Selective Underfitting in Diffusion Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Diffusion models have emerged as the principal paradigm for generative modeling across various domains. During training, they learn the score function, which in turn is used to generate samples at inference. They raise a basic yet unsolved question: *which* score do they actually learn? In principle, a diffusion model that matches the empirical score in the entire data space would simply reproduce the training data, failing to generate novel samples. Recent work addresses this paradox by arguing that diffusion models *underfit* the empirical score due to training-time inductive biases. In this paper, we show that this perspective is incomplete. Instead of underfitting the score everywhere, we show that better diffusion models more accurately approximate the score in certain regions of input space, while underfitting it in others. We characterize these regions and design empirical interventions to validate our perspective. Our results establish that this viewpoint, named *selective underfitting*, is essential for understanding diffusion models, yielding new, testable insights into their generalization and generative performance.

1 Introduction

Diffusion models [37, 15, 38] have recently shown significant success in generating high-quality samples, achieving state-of-the-art performance in tasks like image, video, and audio generation. During *training*, they learn a *score function*–the gradient of the log density of the data distribution convolved with noise–by reconstructing data that has been corrupted with Gaussian noise, a process called *denoising score matching* [43]. During *inference*, the learned score is applied iteratively to denoise Gaussian noise to clean samples.

Despite these successes, the learned score presents an apparent paradox: if training were perfect, the model would learn the *empirical score function*–the score with respect to the finite training data–and simply replicate the training data at inference, never generating novel samples. Recent work has sought to resolve this paradox by proposing various mechanisms by which diffusion models *underfit* the empirical score, due to inductive bias or smoothness of neural networks [19, 30, 35, 33, 54, 53].

In this work, we argue that understanding diffusion models requires us to understand not just *how* they underfit the empirical score, but *where*. Our starting point is the mismatch between the distribution of training samples versus the distribution of inputs encountered during inference trajectories. Consider Figure 1: during training, for most timesteps, inputs concentrate in thin shells around each data point (blue shells). Within these shells, the learned score simply points back to the shells' centers, i.e. the training images (green arrows). In constrast, we find that during inference, denoising trajectories quickly land in regions *beyond* these shells (red points) *that were effectively never supervised with the empirical score at training time*.

This observation prompts us to distinguish two distinct regions in the data space: the **supervision region**, where the model is supervised to approximate the empirical score during training, and the **extrapolation region**, where the model receives no supervision but is queried at inference. Our scaling experiments show that as models improve, their approximation of the empirical score *inside* the supervision region improves, while it worsens *outside* (underfitting). In other words, underfitting

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

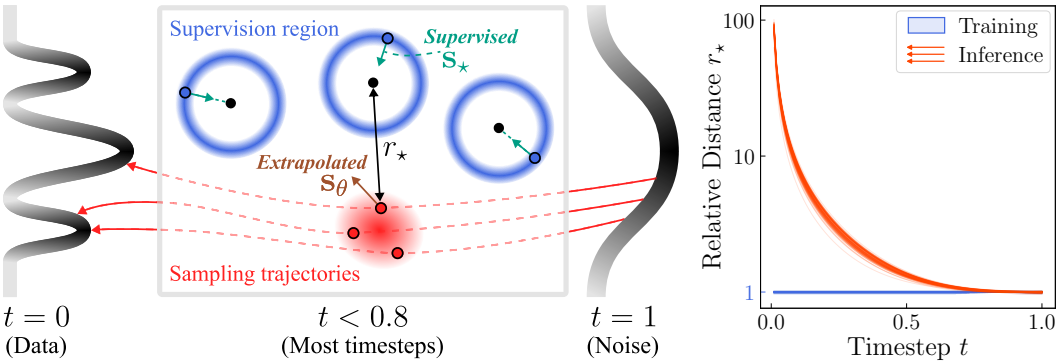


Figure 1: **Selective underfitting.** (a) Training distribution concentrates in a small region of data space, where the model learns the empirical score function $\mathbf{s}_\star$. At inference, sampling trajectories go beyond this region, where predictions are not directly ***supervised*** and must be ***extrapolated***. (b) The plotted distances reflect how far sampling trajectories are from the supervision region.

occurs only in the extrapolation region–a property we term **selective underfitting**, which, as we demonstrate, **is essential for understanding diffusion models.**

In this work, we leverage this lens to design a suite of controlled experiments that shed new light on diffusion models' **(1) generalization**, showing that the *size of the supervision region* strongly affects a model's ability to generalize–enlarging this region degrades generalization, an effect overlooked in prior studies–and **(2) generative performance**, where our novel scaling law between generative performance and *supervision loss* enables directly quantifying the model's ability to *transfer* the supervised score to the extrapolation region. This framework not only clarifies the benefits of recent advances such as REPA [55] and diffusion transformers [32, 28], but also suggests a general principle for designing better training recipes. Taken together, our findings establish *selective underfitting* as a central principle for understanding diffusion models.

Organization. Section 2 reviews diffusion models. Section 3 introduces our perspective of *selective underfitting*, distinguishing between supervision and extrapolation regions in diffusion model learning. Sections 4 and 5 show how this perspective explains the generalization behavior and generative performance of diffusion models, respectively.

2 Background: What and where do diffusion models learn?

In this section, we review the formulation of diffusion models along with commonly held interpretation, which we refer to as *global underfitting*. This sets the stage for our contrasting viewpoint–*selective underfitting*–introduced in the following sections.

Diffusion models[1] define a stochastic forward process that transforms a data distribution into a Gaussian distribution over timesteps $t \in [0, 1]$: $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$ where $\mathbf{x}$ is a clean data and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for schedules $\alpha_t, \sigma_t \colon [0, 1] \to \mathbb{R}$. The models are trained through denoising score matching (DSM) [43] to learn the score function–the gradient of the log density–which is then used at inference to iteratively denoise Gaussian noise into a sample from the data distribution.

Learning Problem. Given training data $\mathcal{D} = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N}$, the DSM objective at timestep t is:

$$\mathcal{L}_{\text{DSM}}(t) := \mathbb{E}_{\mathbf{x}^{(i)} \sim \text{Unif}(\mathcal{D}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \boldsymbol{\epsilon}/\sigma_t \right\|^2 \tag{1}$$

The DSM nature of diffusion model training can be decomposed into two components: (i) *what function* does the model aim to learn? (i.e., what is the minimizer of $\mathcal{L}_{\text{DSM}}$?), and (ii) *where in the data space* does the model learn during training? (i.e., what is the distribution of $\mathbf{z}_t$?). For (i), there exists an analytic optimal solution for Equation (1):

$$\mathbf{s}_\star(\mathbf{z}_t, t) = \frac{1}{\sigma_t^2} \left[-\mathbf{z}_t + \alpha_t \sum_{i=1}^{N} \text{softmax}\left(-\frac{\|\mathbf{z}_t - \alpha_t \mathbf{x}^{(i)}\|^2}{2\sigma_t^2} \right) \mathbf{x}^{(i)} \right]. \tag{2}$$

[1]Throughout this work, we use "diffusion" as an umbrella term for both diffusion and flow matching [25, 26], as they are equivalent [10]. All discussions apply to both.

Since this is the score function of the empirical training data, we refer to $\mathbf{s}_\star$ as the *empirical score function*, often also called the analytic score function. For (ii), a model is trained on noisy versions of the training data, which can be written as a mixture of Gaussians: $\hat{p}_t(\mathbf{z}_t) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{x}^{(i)}, \sigma_t^2 \boldsymbol{I})$. Given (i) and (ii), Equation (1) is equivalent to:

$$\mathcal{L}_{\text{DSM}}(t) = \mathbb{E}_{\mathbf{z}_t \sim \hat{p}_t} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \mathbf{s}_\star(\mathbf{z}_t, t) \right\|^2 + (\text{constant}). \tag{3}$$

In principle, $\hat{p}_t$ *has support over the entire data space* for any $t > 0$, since it is Gaussian. This leads to the following common understanding of the learning problem:

Perspective A (Common)**.** *Diffusion models are supervised to (i) approximate the empirical score function $\mathbf{s}_\star$, (ii) over the entire data space.*

Perspective A, which we call **global underfitting**, has been widely adopted to study diffusion models, for example, in the context of their generalization [19, 18, 35, 53, 46] and memorization [30]. These works primarily focus on (i), analyzing *how* a neural network globally underfits the empirical score function. In contrast, we argue that (ii)–*where* learning occurs–is the critical component, as underfitting occurs *selectively* on certain regions of the data space.

3 Rethinking How Diffusion Models Learn

In this section, we challenge the standard Perspective A from Section 2 that diffusion models learn globally over the entire data space. Instead, we decompose the diffusion model learning into two regions: they are 'supervised' on a restricted region, and forced to 'extrapolate' beyond this region during inference. To motivate this, we begin with a paradox arising from Classifier-Free Guidance (CFG, Ho and Salimans, 2022), which Perspective A cannot explain.

Example 3.1 (CFG Paradox)**.** CFG jointly trains conditional & unconditional models $\mathbf{s}_{\boldsymbol{\theta}}(\cdot, \cdot, c)$ & $\mathbf{s}_{\boldsymbol{\theta}}(\cdot, \cdot, \varnothing)$, and samples with the combined CFG score to improve sample quality:

$$\mathbf{s}^{\omega}_{\text{CFG}} := \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, c) + (\omega - 1)[\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, c) - \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, \varnothing)]$$

CFG works because the *difference between conditional and unconditional scores* is recognizable; if these scores are identical, CFG reduces to standard conditional sampling. Indeed, these two learned scores *differ meaningfully during inference* (Figure 2, red line). According to Perspective A, both learned scores should globally approximate their empirical counterparts on conditional and unconditional training data: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, c) \approx \mathbf{s}^c_\star(\mathbf{z}_t, t)$ and $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, \varnothing) \approx \mathbf{s}^{\varnothing}_\star(\mathbf{z}_t, t)$. Measuring the difference between the empirical scores, however, $\|\mathbf{s}^c_\star - \mathbf{s}^{\varnothing}_\star\|$ over $\mathbf{z}_t \sim \hat{p}_t$, we find it is *vanishingly small during training* for most timesteps (Figure 2, blue line). This leads to a paradox: why is there such a discrepancy?

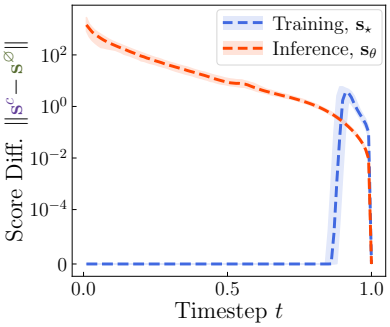


Figure 2: **Difference between conditional and unconditional scores,** measured separately during training and inference. The gap is significant at inference but nearly zero during training.

The paradox prompts us to revisit Perspective A, revealing that the empirical score function during training does not directly translate to the learned score function used at inference. To address this, we now present an overview of our alternative perspective, which naturally resolves this paradox.

Overview. Our perspective, illustrated in Figure 1a, is built on the insight that the score function used at inference is not simply a direct neural network approximation of the empirical score function everywhere (Perspective A), but it is determined by (1) the specific region of the data space where the model is supervised (blue shells), which in turn shapes (2) how the model extrapolates outside this region (red lines). We call this **selective underfitting**, as underfitting turns out to occur only in the latter region, as we demonstrate in Experiment 4.2.

Perspective B (Ours)**.** *Diffusion models are supervised to (i) approximate the empirical score function $\mathbf{s}_\star$, (ii) over a restricted region, and extrapolate beyond this region during inference.*

Resolving the CFG Paradox. During training, the supervised scores for the conditional and unconditional models are nearly identical. The key difference lies in the regions where supervision is applied: the conditional model is supervised on regions formed by *class-wise* data, while the

unconditional model is supervised on regions formed by *all* data. Adapting Perspective B, these mismatched supervision regions induce distinct extrapolation (Figure 2), which explains the observed divergence between the two scores during inference.

In the following sections, we formally describe our Perspective B: Section 3.1 provides a theoretical basis for the restricted nature of the supervision region, and Section 3.2 empirically demonstrates extrapolation beyond this region during inference.

3.1 Supervision Is Restricted During Training

In this section, we theoretically justify the following central claim:

C1. *Diffusion models are effectively supervised only within a restricted region of data space.*

As reviewed in Section 2, the distribution employed at training is the mixture of Gaussians $\hat{p}_t = \frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(\mathbf{z}_t; \alpha_t\mathbf{x}^{(i)}, \sigma_t^2\boldsymbol{I})$. In high-dimensional space, these samples concentrate on a set of thin spherical shells, since for a standard Gaussian vector $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, we have $\|\boldsymbol{\epsilon}\| = \sqrt{d}(1 + \mathcal{O}(1))$ with high probability. Consequently, each component of $\hat{p}_t$ concentrates almost all of its mass on a shell of radius $\sigma_t\sqrt{d}$ centered at $\alpha_t\mathbf{x}^{(i)}$, leading to the proposition below (proof in Appendix B.1):

Proposition 3.2 (Supervision region)**.** *Let $\delta \in (0, 1)$, empirical data $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$, and $\mathbf{z}_t \sim \hat{p}_t$. Then*

$$\mathbb{P}(\mathbf{z}_t \in \mathcal{T}_t(\delta)) \geq 1 - \delta, \quad \mathcal{T}_t(\delta) \colon = \left\{\mathbf{z} : \exists\, i \in [N], \left|\|\mathbf{z} - \alpha_t\mathbf{x}^{(i)}\| - \sigma_t\sqrt{d}\right| \leq \sigma_t\sqrt{d\log(1/\delta)}\right\}.$$

We call $\mathcal{T}_t(\delta)$ the *supervision region*: the union of thin spherical shells around the data points that, taken together, contain $\mathbf{z}_t \sim \hat{p}_t$ with high probability. In practice, these shells are non-overlapping for most time steps because each shell's radius $\sigma_t\sqrt{d}$ is much smaller than the separation between their centers, i.e., $\sigma_t\sqrt{d} \ll \min_{i,j}\left\|\alpha_t\mathbf{x}^{(i)} - \alpha_t\mathbf{x}^{(j)}\right\|$. We quantify this separation using the *Bhattacharyya coefficient* ([2], Appendix B.2)—a standard measure of distributional overlap—computed on the ImageNet dataset [6] and in the latent space of the SD-VAE [34]. Figure 3 (right) shows this overlap coefficient across time; it is essentially zero for $t \in [0, 0.8]$, indicating negligible overlap in this regime.

Empirical Score Collapses. The concentration and separation results reveal more about the empirical score. With high probability, $\mathbf{z}_t \sim \hat{p}_t$ lies in some i-th shell and is far from all others, making the softmax weights in Equation (2) extremely imbalanced: nearly 1 on $\mathbf{x}^{(i)}$ and nearly 0 on all others. As a result, the empirical score reduces to $s_\star(\mathbf{z}_t, t) \approx \left[-z_t + \alpha_t\mathbf{x}^{(i)}\right]/\sigma_t^2$, i.e., a vector pointing to the nearest training data (Figure 1, green arrow). Therefore, despite the elaborate training objective, in fact, the model receives a *trivial* supervision signal inside $\mathcal{T}_t(\delta)$ for most timesteps. We revisit this observation in Section 4 to examine its impact on model generalization.

3.2 Extrapolation Dominates During Inference

In Section 3.1, we characterized how DSM training restricts the supervision region. We now turn to inference and empirically justify the claim below:

C2. *During inference, diffusion models mostly extrapolate (leave the training region early).*

To examine this, we examine sampling trajectories during inference and show that the sampling trajectory deviates from these shells.

Experiment 3.3 (Extrapolation During Inference)**.** Recall from Proposition 3.1 that $\mathbf{z}_t \sim \hat{p}_t$ lies in some i-th shell, characterized by $\left\|\mathbf{z}_t - \alpha_t\mathbf{x}^{(i)}\right\| \approx \sigma_t\sqrt{d}$, with high probability. To measure how much a given $\mathbf{z}_t$ *deviates* from the δ-training region, we define the following quantity:

$$r^{(i)} := \frac{\left\|\mathbf{z}_t - \alpha_t\mathbf{x}^{(i)}\right\|}{\sigma_t\sqrt{d}}, \quad r_\star := r^{(i_\star)} \text{ where } i_\star = \operatorname*{arg\,min}_{i\in\{1,\ldots,N\}}\left|r^{(i)} - 1\right|. \tag{4}$$

If $r_\star \approx 1$, $\mathbf{z}_t$ lies in the supervision region; otherwise, $\mathbf{z}_t$ deviates from the region. We conduct this experiment on the ImageNet dataset using a pretrained SiT-XL model [28]. Figure 1b visualizes the distribution of $r_\star$ during both training and inference. As discussed in Section 3.1, $r_\star$ remains tightly concentrated around 1 during training (blue line). In contrast, $r_\star$ rapidly increases above 1 during inference (red line), indicating that the model escapes the supervision region very early at inference ($t = 1 \to 0$). This demonstrates **C2**: *the model indeed extrapolates during inference, except possibly at the earliest steps.*

4 Generalization through the Lens of Selective Underfitting

In this section, we show how Perspective B demystifies the *generalization* of diffusion models.

Overview. Diffusion model generalization refers to a model's ability to generate unseen (creative) data samples. In the extreme case where a model perfectly learns the empirical score function, it would simply regenerate the training set. Yet the practical success of diffusion models implies they learn different score functions than the empirical one–a phenomenon that remains unexplained, since the DSM loss in Equation (1) is minimized by the empirical score. As this setup departs from standard generalization theory (see Appendix A.2), it has attracted substantial study; however, as we discuss in Section 4.2, prior work overlooks the crucial role of extrapolation. To motivate our perspective, in Section 4.1 we first examine how supervision and extrapolation distinctly shape the learned score function. In our ImageNet experiments, the empirical score $s_\star$ is directly computed using Equation (1).

4.1 Extrapolation as a Key to Generalization

In this section, we show that the learned score exhibits strikingly different behavior within the supervision and extrapolation regions.

Experiment 4.1 (Supervision Encourages Memorization)**.** To analyze the model's learned score in the supervision region, we consider a noisy image $\mathbf{z}_t = \alpha_t \mathbf{x}^{(i)} + \sigma_t \boldsymbol{\epsilon} \sim \hat{p}_t$ (with $\mathbf{x}^{(i)} \in \mathcal{D}$). We then use a pretrained SiT-XL model to iteratively denoise from timestep t to 0. Figure 3 shows that for $t < 0.8$, sampling consistently regresses to the original training image $\mathbf{x}^{(i)}$. This indicates that within this range, the model has (almost perfectly) learned the empirical score, thereby replicating the training image. In other words, the model essentially *memorizes* the training data for $t < 0.8$, in the region where supervision is provided.

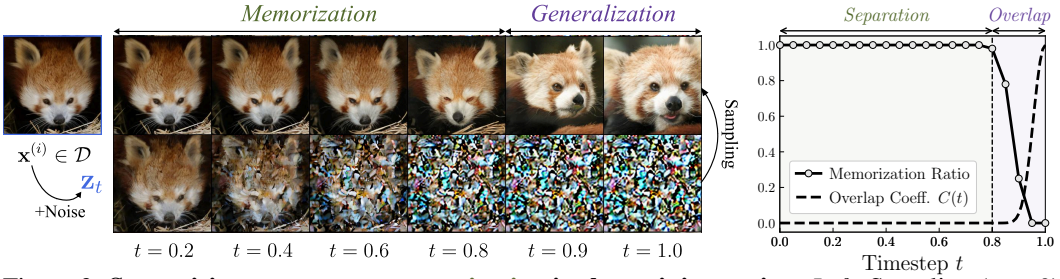


Figure 3: **Supervision encourages memorization in the training region.** *Left*: Sampling ($t \to 0$) from a noisy image $\mathbf{z}_t$ in the supervision region maps back to the original training image $\mathbf{x}^{(i)}$ for most timesteps ($t < 0.8$), demonstrating strong *memorization* due to supervision. *Right*: The memorization ratio exhibits the same trend quantitatively. In addition, *memorization* is linked to the *separation* property of the training region, as shown by the measured overlap coefficient $C(t)$.

Experiment 4.2 (Supervision vs. Extrapolation)**.** We measure the difference of the learned and empirical scores, $\|\mathbf{s}_{\boldsymbol{\theta}} - \mathbf{s}_\star\|^2$, both within the supervision region and along inference-time sampling trajectories. To investigate the effect of scaling, we evaluate a series of pretrained models: SiT-{S, B, L, XL}. Figure 4 reveals a contrast in scaling behavior: As model capacity increases (left to right), $\mathbf{s}_\theta$ approaches $\mathbf{s}_\star$ in the supervision region (blue line). Conversely, in the extrapolation region (during inference), larger models increasingly deviate from $\mathbf{s}_\star$ (red line). This clarifies the reason why distinguishing between the effects of supervision and extrapolation is necessary: diffusion models behave differently in two regions.

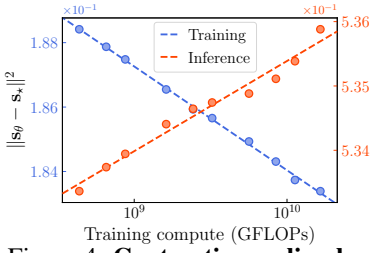


Figure 4: **Contrastive scaling behavior** of $\|\mathbf{s}_\theta - \mathbf{s}_\star\|^2$ measured for supervision region (training) vs. extrapolation region (inference).

These two experiments clearly separate the diffusion model's behavior in the supervision and extrapolation regions. Within the supervision region, the model is driven to learn the empirical score, effectively memorizing the training data (Experiment 4.1)–underfitting *does not* happen. In contrast, in the extrapolation region, the model is led away from the empirical score (Experiment 4.2). This implies that the extrapolation region plays a key role in its generalization, as it is where the model ***selectively underfits*** the empirical score.

4.2 Freedom of Extrapolation: A Mechanism Behind Generalization

As Section 4.1 shows that extrapolation drives the model away from the empirical score, we now ask: *What mechanism enables the model to extrapolate toward such a beneficial score?* In this section, we address this question by introducing the concept of *Freedom of Extrapolation*.

Freedom of extrapolation is motivated by the observation that the learned score function could be sensitive to the size of the supervision region, even though the supervised score remains the same. Specifically, when two diffusion models are trained with the objective $\mathbb{E}_{\mathbf{z}_t\sim q_t}\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)-\mathbf{s}_\star(\mathbf{z}_t,t)\right\|^2$ but with different training distributions q_t, they yield distinct learned scores and generalization capabilities. This leads to our central claim:

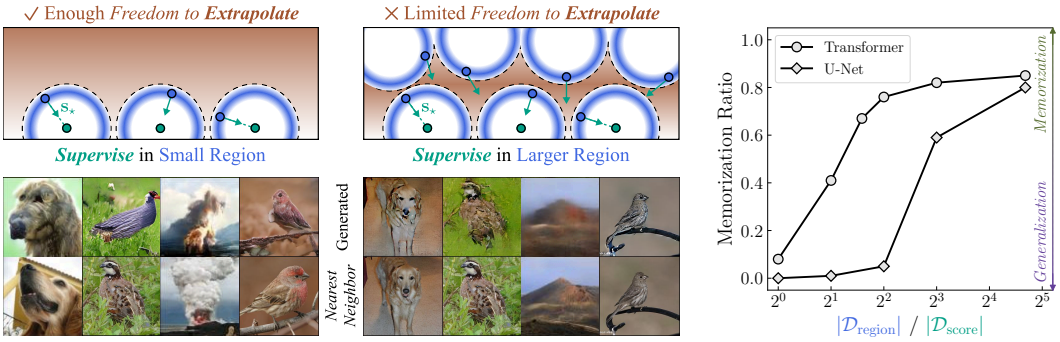
C3. *Diffusion models fail to generalize when the supervision region is broadened.*

We illustrate our intuition in Figure 5. In standard DSM training, as shown in Figure 5a, the network memorizes the training data only within the restricted supervision region (blue) and retains *freedom to extrapolate* elsewhere (brown). Conversely, as shown in Figure 5b, enlarging the supervision region (blue) shrinks the space available for extrapolation, resulting in *limited freedom to extrapolate* (brown). The following controlled experiment formally verifies this claim.

Experiment 4.3 (Experimental Verification)**.** We train a series of models that all learn the same empirical score function, but over varying size of supervision regions–specifically, by changing the (effective) support of $\hat{p}_t$ while keeping $s_\star$ fixed in the objective $\mathbb{E}_{\mathbf{z}_t\sim\hat{p}_t}\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)-\mathbf{s}_\star(\mathbf{z}_t,t)\right\|^2$. To do this, we select two (random) subsets from the ImageNet training set $\mathcal{D}$: the *score subset* $\mathcal{D}_{\text{score}}$ and the *region subset* $\mathcal{D}_{\text{region}}$, with $\mathcal{D}_{\text{score}}\subseteq\mathcal{D}_{\text{region}}\subseteq\mathcal{D}$. In the training objective, two subsets play distinct roles: $\mathcal{D}_{\text{score}}$ defines the empirical score $s_\star$, while $\mathcal{D}_{\text{region}}$ determines the supervision region $\hat{p}_t$. Concretely, we sample $\mathbf{z}_t\sim\hat{p}_t$ by adding noise to points from $\mathcal{D}_{\text{region}}$, and minimize $\mathbb{E}_{\mathbf{z}_t\sim\hat{p}_t}\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t,t)-\mathbf{s}_\star(\mathbf{z}_t,t)\right\|^2$, where $s_\star$ is computed with respect to $\mathcal{D}_{\text{score}}$. Consequently, varying only $|\mathcal{D}_{\text{region}}|$ lets us cleanly assess how the supervision-region size affects generalization.

We present the results in Figure 5, fixing $|\mathcal{D}_{\text{score}}|$ and varying $|\mathcal{D}_{\text{region}}|$ from $|\mathcal{D}_{\text{score}}|$ to $|\mathcal{D}|$. Regardless of architecture, when $|\mathcal{D}_{\text{region}}|=|\mathcal{D}_{\text{score}}|$ (*with freedom of extrapolation*), the model *generalizes*; when $|\mathcal{D}_{\text{region}}|\gg|\mathcal{D}_{\text{score}}|$ (*without freedom of extrapolation*), the model *memorizes*, verifying **C3**.

Discussion. Experiment 4.3 suggests that the narrow supervision of DSM–forcing the model to fit the empirical score in a limited region and leaving it free to extrapolate beyond–is precisely what enables the generalization of diffusion models. Our architecture-agnostic argument contrasts with prior works, which mainly attribute generalization to architectural inductive bias. Overall, the takeaway is: **prior analytic theories for generalization [19, 30] are incomplete, as they only focus on how inductive bias shapes the approximation of the empirical score during supervision (Perspective A); future work should also account for selective underfitting during extrapolation and its connection to generalization (Perspective B).**



(a) **Generalization.** Small supervision region ($|\mathcal{D}_{\text{region}}|=|\mathcal{D}_{\text{score}}|$) ensures *freedom to extrapolate*. (b) **Memorization.** Larger supervision region ($|\mathcal{D}_{\text{region}}|\gg|\mathcal{D}_{\text{score}}|$) limits *freedom to extrapolate*. (c) **Quantitatively,** enlarging the supervision region $\mathcal{D}_{\text{region}}$ increases the ratio of *memorized* samples.

Figure 5: **Experimental verification of *freedom of extrapolation* on ImageNet.** Deliberately limiting the *freedom to extrapolate* transforms the diffusion model from *generalization* to *memorization*.

5 Analyzing generative performance via selective underfitting

In this section, we show how our selective underfitting perspective provides a novel viewpoint on *generative performance* of diffusion models–their ability to produce high-quality samples. We first present our findings on REPA [55] and, motivated by this, we introduce a quantitative scaling law framework for analyzing generative performance (Section 5.1). Throughout, we use FID (*lower is better*, [13]) as the primary metric for measuring generative performance.

Our Findings on REPA. REPA augments the standard DSM loss with an additional regularization term that aligns the model's intermediate representations with features from a pretrained encoder. This simple modification yields a substantial improvement in FID. Interestingly, we find that REPA affects supervision and extrapolation differently. Specifically, we measure the ℓ_2 difference between the learned scores of SiT models trained with and without REPA–both in the supervision region and the extrapolation region. As shown in Figure 6, REPA *barely alters* the model's behavior in the supervision region, but *greatly alters* its extrapolation behavior. This observation not only clarifies REPA's impact on model extrapolation, but also *motivates* a comprehensive tool to understand generative performance.

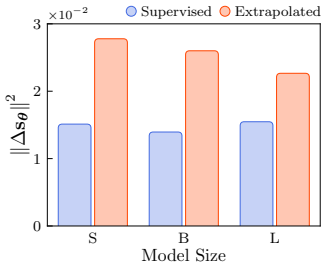


Figure 6: **Effect of REPA on the learned score $\mathbf{s}_{\boldsymbol{\theta}}$.** Greater change $\|\Delta \mathbf{s}_{\boldsymbol{\theta}}\|^2$ occurs for extrapolated scores than for supervised scores.

5.1 Decomposed Analysis of Generative Performance

We introduce our framework for analyzing generative performance, which can be formulated as

$$\text{Generative Performance} := \text{FID} = f_{\text{extrapolation}}(\text{supervision loss } \mathcal{L}). \tag{5}$$

In words, generative performance is decomposed into two components: (i) supervision loss $\mathcal{L}$, which measures how well the empirical score is fitted by the *model*, and (ii) extrapolation function $f_{\text{extrapolation}}$, which characterizes the extrapolation behavior of the *training recipe*. Conceptually, $f_{\text{extrapolation}}$ acts as a transfer function mapping supervision loss to generative performance. Figure 7a illustrates this: for a fixed training recipe A, represented by $f^{\text{A}}_{\text{extrapolation}}$, smaller supervision loss leads to better FID (*solid line*). Changing the training recipe to B, e.g., modifying the architecture or adding a regularization term, alters the extrapolation function to $f^{\text{B}}_{\text{extrapolation}}$ (*dashed lines*).

This framework provides a principled way to assess the efficiency of a *training recipe alone*. A downward shift of the $f_{\text{extrapolation}}$ line indicates greater efficiency–achieving better FID under the same supervision loss–while an upward shift signals reduced efficiency in converting supervision into generative performance. As shown in Figure 7b, consistent with our earlier observations, REPA yields a notably efficient $f_{\text{extrapolation}}$ for diffusion models, explaining much of its empirical success.

Transformer vs. U-Net. We next examine the impact of architecture–an important axis of the training recipe–by comparing transformer (attention), U-Net (convolution), and U-Net (attention+convolution) models. As shown in Figure 7c, (i) architectures with more convolutional layers exhibit more efficient extrapolation behavio (downward line shift); but (ii) for a fixed compute budget (marker size), convolution-based architectures have much worse supervision loss than attention-based counterparts. In summary, transformers, compared to U-Nets, yield worse extrapolation efficiency ($\mathcal{L} \to$ FID), but

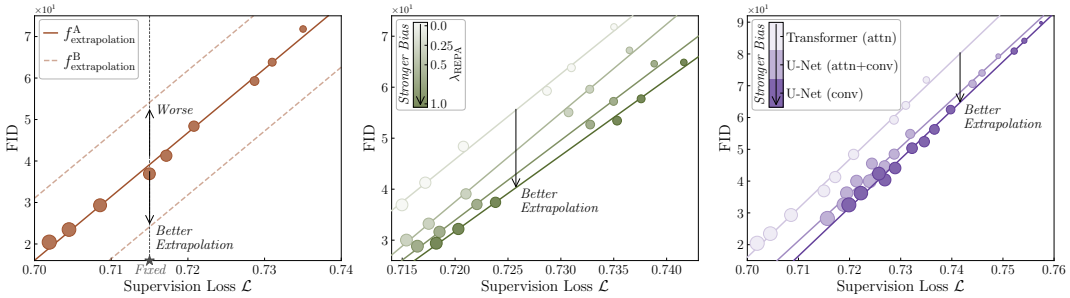


Figure 7: **Illustration of the decomposed analysis of generative performance.** (a) FID (*y-axis*) is plotted as a linear function $f_{\text{extrapolation}}$ (*line*) of supervision loss $\mathcal{L}$ (*x-axis*). $f_{\text{extrapolation}}$ depends on the training recipe (A vs. B) and is more efficient when shifted downward. Our analysis shows that (b) REPA and (c) U-Net (vs. transformers) yield more efficient $f_{\text{extrapolation}}$, i.e., *better extrapolation behavior*. Marker size indicates each model's total training compute (FLOPs).

much scalable supervision efficiency (FLOPs $\to \mathcal{L}$). This decomposition explains the trend toward transformers for large-scale diffusion training: taking the chain of the two factors, transformers achieve better overall compute efficiency (FLOPs $\to$ FID). Our results also suggest that one could combine the best of both worlds–the extrapolation efficiency of U-Nets and the supervision efficiency of transformers–as in recent works [40, 48].

5.2 Unified Hypothesis for Better Generative Performance

In Section 5.1, we introduced a framework for quantifying the extrapolation behavior of different training recipes and presented examples that yield more efficient $f_{\text{extrapolation}}$, potentially leading to improved generative performance—a property of practical interest. This naturally raises the question: *is there a simple principle for designing training recipes that induce more efficient extrapolation?* To this end, we propose a unified hypothesis: *Perception-Aligned Training* (PAT).

PAT. As shown in Section 3, a diffusion model *extrapolates* beyond the supervision region to generate samples; this geometrically corresponds to *interpolating* among training images. Figure 8 shows two scenarios: perceptually distant images may be close in Euclidean space, so naive interpolation produces perceptually bad samples; when the space is aligned with *perception*, it yields perceptually good samples. This highlights the importance of perception and motivates the following hypothesis:

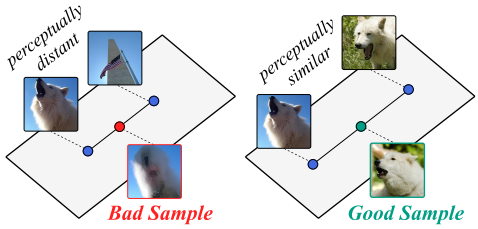
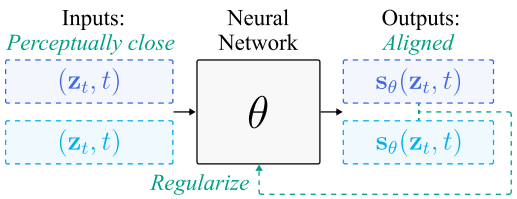


Figure 8: **Good vs. bad samples and their connection to *perception*.**

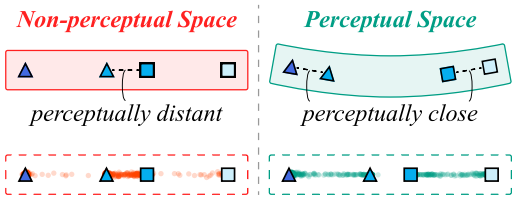
C4. *Aligning a neural network with perception leads to better extrapolation behavior.*

The principle of PAT is illustrated in Figure 9a. Formally, PAT refers to any training recipe that encourages a neural network $\mathbf{s}_{\boldsymbol{\theta}}$ to produce closely aligned outputs $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ for perceptually close inputs $\mathbf{z}_t$. We present three representative instances of PAT in Figures 9b to 9d, which together unify a wide range of practical training approaches, including equivariant or perceptually aligned VAEs [21, 52], representation-learning-based methods [55, 11], and recent architectural advances [40, 48], as detailed in Appendix A.3. We also verify our PAT hypothesis using 2D toy data in Appendix C.8.

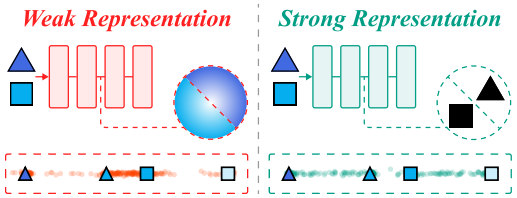
Discussion. We posit that a diffusion model's training efficiency decomposes into **(i)** *supervision efficiency*: how well the model fits the training data, and **(ii)** *extrapolation efficiency*: how this fit translates into generation performance. This decomposition enables a thorough analysis of different *training recipes*, beyond simply comparing per-model FID. With this perspective, PAT offers a unified principle for designing training methods that induce a favorable extrapolation function, i.e., improving **(ii)**. However, PAT can sometimes weaken **(i)**; thus, maximizing generative performance under a fixed compute requires balancing supervision and extrapolation. Finally, we emphasize that **these practical tools are a natural byproduct of Perspective B, impossible with Perspective A.**



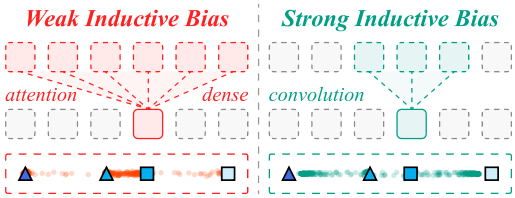
(a) **Definition of PAT.**



(b) **Aligning diffusion space.**



(c) **Aligning representation.**



(d) **Aligning architectural bias.**

Figure 9: **Illustration of *Perception-Aligned Training* (PAT) with toy experiment results.** PAT regularizes a neural network θ to output *aligned* scores $\mathbf{s}_\theta$ for *perceptually close* inputs $\mathbf{z}_t$. Many successful training techniques can be unified under this view, categorized as the three instances.

6 Conclusion

In this work, we argue that diffusion models exhibit distinct behaviors in supervision and extrapolation regions–a perspective we call *selective underfitting*, which offers a novel viewpoint on their generalization and generative performance. Theoretically, however, the mechanism enabling extrapolation remains unclear; future work may further analyze *how the score in the extrapolation region is shaped*, building on our findings about the freedom of extrapolation. Practically, while we do not propose a new training method that optimally balances supervision and extrapolation, our results suggest that *many well-balanced training algorithms can be developed using our decomposed analysis framework*. Ultimately, we expect these insights to extend to other generative models, contributing to a deeper understanding and the development of more powerful generative systems.

References

[1] Quentin Bertrand, Anne Gagneux, Mathurin Massias, and Rémi Emonet. On the closed-form of flow matching: Generalization does not arise from target stochasticity. *arXiv preprint arXiv:2506.03719*, 2025.

[2] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943. ISSN 0008-0659. URL `https://archive.org/details/dli.calcutta.11603`.

[3] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX security symposium (USENIX Security 23)*, pages 5253–5270, 2023.

[4] Defang Chen, Zhenyu Zhou, Can Wang, and Siwei Lyu. Geometric regularity in deterministic sampling of diffusion-based generative models. *arXiv preprint arXiv:2506.10177*, 2025.

[5] Zhengdao Chen. On the interpolation effect of score smoothing. *arXiv preprint arXiv:2502.19499*, 2025.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[8] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[9] Tyler Farghly, Patrick Rebeschini, George Deligiannidis, and Arnaud Doucet. Implicit regularisation in diffusion models: An algorithm-dependent generalisation analysis. *arXiv preprint arXiv:2507.03756*, 2025.

[10] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin Patrick Murphy, and Tim Salimans. Diffusion models and gaussian flow matching: Two sides of the same coin. In *The Fourth Blogpost Track at ICLR 2025*, 2025.

[11] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 23164–23173, 2023.

[12] Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[16] Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. *arXiv preprint arXiv:2505.02831*, 2025.

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[18] Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. *arXiv preprint arXiv:2310.02557*, 2023.

[19] Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*, 2024.

[20] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024.

[21] Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. Eq-vae: Equivariance regularized latent space for improved generative image modeling. *arXiv preprint arXiv:2502.09509*, 2025.

[22] Theodoros Kouzelis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. *arXiv preprint arXiv:2504.16064*, 2025.

[23] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025.

[24] Xiang Li, Yixiang Dai, and Qing Qu. Understanding generalizability of diffusion models requires rethinking the hidden gaussian structure. *Advances in neural information processing systems*, 37:57499–57538, 2024.

[25] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[26] 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.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[28] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.

[29] Matthew Niedoba, Dylan Green, Saeid Naderiparizi, Vasileios Lioutas, Jonathan Wilder Lavington, Xiaoxuan Liang, Yunpeng Liu, Ke Zhang, Setareh Dabiri, Adam Ścibior, et al. Nearest neighbour score estimators for diffusion generative models. *arXiv preprint arXiv:2402.08018*, 2024.

[30] Matthew Niedoba, Berend Zwartsenberg, Kevin Murphy, and Frank Wood. Towards a mechanistic explanation of diffusion model generalization. *arXiv preprint arXiv:2411.19339*, 2024.

[31] 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.

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

[33] Jakiw Pidstrigach. Score-based generative models detect manifolds. *Advances in Neural Information Processing Systems*, 35:35852–35865, 2022.

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[35] Christopher Scarvelis, Haitz Sáez de Ocáriz Borde, and Justin Solomon. Closed-form diffusion models. *arXiv preprint arXiv:2310.12395*, 2023.

[36] Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. *arXiv preprint arXiv:2502.14831*, 2025.

[37] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[38] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[39] Vladislav Sovrasov. ptflops: a flops counting tool for neural networks in pytorch framework, 2018. URL `https://github.com/sovrasov/flops-counter.pytorch`.

[40] Yuchuan Tian, Zhijun Tu, Hanting Chen, Jie Hu, Chao Xu, and Yunhe Wang. U-dits: Downsample tokens in u-shaped diffusion transformers. *arXiv preprint arXiv:2405.02730*, 2024.

[41] Yuchuan Tian, Jing Han, Chengcheng Wang, Yuchen Liang, Chao Xu, and Hanting Chen. Dic: Rethinking conv3x3 designs in diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2469–2478, 2025.

[42] John J Vastola. Generalization through variance: how noise shapes inductive biases in diffusion models. *arXiv preprint arXiv:2504.12532*, 2025.

[43] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[44] Zhengchao Wan, Qingsong Wang, Gal Mishne, and Yusu Wang. Elucidating flow matching ode dynamics with respect to data geometries. *arXiv e-prints*, pages arXiv–2412, 2024.

[45] Binxu Wang and John J Vastola. The unreasonable effectiveness of gaussian score approximation for diffusion models and its applications. *arXiv preprint arXiv:2412.09726*, 2024.

[46] Hanyu Wang, Yujin Han, and Difan Zou. On the discrepancy and connection between memorization and generation in diffusion models. In *ICML 2024 Workshop on Foundation Models in the Wild*, 2024. URL `https://openreview.net/forum?id=ZqG5lo18tq`.

[47] Runqian Wang and Kaiming He. Diffuse and disperse: Image generation with representation regularization. *arXiv preprint arXiv:2506.09027*, 2025.

[48] Shuai Wang, Zexian Li, Tianhui Song, Xubin Li, Tiezheng Ge, Bo Zheng, and Limin Wang. Flowdcn: Exploring dcn-like architectures for fast image generation with arbitrary resolution. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 87959–87977, 2024.

[49] Jiafu Wu, Yabiao Wang, Jian Li, Jinlong Peng, Yun Cao, Chengjie Wang, and Jiangning Zhang. Swin dit: Diffusion transformer using pseudo shifted windows. *arXiv preprint arXiv:2505.13219*, 2025.

[50] Yilun Xu, Shangyuan Tong, and Tommi Jaakkola. Stable target field for reduced variance score estimation in diffusion models. *arXiv preprint arXiv:2302.00670*, 2023.

[51] Yixian Xu, Shengjie Luo, Liwei Wang, Di He, and Chang Liu. Diagnosing and improving diffusion models by estimating the optimal loss value. *arXiv preprint arXiv:2506.13763*, 2025.

[52] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15703–15712, 2025.

[53] Mingyang Yi, Jiacheng Sun, and Zhenguo Li. On the generalization of diffusion model. *arXiv preprint arXiv:2305.14712*, 2023.

[54] TaeHo Yoon, Joo Young Choi, Sehyun Kwon, and Ernest K Ryu. Diffusion probabilistic models generalize when they fail to memorize. In *ICML 2023 workshop on structured probabilistic inference $\{\backslash\&\}$ generative modeling*, 2023.

[55] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

[56] Huijie Zhang, Zijian Huang, Siyi Chen, Jinfan Zhou, Zekai Zhang, Peng Wang, and Qing Qu. Understanding generalization in diffusion models via probability flow distance. *arXiv preprint arXiv:2505.20123*, 2025.

[57] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *arXiv preprint arXiv:2306.09305*, 2023.

Contents

A Discussion

A.1 Related Work

Empirical Score of Diffusion Models. The primary formulations of diffusion models [15, 38, 37] did not explicitly present the analytic form of the empirical score function, and it took several years for the community to recognize that standard training learns precisely this quantity, yielding the expression in Equation (2). This realization has sparked several lines of work: the mainstream thread seeks to understand diffusion model generalization, which we review comprehensively review in the next paragraph. In parallel, the analytic form has been used to study the geometry of ODE sampling trajectories [4, 44] and to improve diffusion model training in various ways [29, 50, 35, 51].

Diffusion Model Generalization. Early works [12, 54] showed a transition from memorization to generalization as the training dataset size grows under a fixed model size. Moreover, if a model perfectly learns the empirical score–i.e., if the DSM loss in Equation (3) is fully minimized–it exactly regenerates the training data. This principle-practice discrepancy has motivated efforts to understand diffusion model generalization. Several works argue and empirically demonstrate that, while the model is supervised with the empirical score, inductive biases lead it to learn a better score–one that produces novel samples–via architectural bias [19, 18, 30], sampling and optimization effects [53], and stochasticity [42].Complementary studies have observed systematic differences between the learned and empirical scores: Wang and Vastola [45], Li et al. [24] report an approximate linear structure in the learned score, and Chen [5], Wang et al. [46] show that it is substantially smoother than its empirical counterpart. Additionally, Bertrand et al. [1] argues that DSM stochasticity may not be the primary driver of generalization, and Zhang et al. [56] proposes a metric for diffusion model generalization that yields accurate scaling laws. Theoretically, Farghly et al. [9] use algorithmic stability analysis to argue that such models can generalize toward the ground-truth score. Compared to these works that focus on how a model *globally underfits* the empirical score, we divide the data space into two regions and reveal *selective underfitting*–where underfitting occurs only in the extrapolation region–as a key principle of generalization.

A.2 Generalization in Diffusion Models vs. Standard Deep Learning

Generalization is a classic problem in deep learning, typically studied in supervised learning, where a model is trained on finite data but is expected to learn a function that goes beyond mere interpolation. Our *selective underfitting* framework differ from this standard generalization studies in two key ways: (1) *training distribution* $\neq$ *test distribution*: whereas supervised learning assumes train and test data from the same distribution, diffusion models operate with distinct distributions, as demonstrated in Section 3. (2) *existence of an empirical score*: diffusion models admit a natural, analytically defined empirical score function that serves as a canonical interpolant on the supervised region, enabling more precise analysis of generalization, where as standard supervised learning lacks such a canonical choice. These distinctions show why our findings stand apart from standard notions of generalization.

A.3 Examples of Perception-Aligned Training Recipes

In Section 5.2, we argued that many practical training techniques can be unified under the Perception-Aligned Training (PAT) framework. Here, we discuss specific works interpreted as PAT, grouped into three main instances.

Aligning Diffusion Space. The most straightforward instance of PAT is training a model in a perceptually aligned space (Figure 9b). Pixel space and standard VAE latent space, on which diffusion models are commonly trained, often entangle non-perceptual factors [36], making Euclidean distance a poor proxy for perceptual similarity. In contrast, a perceptually aligned space allows a smooth score network $\mathbf{s}_{\boldsymbol{\theta}}$ to naturally produce consistent outputs for perceptually similar inputs. From this PAT viewpoint, it is not surprising that many recent works have reported significant performance improvements by constructing perceptually aligned latent spaces, albeit through different strategies: enforcing spatial equivariance (EQ-VAE) [21], aligning with a vision foundation model (VA-VAE, REPA-E) [52, 23], concatenating semantic features from a vision foundation model to form a new perceptual space (ReDi) [22], or regularizing to suppress non-perceptual information [36].

Aligning Representation. Another approach aligns the model's internal representation with perception (Figure 9c). While keeping output supervision unchanged, the training objective encourages

perceptually meaningful intermediate features, which implicitly regularize the outputs. Empirically, this yields more favorable extrapolation (Figure 7b). For example, the pioneering work REPA [55] aligns the model's internal representation with perceptual features from a pretrained encoder such as DINO [31], resulting in significant performance gains. Follow-up works have explored representation learning without external encoders, using self-distillation [16] or contrastive losses [47]. Alternatively, methods like MDT [11] and MaskDiT [57] directly incorporate masked reconstruction objectives, which are known to promote strong internal representations. Despite the variety of strategies, these approaches all share the common mechanism of improving extrapolation by enhancing the model's internal representation.

Aligning Architectural Bias. A third instance is aligning the network's inductive bias with perception (Figure 9d). For images, convolution imparts translational equivariance and locality, aligning well with perceptual invariance such as translation or cropping. This explains the reason why convolution-based architectures tend to induce a more efficient extrapolation function $f_{\text{extrapolation}}$ (Figure 7c). However, as noted in Section 5.1, these designs are less favorable in supervision efficiency (FLOPs $\to \mathcal{L}$), which often limits their practical effectiveness at large training scales. Interestingly, recent works have managed to balance these two efficiency factors and surpass the performance of pure diffusion transformers, including hybrid architectures that incorporate convolution layers into transformers (U-DiT, Swin-DiT) [40, 49], as well as purely convolutional models carefully tuned for efficiency and scalablity (FlowDCN, DiC) [48, 41]. This highlights the importance of balancing extrapolation and supervision efficiency to enable compute-efficient diffusion training.

B Theoretical Details

B.1 Concentration of the Supervision Region

Proposition 3.1 (Supervision region)**.** *Let $\delta \in (0,1)$, empirical data $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$, and $\mathbf{z}_t \sim \hat{p}_t$. Then*

$$\mathbb{P}(\mathbf{z}_t \in \mathcal{T}_t(\delta)) \geq 1-\delta, \quad \mathcal{T}_t(\delta) \colon = \left\{ \mathbf{z} : \exists\, i \in [N], \left| \|\mathbf{z} - \alpha_t \mathbf{x}^{(i)}\| - \sigma_t \sqrt{d} \right| \leq \sigma_t \sqrt{d \log(1/\delta)} \right\}.$$

Proof. We use the well-known fact that for a d-dimensional standard Gaussian vector $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$,

$$\left| \|\boldsymbol{\epsilon}\| - \sqrt{d} \right| \leq \sqrt{d \log(1/\delta)}$$

which holds with probability at least $1-\delta$. Next, recall the δ-supervision region

$$\mathcal{T}_t(\delta) \colon = \left\{ \mathbf{z} : \exists\, i \in [N], \left| \|\mathbf{z} - \alpha_t \mathbf{x}^{(i)}\| - \sigma_t \sqrt{d} \right| \leq \sigma_t \sqrt{d \log(1/\delta)} \right\}.$$

Therefore, for $\mathbf{z}_t \sim \hat{p}_t$,

$$\mathbb{P}\left(\mathbf{z} \notin \mathcal{T}_t(\delta)\right) \leq \frac{1}{N} \sum_{i=1}^{N} \mathbb{P}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}), \mathbf{z} = \alpha_t \mathbf{x}^{(i)} + \sigma_t \boldsymbol{\epsilon}} \left(\left| \|\mathbf{z} - \alpha_t \mathbf{x}^{(i)}\| - \sigma_t \sqrt{d} \right| > \sigma_t \sqrt{d \log(1/\delta)} \right) \leq \delta.$$

□

B.2 Non-Overlapping Property of the Supervision Region

We quantify distributional overlap of supervised shells using the Bhattacharyya coefficient: for two distributions p and q on $\mathbb{R}^d$,

$$0 \leq \int_{\mathbb{R}^d} \sqrt{p(\mathbf{z}) q(\mathbf{z})} \mathrm{d}\mathbf{z}_t \leq 1.$$

For the Gaussians $\mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{x}^{(i)}, \sigma_t \boldsymbol{I})$, the worst-case Bhattacharyya coefficient across shells simplifies to

$$C(t) := \max_{i \neq j} \int_{\mathbb{R}^d} \sqrt{\hat{p}_t^{(i)}(\mathbf{z}_t) \hat{p}_t^{(j)}(\mathbf{z}_t)} \mathrm{d}\mathbf{z}_t = \max_{i \neq j} \left[\exp\left(-\frac{\alpha_t^2 \left\| \mathbf{x}^{(i)} - \mathbf{x}^{(j)} \right\|^2}{8\sigma_t^2} \right) \right].$$

Evaluating this constant on the ImageNet dataset [6] and in the latent space of the SD-VAE [34] yields Figure 3.

C Experimental Details

C.1 Default Experimental Setup

As many of our experiments share a similar setup, we begin by outlining the default configuration. Most implementation details and hyperparameters follow those of SiT [28].

Dataset. We conduct all of our real-world image generation experiments on ImageNet at 256×256, consisting of $N = 1281167$ images across 1000 classes.We apply horizontal flip augmentation during preprocessing, effectively doubling the dataset size to $|\mathcal{D}| = 2N$.

Diffusion. We use simple linear interpolants with $\alpha_t = 1 - t$ and $\sigma_t = t$, and adopt the $\mathbf{v}$-prediction objective. The velocity $\mathbf{v}_{\boldsymbol{\theta}}$ is related to the score $\mathbf{s}_{\boldsymbol{\theta}}$ by the linear relation $\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) = -\frac{t}{1-t}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \frac{1}{1-t}\mathbf{z}_t$. As an exception, models in Experiment 4.3 uses $\mathbf{x}$-prediction for numerical stability, which is also linearly related to the score: $\mathbf{x}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) = \frac{t^2}{1-t}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \frac{1}{1-t}\mathbf{z}_t$. For sampling, we use the `dopri5` adaptive ODE solver with `atol=1e-6` and `rtol=1e-3`.

Training. Models are trained on 4 NVIDIA H100 GPUs with the AdamW optimizer [27] (no weight decay), with $(\beta_1, \beta_2) = (0.9, 0.999)$, a constant learning rate of 10^{-4}, and batch size of 256 for 400K training iterations. We maintain an exponential moving average (EMA) of model weights with a decay rate of 0.9999. All models are latent diffusion models trained on $32 \times 32 \times 4$ latents precomputed using SD-VAE [34]. Additionally, all models are *class-conditional*, modeling 1000 distinct distributions, and trained with a class dropout probability of 0.1 to enable estimation of *unconditional* scores. We employ two different architectures:

(1) Diffusion Transformer: Following the SiT architecture [28], with four configurations (S, B, L, XL), using a patch size of 2.

(2) U-Net: Based on Dhariwal and Nichol [7], with four configurations (XS, S, B, L) varying model channels [20] as $(64, 128, 192, 320)$. Attention layers are applied at resolutions $(1, 2, 4)$, except in the convolution-only U-Net, where attention is disabled.

Unless otherwise specified, we primarily use diffusion transformers (SiT) in our experiments.

Evaluation. For FID evaluation, we follow the protocol of Dhariwal and Nichol [7] using their official implementation. FLOPs are measured with `ptflops` [39], and total training compute is calculated as FLOPs per forward pass multiplied by the number of training iterations.

Supervision and Extrapolation Regions. Many of our experiments measure the expectation of a *quantity*, $\mathbb{E}[\mathcal{Q}]$, separately over the supervision and extrapolation regions. As these regions correspond to continuous probability distributions in data space, we report Monte Carlo estimates by sampling from each region and averaging the computed quantities.

For the supervision region, we sample $\mathbf{z}_t^{\text{sup}}$ from $\hat{p}_t$ by randomly selecting a training data point $\mathbf{x}$ and Gaussian noise $\boldsymbol{\epsilon}$, then running the forward process: $\mathbf{z}_t^{\text{sup}} = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$, identical to the training procedure. For the extrapolation region, we sample $\mathbf{z}_t^{\text{ext}}$ from an inference trajectory, obtained by solving the ODE with a pretrained model starting from random noise $\mathbf{z}_1^{\text{ext}} = \boldsymbol{\epsilon}$. We then average the quantity $\mathcal{Q}$ over N samples:

$$\hat{\mathbb{E}}_t^{\text{sup}}(\mathcal{Q}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{Q}(\mathbf{z}_t^{i,\text{sup}}, t), \; \hat{\mathbb{E}}_t^{\text{ext}}(\mathcal{Q}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{Q}(\mathbf{z}_t^{i,\text{ext}}, t)$$

Often, we measure $\mathcal{Q}$ across all timesteps rather than per timestep. In these cases, we use a timestep-dependent weighting function $\lambda(t)$ to ensure comparable scales across timesteps. Specifically, we sample T random timesteps $\{t_j\}_{j=1}^{N}$ from $\text{Unif}(0, 1)$ and repeat the above process, yielding:

$$\hat{\mathbb{E}}^{\text{sup}}(\mathcal{Q}, \lambda) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{i=1}^{T} \lambda(t_j)\mathcal{Q}(\mathbf{z}_{t_j}^{i,\text{sup}}, t_j), \; \hat{\mathbb{E}}^{\text{ext}}(\mathcal{Q}, \lambda) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{i=1}^{T} \lambda(t_j)\mathcal{Q}(\mathbf{z}_{t_j}^{i,\text{ext}}, t_j)$$

Here, $\{\mathbf{z}_{t_j}^{i,\text{sup}}\}_{j=1}^{T}$ vary only in timesteps for the same data-noise pair $(\mathbf{x}, \boldsymbol{\epsilon})$, and $\{\mathbf{z}_{t_j}^{i,\text{ext}}\}_{j=1}^{T}$ vary only in timestep along the same inference trajectory. For the rest of the appendix, we simply specify N and $\mathcal{Q}$ (and T and λ, if integrating over all timesteps), using the above notations.

C.2 Details on CFG Paradox Experiment (Example 3.1)

In Example 3.1, we plot the difference between conditional and unconditional scores, $\|\mathbf{s}^c - \mathbf{s}^\varnothing\|$, in both the supervision and extrapolation regions. The difference is measured separately at 100 different timesteps, $t = \{0.01, 0.02, \ldots, 1.00\}$, using $\hat{\mathbb{E}}_t^{\text{sup}}(\mathcal{Q})$ and $\hat{\mathbb{E}}_t^{\text{exp}}(\mathcal{Q})$ of Appendix C.1 as described in Appendix C.1. For the supervision region, we use $\mathcal{Q}(\mathbf{z}_t, t) = \|\mathbf{s}_\star^c(\mathbf{z}_t, t) - \mathbf{s}_\star^\varnothing(\mathbf{z}_t, t)\|$ (difference of the empirical score), and for the extrapolation region, $\mathcal{Q}(\mathbf{z}_t, t) = \|\mathbf{s}_{\boldsymbol{\theta}}^c(\mathbf{z}_t, t) - \mathbf{s}_{\boldsymbol{\theta}}^\varnothing(\mathbf{z}_t, t)\|$ (difference of the learned score) for the extrapolation region, with $N = 200$ samples per timestep. For better visualization, we plot the median (dashed line) and the $10^{\smile}90\%$ percentile range (shaded area) for each timestep.

C.3 Details on Extrapolation During Inference Experiment (Experiment 3.3)

In Experiment 3.3, we plot the $r_\star$ values defined in Equation (4) measured in the supervision and extrapolation regions (Figure 1b). Similarly to Appendix C.2, we measure the $r_\star$ values on 100 different timesteps, with $N = 100$ samples per timestep. For clarity, we plot each training data point and each sampling trajectory as a separate line, rather than showing averages.

C.4 Details on Memorization Experiment (Experiment 4.1)

In Experiment 4.1, we sample $N = 200$ images $\mathbf{x}^{(i)} \in \mathcal{D}$ from the training dataset. For each image, we construct a noisy image $\mathbf{z}_t = \alpha_t \mathbf{x}^{(i)} + \sigma_t \boldsymbol{\epsilon}$ at 21 timesteps $t = 0.00, 0.05, 0.10, \ldots, 1.00$. We then run ODE sampling ($t \to 0$) from each noisy image $\mathbf{z}_t$ and compare the final output to the original image $\mathbf{x}^{(i)}$. Figure 3 (left) shows a qualitative example, while Figure 3 (right) plots the memorization ratio, defined as the fraction of sampled images for which the ℓ_2-closest image in the training set is the original $\mathbf{x}^{(i)}$ (following the calibrated ℓ_2 metric in Carlini et al. [3]). Additionally, the overlap coefficient from Appendix B.2 is measured on the class-conditional distribution for a single class (red panda, id $= 387$).

C.5 Details on Supervision vs. Extrapolation Experiment (Experiment 4.2)

In Experiment 4.2, we measure the score error $\|\mathbf{s}_\theta - \mathbf{s}_\star\|^2$ in the supervision and extrapolation regions across different model sizes. The score error in each region is computed as $\hat{\mathbb{E}}^{\text{sup}}(\mathcal{Q}, \lambda)$ and $\hat{\mathbb{E}}^{\text{ext}}(\mathcal{Q}, \lambda)$ described in Appendix C.1, using the following parameters: $\mathcal{Q}(\mathbf{z}_t, t) = \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \mathbf{s}_\star(\mathbf{z}_t, t)\|^2$, $\lambda(t) = \frac{t^2}{(1-t)^2}$, $N = 1000$, and $T = 100$. The weighting function $\lambda(t)$ is chosen so that $\lambda(t)\mathcal{Q}(\mathbf{z}_t, t) = \|\mathrm{v}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \mathbf{v}_\star(\mathbf{z}_t, t)\|^2$, which reflects the $\mathbf{v}$-prediction objective and ensures uniform aggregation of the score error across timesteps.

C.6 Details on Freedom of Extrapolation experiment (Experiment 4.3)

We detail the setup of the *freedom of extrapolation* experiment, including subset construction, training procedure, and evaluation metric.

Score and Region Subsets. As described in Section 4.2, we use two nested subsets $\mathcal{D}_{\text{score}} \subseteq \mathcal{D}_{\text{region}}$, drawn from the ImageNet training set $\mathcal{D}$. These subsets play distinct roles in the DSM objective: $\mathcal{D}_{\text{score}}$ defines the target (empirical) score $\mathbf{s}_\star$, while $\mathcal{D}_{\text{region}}$ defines the data distribution $\hat{p}_t$ for sampling inputs in the DSM objective:

$$\mathcal{L}_{\text{DSM}}(t) = \mathbb{E}_{\mathbf{z}_t \sim \hat{p}_t} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \mathbf{s}_\star(\mathbf{z}_t, t) \right\|^2. \tag{6}$$

To construct $\mathcal{D}_{\text{score}}$, we sample 100 images per ImageNet class, resulting in $|\mathcal{D}_{\text{score}}| = 1000 * 100 = 10^5$ images. To obtain $\mathcal{D}_{\text{region}}$ of varying sizes while preserving $\mathcal{D}_{\text{score}}$, we augment $\mathcal{D}_{\text{score}}$ with additional images sampled uniformly without replacement from the remaining images in each class. Fixing $\mathcal{D}_{\text{score}}$ and varying $\mathcal{D}_{\text{region}}$, we train both SiT-L and UNet-B models.

Training (Importance Sampling). Directly applying the DSM loss is infeasible when $\mathcal{D}_{\text{score}} \neq \mathcal{D}_{\text{region}}$, since it would require computing the empirical score at every iteration. To address this, we use an importance-sampling technique inspired by Niedoba et al. [29]. For a given $\mathbf{z}_t$, we define an auxiliary distribution over $\mathcal{D}_{\text{score}} := \{\mathbf{x}^{(i)}\}_{i=1}^n$:

$$r_{\mathbf{z}_t}(\mathbf{y}) \propto \exp\left(-\frac{\|\mathbf{z}_t - \alpha_t \mathbf{y}\|^2}{2\sigma_t^2}\right) :\ = \mathrm{softmax}\left(-\frac{\|\mathbf{z}_t - \alpha_t \mathbf{y}\|^2}{2\sigma_t^2}\right),$$

Using $r_{\mathbf{z}_t}$, we define the alternative loss:

$$\mathcal{L}'_\theta(t) :\ = \mathbb{E}_{t,\mathbf{x},\mathbf{z}_t,\mathbf{y}} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \frac{\mathbf{z}_t - \alpha_t \mathbf{y}}{\sigma_t^2} \right\|^2, \quad \mathbf{x} \sim \mathrm{Unif}(\mathcal{D}_{\text{region}}),\ \mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon},\ \mathbf{y} \sim r_{\mathbf{z}_t}. \quad (7)$$

In words, we sample $\mathbf{x}$ from $\mathcal{D}_{\text{region}}$ and form $\mathbf{z}_t$ as in standard DSM training, then additionally sample $\mathbf{y}$ from $r_{\mathbf{z}_t}$.

Equivalence Proof. We now show that Equation (6) and Equation (7) are equivalent up to a constant:

$$\begin{aligned}
&\mathbb{E}_{t,\mathbf{x},\mathbf{z}_t,\mathbf{y}} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \frac{\mathbf{z}_t - \alpha_t \mathbf{y}}{\sigma_t^2} \right\|^2 \\
=&\mathbb{E}_{t,\mathbf{x},\mathbf{z}_t} \left[\int_{\mathbf{y}} \left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \frac{\mathbf{z}_t - \alpha_t \mathbf{y}}{\sigma_t^2} \right\|^2 r_{\mathbf{z}_t}(\mathbf{y}) d\mathbf{y} \right] \\
=&\mathbb{E}_{t,\mathbf{x},\mathbf{z}_r} \left[\left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \int_{\mathbf{y}} \frac{\mathbf{z}_t - \alpha_t \mathbf{y}}{\sigma_t^2} r_{\mathbf{z}_t}(\mathbf{y}) \right\|^2 d\mathbf{y} \right] + \mathcal{C} \\
=&\mathbb{E}_{t,\mathbf{x},\mathbf{z}_t} \left[\left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) + \sum_{\mathbf{y} \in \mathcal{D}_{\text{score}}} \frac{\mathbf{z}_t - \alpha_t \mathbf{y}}{\sigma_t^2} \mathrm{softmax}\left(-\frac{\|\mathbf{z}_t - \alpha_t \mathbf{y}\|^2}{2\sigma_t^2}\right) \right\|^2 \right] + \mathcal{C} \\
=&\mathbb{E}_{t,\mathbf{x},\mathbf{z}_t} \left[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \mathbf{s}_\star(\mathbf{z}_t, t)\|^2 \right] + \mathcal{C}.
\end{aligned}$$

where $\mathcal{C}$ is a constant independent of $\mathbf{s}_{\boldsymbol{\theta}}$. The second equality applies a variance decomposition over $\mathbf{y} \sim r_{\mathbf{z}_t}$ (the resulting variance term does not depend on $\mathbf{s}_{\boldsymbol{\theta}}$ and is absorbed into $\mathcal{C}$), and the last equality uses the definition of the empirical score $\mathbf{s}_\star(\mathbf{z}_t, t)$ in Equation (2). □

In summary, this loss is equivalent to the original DSM loss up to a constant, but avoids explicit computation of the empirical score at every iteration, enabling efficient training when $\mathcal{D}_{\text{score}} \neq \mathcal{D}_{\text{region}}$. We implement $r_{\mathbf{z}_t}$ using GPU L2 indexing with Faiss [17, 8], incurring minimal overhead.

Evaluation (Memorization Ratio). After training, we evaluate each model's generalization beyond the score subset $\mathcal{D}_{\text{score}}$ using the *calibrated ℓ_2 metric* from Carlini et al. [3]. For a generated image $\mathbf{x}$, we compute

$$\frac{\|\mathbf{x} - \mathbf{x}'^{(1)}\|_2^2}{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x} - \mathbf{x}'^{(i)}\|_2^2},$$

where $\mathbf{x}'^{(i)}$ is the i-th nearest training image in $\mathcal{D}_{\text{score}}$. A value near 0 indicates memorization (the sample is unusually close to a training image); a value near 1 indicates no memorization. We set $n = 50$ and, for each model, generate 256 images to compute the memorization ratio.

C.7 Details on Decomposed Analysis of Generative Performance (Section 5.1)

In this section, we provide details for our decomposed analysis of generative performance.

Supervision Loss. The supervision loss $\mathcal{L}$ quantifies how well the model $\mathbf{s}_\theta$ approximates the empirical score $\mathbf{s}_\star$ within the supervision region. It is computed as $\hat{\mathbb{E}}^{\text{sup}}(\mathcal{Q}, \lambda)$ described in Appendix C.1, where $\mathcal{Q}(\mathbf{z}_t, t) = \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \mathbf{s}_\star(\mathbf{z}_t, t)\|^2$, $\lambda(t) = \frac{t^2}{(1-t)^2}$, $N = 1000$, and $T = 100$. These are identical parameters to those used to compute score errors in Experiment 4.2.

REPA. We closely follow the original implementation [55] to incorporate REPA into SiT. Specifically, we align the intermediate representation of SiT at depth 8 with features from DINOv2-B [31] using negative cosine similarity loss weighted by λ_{REPA}. To study the effect of stronger representation alignment on the extrapolation function, we vary λ_{REPA} in $\{0.0, 0.25, 0.5, 1.0\}$. For each λ_{REPA} (each line in the scaling plot Figure 9c), each data point corresponds to a model from SiT-$\{\text{S}, \text{B}\}$ trained for $\{200\text{K}, 300\text{K}, 400\text{K}\}$ iterations.

Transformer vs. U-Net. We compare three architectures: Transformer (attention), U-Net (attention+convolution), and U-Net (convolution), with specifications detailed in Appendix C.1. Figure 9d shows scaling plots for SiT-$\{S, B, L\}$ and U-Net-$\{XS, S, B, L\}$, with each model evaluated at $\{200K, 300K, 400K\}$ training iterations.

C.8 Verification of Perception-Aligned Training Hypothesis (Section 5.2)

In this section, we verify our central Hypothesis **C4** proposed in Section 5.2. Using a carefully designed 2D toy dataset, we illustrate how PAT–*aligning a neural network with perception*–leads to *better extrapolation*, and, consequently, perceptually better samples. Importantly, models are trained on a tiny training set $\mathcal{D}$, making *supervision nearly perfect* (i.e., models fit the training region almost exactly), which allows us to study *extrapolation in isolation*. In this setting, in Equation (5), the supervision loss is fixed at $\mathcal{L} \approx 0$, so generative performance (perceptual quality of samples) directly reflects the extrapolation function $f_{\text{extrapolation}}$.

Setup. Our 2D toy data is designed to be *analogous* to image data, as shown in Figure 10.

(S1) Training data $\mathcal{D} := \{▲(-1, 0), ▲(-0.2, 0), ■(0.2, 0), □(1, 0)\}$, small enough to fit perfectly.

(S2) Two points are Euclidean close if their (x, y)-distance is small (e.g. ▲, ■). We (arbitrarily) define two points as *perceptually close if angular distance is small* (e.g. ▲, ▲ or ■, □).

(S3) As shown in Figure 8 (middle), "bad" samples interpolate between Euclidean-close data (▲ – ■), while "good" samples interpolate between perceptually close data (▲ – ▲ or ■ – □).

Implementation of PAT. Based on this setup, we describe the implementation of PAT Instances. For the baseline (without PAT), we use a sufficiently large 2-layer MLP, except for the last instance.

(P1) *Aligning diffusion space*: Instead of the default Cartesian space (x, y) (non-perceptual), we train a model in polar space $(r, \cos\theta, \sin\theta)$, emphasizing the angular (perceptual) component.

(P2) *Aligning representation*: Since directly manipulating a neural net's internal representation is nontrivial, we instead train kernel ridge regression models with a Gaussian kernel. The baseline uses non-perceptual features (x, y), while PAT uses perceptual features $(r, \sin\theta, \cos\theta)$.

(P3) *Aligning architectural bias*: Analogous to translational equivariance in image convolutions, we use a radial-equivariant MLP as a stronger inductive bias.

Results. Generated samples are shown in Figures 9b to 9d: baseline (left) vs. PAT (right).

(R1) Baselines generate "bad" samples by interpolating between Euclidean-close points, due to the lack of perceptual bias. In contrast, all PAT instances generate "good" samples by interpolating between perceptually close points. As supervision is kept near-perfect, this verifies **C4**: *PAT, purely by improving extrapolation, enables generating dramatically better samples*.

(R2) All three PAT implementations yield similarly shaped sample distributions, showing that, despite methodological differences, they operate on the same underlying principle.

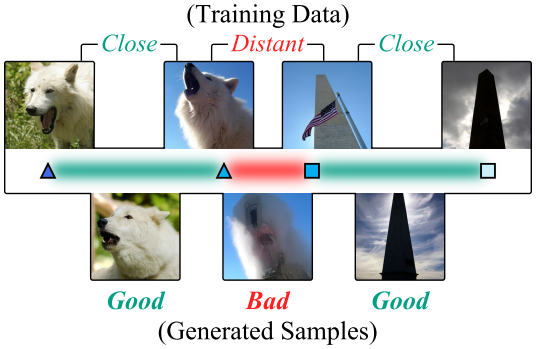


Figure 10: **Comparison of "good" vs. "bad" samples** on ImageNet, illustrated with an analogy to a 2D toy example. Toy training data: (▲, ▲, ■, □); generated samples: , .