse the conditional pathways learned by the base model. This is especially important for complex conditioning, like text. Without this paramertization, which we also explored initially, we found it difficult for the noise hypernetwork to learn an effective conditioning with limit data. While larger-scale training could be a solution to this, we found this LoRA parameterization to be an efficient solution. For a condition independent reward, e.g. the redness one, it is less important to choose such a parametrization.

Initialization As described in Section 3.2, we initialize the noise network to output $f_\phi(\cdot) = \mathbf{0}$) at the start of training. We implement this by setting the output of the last base layer to $\mathbf{0}$ and initializing the LoRA weights of the second LoRA weight matrix (also reffered to as B) to 0. This effectively initializes $f_\phi(\cdot) = \mathbf{0}$). For a stable training, this initialization is important as the model $g_\theta(f_\phi(\mathbf{x}_0) + \mathbf{x}_0))$ generates meaningful images at the start of training. In that way f_ϕ only needs to learn how to refine $\mathbf{x}_0$.

Memory efficient implementation. Section 3.2, we train our noise hypernetwork f_ϕ as a special LoRA version of our base model g_θ, which ignores the last layer of the base model. As visualized in Figure 2, we only need to keep the base model in memory once. Thus, the GPU memory overhead is just the added LoRA weights ϕ. Additionally, we employ Pytorch Memsave [7] to all models, which further reduces the needed GPU memory during training enabling us to use larger batch sizes. We run all experiments in `bfloat16`. Additionally, we can leverage gradient checkpointing on the first call of the model with activated LoRA parameters to further reduce memory. We use this for our FLUX-Schnell training.

B.1 Redness Reward

For the Redness Reward, we use SANA-Sprint 0.6B [13] as the base model. We train the model with the redness reward

$$r(\mathbf{x}) = \tfrac{1}{100} * (\mathbf{x}^0 - \frac{1}{2}(\mathbf{x}^1 + \mathbf{x}^2)),$$

where $\mathbf{x}^i$ denotes the i-th color channel of $\mathbf{x}$. We use the same amount of LoRA parameters for fine-tuning and noise hypernetwork training. In general, we keep the hyperparameters for our comparison between fine-tuning and noise hypernetwork training exactly the same. Due to the sake of illustration, we lower the learning rate for fine-tuning in this case as otherwise the model collapses to generating pure red images after a few training steps. We train on 30 prompts from the GenEval [26] promptset and evaluate on the four unseen prompts `["A photo of a parrot", "A photo of a dolphin", "A photo of a train", "A photo of a car"]`. After each epoch on the 30 prompts, we compute the redness reward as well as an "imageness score" for each of the 4 evaluation prompts and average. For the imageness score, we use the ImageReward [103] human-preference reward model as it was shown to correctly quantify prompt-following capabilities. We provide the full hyperparameters in Table 3. This experiment was conducted on 1 H100 GPU.

Table 3: Hyperparameters for the Redness Reward setting

	Fine-tuning	Noise Hypernetwork
Model	SANA-Sprint [13]	SANA-Sprint [13]
Learning rate	$1e-4$	$1e-3$
GradNorm Clipping	1.0	1.0
LoRA rank	128	128
LoRA alpha	256	256
Optimizer	SGD	SGD
Batch size	3	3
Training epochs	200	200
Number of training prompts	30	30
Image size	1024×1024	1024×1024

B.2 Human Preference Reward Models

For our large-scale experiments, we consider SD-Turbo [83] and SANA-Sprint [13] as our two base models. For SD-Turbo we generate images in 512×512 while for SANA-Sprint we generate them of size 1024×1024. The training for the noise hypernetwork is done using ~70k prompts from Pick-a-Picv2 [48], T2I-Compbench train set [37], and Attribute Binding (ABC-6K) [25] prompts. As the reward we follow ReNO [22] and use a combination of human-preference trained reward models consisting of ImageReward [103], HPSv2.1 [101], PickScore [48], and CLIP-Score [38]. To balance these, we weigh each reward model with the same weightings as proposed in ReNO [22] and employ them with the following implementation details. All training runs were conducted on 6 H100 GPUs.

Human Preference Score v2.1 (HPSv2.1) HPSv2.1 [101] is an improved version of the HPS [102] model, which uses an OpenCLIP ViT-H/14 model and is trained on prompts collected from DiffusionDB [100] and other sources.

PickScore PickScore also uses the same ViT-H/14 model, however is trained on the Pick-a-Pic dataset which consists of 500k+ preferences that are collected through crowd-sourced prompts and comparisons.

ImageReward ImageReward [103] trains a MLP over the features extracted from a BLIP model [51]. This is trained on a dataset of images collected from the DiffusionDB [100] prompts.

CLIPScore Lastly, we use CLIPScore [32, 76], which was not designed specifically as a human preference reward model. However, it measures the text-image alignment with a score between 0 and 1. Thus, it offers a way of evaluating the prompt faithfulness of the generated image that can be optimized. We use the model provided by OpenCLIP [38] with a ViT-H/14 backbone.

Table 4: Hyperparameters for the Human-preference Reward setting

	Noise Hypernetwork	Fine-tuning	Noise Hypernetwork	Noise Hypernetwork
Model	SD-Turbo [83]	SANA-Sprint [13]	SANA-Sprint [13]	FLUX-Schnell
Learning rate	$1e-3$	$1e-3$	$1e-3$	$1e-3$
GradNorm Clipping	10.0	1.0	1.0	10.0
LoRA rank	128	128	128	128
LoRA alpha	128	256	256	$5*128$
Optimizer	SGD	SGD	SGD	SGD
Batch size	18	18	18	7
Accumulation Steps	1	3	3	4
Training Epochs	≈ 25	≈ 25	≈ 25	≈ 25
Number of training prompts	$\approx 70k$	$\approx 70k$	$\approx 70k$	$\approx 70k$
Image size	512×512	1024×1024	1024×1024	512×512

GenEval Our main evaluation metric is GenEval, an object-focused framework introduced by Ghosh et al. [26] for evaluating the alignment between text prompts and generated images from Text-to-Image (T2I) models. GenEval leverages existing object detection methods to perform a fine-grained, instance-level analysis of compositional capabilities. The framework assesses various aspects of image generation, including object co-occurrence, position, count, and color. By linking the object detection pipeline with other discriminative vision models, GenEval can further verify properties like object color. All the metrics on the GenEval benchmarks are evaluated using a MaskFormer object detection model with a Swin Transformer [57] backbone. Lastly, GenEval is evaluated over four seeds and reports the mean for each metric, which we follow. Note that our FLUX-Schnell differ from the ones in Eyring et al. [22] as we use `bfloat16` instead of `float16`.

B.3 Test-time techniques

For ReNO [22], we use the default parameters as described in their paper with 50 forward passes for one image generation. For Best-of-N [44] we use $N = 50$ with the same reward ensemble for a fair comparison. For LLM-based prompt optimization [4, 61], we use the default setup from the MILS [4] repository (https://github.com/facebookresearch/MILS/blob/main/main_

image_generation_enhancement.py) with local Llama 3.1 8B Instruct as the LLM. The time reflected in Table 1 reflects these local LLM calls. Note that we left the GPU memory to just the base image generation model. We modify the hyperparameters to 5 prompt proposals for each LLM call and 10 iterations, such that we also end up with 50 image evaluations for a fair comparison.

C Additional results

In this section we report additional quantiative ablation results and further qualitative results.

C.1 Additional Benchmarks

Here, we report further results on two more benchmarks commonly employed in the evaluation of T2I generation. Note that again, none of the prompts in the used benchmarks are part of the training data, showcasing the generalizability of the Noise Hypernetwork to unseen prompts and also that our optimization objective through human-preference reward mdoels is disentangled from these benchmarks.

T2I-CompBench. T2I-CompBench is a comprehensive benchmark proposed by Park et al. [71] for evaluating the compositional capabilities of text-to-image generation models. We evaluate on the Attribute binding tasks, which includes color, shape, and texture sub-categories, where the model should bind the attributes with the correct objects to generate the complex scene. The attribute binding subtasks are evaluated using BLIP-VQA (i.e., generating questions based on the prompt and applying VQA on the generated image). We perform these evaluations on the validation set of prompts and results are shown in Tab. 5 and observe consistent improvements across steps and categories.

Table 5: **Quantitative Results on T2I-CompBench**. The Noise Hypernetwork consistently improves performance.

SANA-Sprint 0.6B [13]	**NFEs**	**Color** ↑	**Shape**↑	**Texture**↑
One-step	1	0.72	0.49	0.63
+ Noise Hypernetwork	2	**0.75**	**0.53**	**0.64**
Two-step	2	0.73	0.50	0.64
+ Noise Hypernetwork	3	**0.76**	**0.53**	0.64
Four-step	4	0.73	0.50	0.64
+ Noise Hypernetwork	5	**0.76**	**0.54**	**0.65**

DPG-Bench. We provide results on DPG-Bench [36] in Tab. 6. Broadly, while performance increases for all models with increasing timesteps, we note that the results for the four step SANA-Sprint model is nearly matched by the one-step model with our noise hypernetwork. We also note that the DPG-Bench score of 80.82 surpasses powerful models such as SDXL [73], Pixart-Σ, and is only surpassed by much larger models such as SD3 [21], and Flux. Finally, we also note that the human-preference reward models that we utilize all have a CLIP/BLIP encoder that limits the length of the captions to < 77 tokens, which offers minimal scope of improvements for benchmarks involving much longer prompts that exceed this context window. Future reward models that either utilize different CLIP models (e.g. Long-CLIP [104]) or LLM-based decoders (e.g. VQAScore [54]) would enable improving prompt following of these models more dramatically in the case of long prompts.

Table 6: DPG-Bench results for SANA-Sprint highlighting generalization across inference timesteps of our Noise Hypernetwork.

SANA-Sprint 0.6B [13]	**NFEs**	**DPG-Bench Score**↑
One-step	1	77.59
+ Noise Hypernetwork	2	**79.20**
Two-step	2	79.07
+ Noise Hypernetwork	3	**79.74**
Four-step	4	79.54
+ Noise Hypernetwork	5	**80.82**

Figure 6: Examples of artifacts introduced by directly Direct Fine-tuning diffusion models on rewards [14, 53, 74] for the same reward objective in comparison to Noise Hypernetwork training with same initial noise.

C.2 Diversity Analysis

We also investigate the impact of the diversity of the generated outputs as the result of our hypernetwork. For this purpose, we generate 50 images by varying the seed from the 553 prompts of the GenEval benchmark. The average similarity of different images for the same prompt are measured using similarities from

Table 7: We measure the average LPIPS and DINO similarity scores over images generated for 50 different seeds for the 553 prompts from GenEval.

	LPIPS ↑	DINO ↓
SANA-Sprint	0.608 ±0.074	0.780 ±0.103
+ Noise HyperNetwork	0.592 ±0.059	0.825 ±0.090

LPIPS [106] and DINOv2 [69] embeddings. The results in Tab. 7 indicate that the noise hypernetwork does not cause any collapse due to "reward-hacking" and broadly, the diversity of the generated images is in the same ballpark as the base model.

C.3 Multi-step analysis

Here, in addition to the main text Table 2, we analyze the behavior of Noise Hypernetworks when moving beyond the few-step regime of $1-4$ steps. Remarkably, even when going up to 32 inference steps, we find that Noise Hypernetworks trained with the one-step generator, improve performance. We find that as we increase the NFEs, the added performance boost of the Noise Hypernetwork reduces. However, note that the underlying model SANA-sprint [13] was not trained to be used in the multi-step regime, but specifically for few-step generation.

Table 8: Mean GenEval results for SANA-Sprint highlighting generalization across inference timesteps of our Noise Hypernetwork.

SANA-Sprint [13]	**NFEs**	**GenEval Mean↑**
One-step	1	0.70
+ Direct fine-tune [74]	1	0.67
+ Noise Hypernetwork	2	**0.75**
Two-step	2	0.72
+ Direct fine-tune [74]	2	0.66
+ Noise Hypernetwork	3	**0.76**
Four-step	4	0.73
+ Direct fine-tune [74]	4	0.62
+ Noise Hypernetwork	5	**0.77**
Eight-step	8	0.74
+ Noise Hypernetwork	9	**0.76**
Sixteen-step	16	0.73
+ Noise Hypernetwork	17	**0.75**
Thirty-two-step	32	0.71
+ Noise Hypernetwork	33	**0.72**

C.4 Challenges with Direct Fine-tuning

We also qualitatively illustrate the problems with directly fine-tuning diffusion models on differentiable rewards in Figure 6. As visualized, there are drastic artifacts introduced on the image which significantly contribute to improving the reward scores. These artifacts are very similar to the ones noticed in several works [14, 41, 53] and require the development of several regularization strategies to address these issues. However as explained in Section 2, in the few-step regime the KL regularization term to the base model is difficult to be made tractable and thus, to the best of our knowledge there exists no theoretical grounded approach to learn the reward tilted distribution (Equation 3) with a one-step generator. The Noise Hypernework strategy on the other hand, ensures that the images remain in the original data distribution with its principled regularization.

C.5 LoRA Rank analysis

Here, we ablate the LoRA rank for both HyperNoise and direct fine-tuning on SANA-Sprint. We find that a rank of 64 also seems to be sufficient to achieve almost the same improvements as rank 128, while a lower rank seems not to be expressive enough. On the other hand, fine-tuning seems to be suffering from increased overfitting on the reward.

C.6 Qualitative Results

We provide additional qualitative samples for the base SANA-Sprint result along with the generation with our proposed noise hypernetwork in Figures 6 and 8. We broadly observe improved prompt following as well as superior visual quality in the generated images.

Table 9: GenEval results for HyperNoise on SANA-Sprint, showing generalization across timesteps.

Method	**NFEs**	**GenEval Mean↑**
SANA-Sprint (One-step)	1	0.70
LoRA-Rank 128 + HyperNoise	2	0.75
LoRA-Rank 64 + HyperNoise	2	0.75
LoRA-Rank 16 + HyperNoise	2	0.71
LoRA-Rank 8 + HyperNoise	2	0.70
SANA-Sprint (Two-step)	2	0.72
HyperNoise	3	0.76
SANA-Sprint (Four-step)	4	0.73
HyperNoise	5	0.77
HyperNoise (LoRA-Rank=64)	5	0.76

Table 10: GenEval results for direct LoRA fine-tuning on SANA-Sprint.

Method	**NFEs**	**GenEval Mean↑**
SANA-Sprint (One-step)	1	0.70
LoRA-Rank 128 + LoRA fine-tune	1	0.67
LoRA-Rank 64 + LoRA fine-tune	1	0.68
LoRA-Rank 16 + LoRA fine-tune	1	0.65
LoRA-Rank 8 + LoRA fine-tune	1	0.59
SANA-Sprint (Two-step)	2	0.72
LoRA fine-tune	2	0.66
SANA-Sprint (Four-step)	4	0.73
LoRA fine-tune	4	0.62

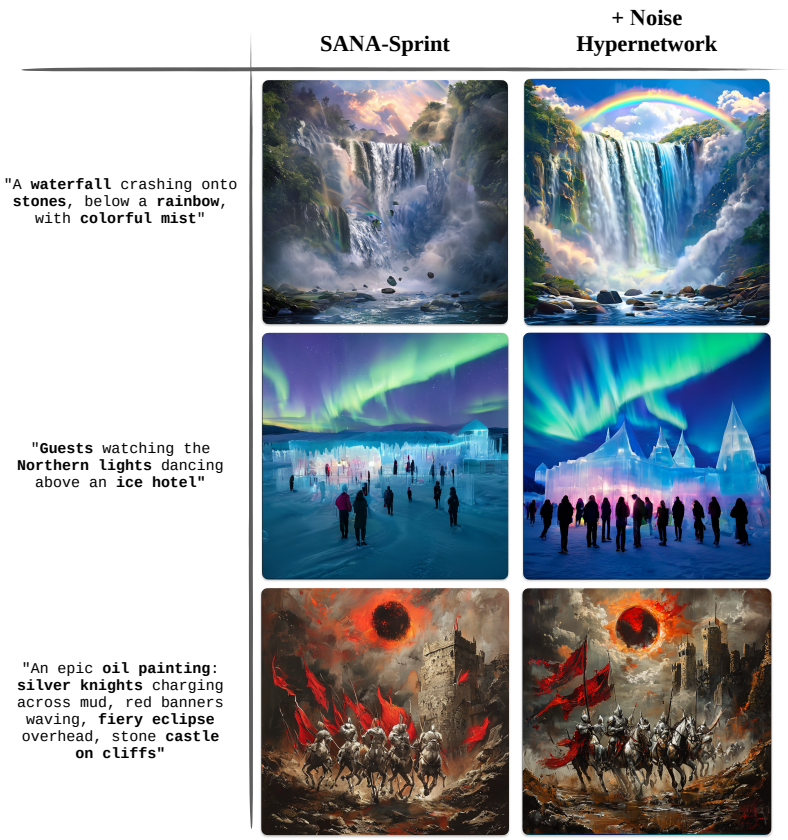

Figure 7: More qualitative results on the human-preference reward setting. Base SANA-Sprint compared to HyperNoise with same initial noise.

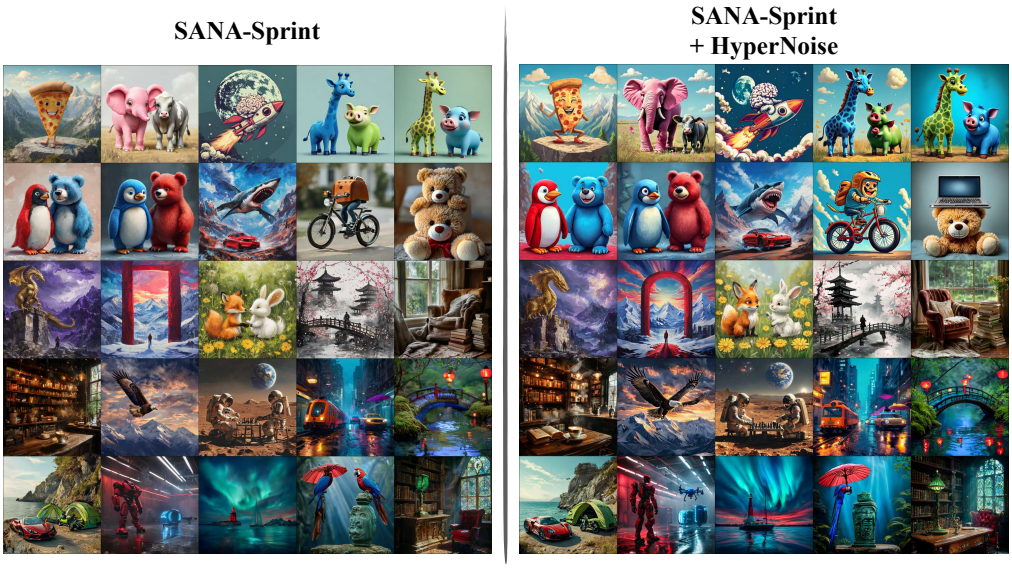

Figure 8: Non-cherry picked results on the human-preference reward setting. Base SANA-Sprint compared to HyperNoise with same initial noise.