

# Studying Classifier(-Free) Guidance From a Classifier-Centric Perspective

Xiaoming Zhao Alexander G. Schwing  
 University of Illinois Urbana-Champaign  
 {xz23, aschwing}@illinois.edu

## Abstract

*Classifier-free guidance has become a staple for conditional generation with denoising diffusion models. However, a comprehensive understanding of classifier-free guidance is still missing. In this work, we carry out an empirical study to provide a fresh perspective on classifier-free guidance. Concretely, instead of solely focusing on classifier-free guidance, we trace back to the root, i.e., classifier guidance, pinpoint the key assumption for the derivation, and conduct a systematic study to understand the role of the classifier. We find that both classifier guidance and classifier-free guidance achieve conditional generation by pushing the denoising diffusion trajectories away from decision boundaries, i.e., areas where conditional information is usually entangled and is hard to learn. Based on this classifier-centric understanding, we propose a generic post-processing step built upon flow-matching to shrink the gap between the learned distribution for a pre-trained denoising diffusion model and the real data distribution, majorly around the decision boundaries. Experiments on various datasets verify the effectiveness of the proposed approach.*

## 1. Introduction

Conditional generation, *e.g.*, class-to-image, text-to-image, or image-to-video, is omnipresent as it provides a compelling way to control the output. Ideally, conditional generation results are both *diverse* and of *high-fidelity*. Namely, the generative models’ outputs align with the conditioning information perfectly and diligently follow the training data diversity. However, there is a trade-off between high-fidelity and diversity: without constraining diversity there are always possibilities to sample from areas on the data distribution manifold that are not well-trained. Thus, trading diversity for fidelity is a long-standing problem and the community has developed various approaches, *e.g.*, the truncation trick for generative adversarial nets (GANs) [7, 28], low-temperature sampling for probabilistic models [2], or temperature control in large language models [1, 19].

More recently, to trade diversity and fidelity in denois-

ing diffusion models [25, 35, 53, 58], several techniques have been developed [16, 26, 31], from which classifier-free guidance [24] emerged as the de-facto standard. For instance, classifier-free guidance, especially at sufficient scale, has been found to be critical for high-quality text-to-image [50] and text-to-3D [47] generation.

Despite its popularity, we think a solid understanding of classifier-free guidance is missing. Recently, several efforts provide insights by studying classifier-free guidance from a theoretical perspective [6, 12, 61] showing that sampling from classifier-free guidance is not the same as sampling from a sharpened distribution.

Instead of solely focusing on classifier-free guidance as done in the works mentioned above, we trace back to the root of classifier-free guidance, *i.e.*, classifier guidance [16]. It is classifier guidance that decomposes the *conditional* generation into a combination of an *unconditional* generation and a classifier prediction. Classifier-free guidance directly mimics this decomposition, replacing the classifier by randomly dropping conditioning information during training [24]. This connection motivates us to carefully study classifier guidance’s derivation and its behavior.

We first pinpoint the key assumption that underlies the decomposition of classifier guidance. It also turns out to be the cornerstone of classifier-free guidance, due to the connection mentioned above. However, we find that the assumption does not generally hold. This issue results in different behaviors for 1) a vanilla denoising diffusion conditional generation; and 2) a generation that follows the decomposition of classifier guidance as well as classifier-free guidance. On synthetic 1D data, the vanilla conditional generative model produces straight-like denoising trajectories while the decomposed version results in distorted trajectories that are pushed away from the classifier’s decision boundary. This discrepancy is exacerbated with the commonly used large guidance scale.

The above observation motivates us to further study the sensitivity of classifier guidance to the accuracy of the *classifier*. We find that classifier guidance generations are dominated by the behavior of the classifier that provides guidance. In other words, the conditional generations from clas-

sifier guidance can be arbitrarily low-quality if the classifier provides entirely misleading guidance. A similar observation is obtained for classifier-free guidance as well.

In order to improve the fidelity for conditional generation from a trained denoising diffusion model (even if the guidance is severely off), we propose a generic postprocessing step. Concretely, we push the distribution of conditional generations to the real distribution via training a rectified flow [40, 41] between samples from the trained model and their nearest neighbors in the real data. We verify our proposed approach on various datasets. In summary, our contributions are two-fold:

1. we provide a systematic empirical study of classifier guidance and classifier-free guidance from a classifier-centric perspective to obtain an intuitive understanding;
2. we propose a generic framework that improves the fidelity of conditional generations for classifier guidance and classifier-free guidance via a flow matching-based postprocessing step.

## 2. Related Works

**Trading diversity for fidelity in conditional generation** is a long-standing problem that has been actively studied by the community. For probabilistic models trained with the maximum likelihood objective, Ackley et al. [2] propose low-temperature sampling to effectively focus on the mode of the learned distribution, borrowing ideas from statistical mechanics [43]. This technique has also been employed beneficially for high-quality image synthesis [33, 45]. Recent large language models (LLMs) [1, 19] also exploit this idea, keeping a balance between creativity and determinism via temperature control during next token prediction via the learned probability model [8]. For image synthesis with generative adversarial nets (GANs), the truncation trick [7, 28] was developed to enforce sampling from a truncated normal distribution rather than the standard normal prior. This encourages conditional generations to remain close to the mode of the data distribution observed during training, preventing them from diverging too far. More recently, denoising diffusion models have demonstrated impressive generation capabilities in various domains [11, 36, 47, 50]. Classifier-free guidance [24], built upon classifier guidance [16], has emerged as a standard for controlling conditional generations in the era of denoising diffusion models. Our work contributes to the understanding of the trade-off between diversity and fidelity in the field of denoising diffusion models. For this, we carefully study classifier(-free) guidance. The gained insights motivate a generic postprocessing step that improves the fidelity for both classifier guidance and classifier-free guidance.

**Generation with guidance** is closely related to our study. Techniques discussed in the preceding paragraph, except classifier guidance, solely require trained generative mod-

els, *e.g.*, the generator in GANs, to control the diversity and fidelity trade-off. In contrast, guidance relies on a separate model to influence the conditional generation. Rejection sampling [9] is an active area of research in this direction. For GANs, prior works use the discriminator paired with the generator to reject generations for which the discriminator has high confidence [4, 57]. Alternatively, Che et al. [10] utilize the discriminator to reject samples in the latent space. For variational autoencoders (VAEs) [34], learnable acceptance functions have been studied for both prior [3, 5] and posterior [21, 27] rejection sampling. Other works explore pre-trained classifiers to provide guidance. Razavi et al. [49] use a classifier trained on ImageNet [15] to reject samples that cannot be well-recognized. Thanks to the recent progress of representation learning, several prior works exploit CLIP [48] to provide guidance on conditional generations with GANs [20, 46]. With respect to denoising diffusion models, Kim et al. [31] improve the quality of a pre-trained model via refining denoising trajectories with guidance from a discriminator that distinguishes between real and fake denoising paths. Additionally, classifier guidance [16] influences the conditional generation with a classifier trained to predict conditional information on the denoising path. Inspired by generation with guidance, our study focuses on a classifier-centric perspective, providing an intuitive understanding of the behavior of classifier guidance and classifier-free guidance.

**Classifier-free guidance** has attracted more and more attention in the community. Theoretically, several recent works clarify that sampling with classifier-free guidance does not correspond to sampling from a tilted distribution [6, 12, 60, 61], a misconception that is popular in the community. Bradley and Nakkiran [6] further prove that CFG is equivalent to the predictor-corrector mechanism [54] in the continuous-time limit. Empirically, prior works improve generation quality by refining classifier-free guidance. Sadat et al. [52] dynamically adjust the scale of classifier-free guidance to improve the generation diversity. Lin and Yang [39] argue that classifier-free guidance essentially behaves as a perceptual loss and propose to incorporate a self-perceptual objective during training. Chung et al. [13] introduce CFG++ to mitigate the issue of an off-manifold denoising path via a refined sampling formulation and a small scale. Different from these works that solely focus on classifier-free guidance, we instead trace back to the origin, *i.e.*, classifier guidance [16]. We systematically study the role the classifier plays in the performance of classifier guidance and found classifier guidance essentially pushes the generation away from the decision boundary. Furthermore, we demonstrate that this is also true for classifier-free guidance. Based on this understanding from a classifier-centric perspective, we propose a postprocessing step that effectively aligns generations to the

real data distribution for classifier(-free) guidance.

### 3. Analysis

#### 3.1. Conditional Generation as Denoising Diffusion

The goal of conditional generation is to sample the data of interest  $\mathbf{x}_0$ , *e.g.*, images, from a conditional distribution, *i.e.*,  $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|c)$ . Here,  $c$  is the conditioning information, *e.g.*, class labels. Note, hereafter we use  $\theta$  to subsume all learnable parameters for simplicity.

In this work, we focus on denoising diffusion models [25, 53, 58]. A denoising diffusion process generates data from white noise by introducing a sequence of latent variables  $\mathbf{x}_{1:T} \triangleq \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$  that form a Markov chain:

$$p_\theta(\mathbf{x}_0|c) = \int p_\theta(\mathbf{x}_0, \mathbf{x}_{1:T}|c) d\mathbf{x}_{1:T} \quad (1)$$

$$\triangleq \int p(\mathbf{x}_T|c) \prod_{t=0}^{T-1} p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, c) d\mathbf{x}_{1:T} \quad (2)$$

$$\approx \int p(\mathbf{x}_T) \prod_{t=0}^{T-1} p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, c) d\mathbf{x}_{1:T}. \quad (3)$$

The last step is due to  $p(\mathbf{x}_T|c)$  being almost identical to an isotropic Gaussian, independent of the condition  $c$ .

Following DDPM [25],  $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, c)$  is defined as a Gaussian  $\mathcal{N}(\mathbf{x}_t; \mu_\theta(\mathbf{x}_{t+1}, t+1, c), (1 - \alpha_{t+1})\mathbf{I})$ , trying to reverse a forward diffusion process. Here  $\{\alpha_t\}_{t=1}^T$  is a pre-defined schedule for the forward diffusion process and  $\mu_\theta(\mathbf{x}_{t+1}, t+1, c)$  is tasked to predict the corresponding  $\mathbf{x}_t$  in the forward diffusion process. Specifically, the forward diffusion process gradually corrupts the clean data  $\mathbf{x}_0$  with Gaussian noise:  $\mathbf{x}_{t+1} \sim q(\mathbf{x}_{t+1}|\mathbf{x}_t) \triangleq \mathcal{N}(\mathbf{x}_{t+1}; \sqrt{\alpha_{t+1}}\mathbf{x}_t, (1 - \alpha_{t+1})\mathbf{I}), \forall t \in \{0, \dots, T-1\}$ .

Notably, with  $\bar{\alpha}_t \triangleq \prod_{s=1}^t \alpha_s$ , we have  $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$ , where  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . Therefore, Ho et al. [25] propose to reduce the learning of  $\mu_\theta(\mathbf{x}_{t+1}, t+1, c)$  to predicting the noise with  $\epsilon_\theta(\mathbf{x}_{t+1}, t+1, c)$  as we have  $\mu_\theta(\mathbf{x}_{t+1}, t+1, c) =$

$$\frac{1}{\sqrt{\bar{\alpha}_t}} \left( \mathbf{x}_{t+1} - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_{t+1}, t+1, c) \right). \quad (4)$$

When leveraging the connection between the denoising diffusion model and score matching [54, 58], we have

$$\epsilon_\theta(\mathbf{x}_t, t, c) = -\sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|c). \quad (5)$$

#### 3.2. Classifier Guidance Revisited

Dhariwal and Nichol [16] propose *classifier guidance* to decompose the conditional denoising diffusion process in Eq. (3) as follows:

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, c) = Z p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}) p_\theta(c|\mathbf{x}_t). \quad (6)$$

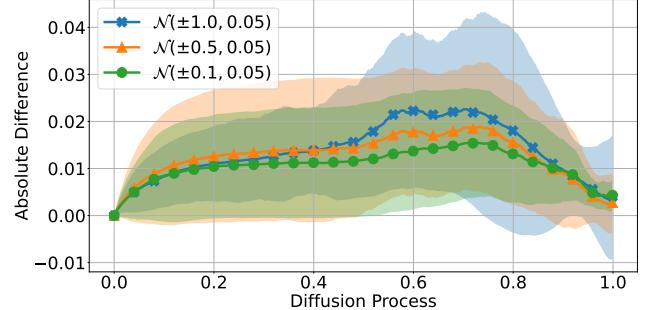


Figure 1. **Classifier guidance decomposition (Eq. (6)) does not always hold.** We apply classifier guidance on 1D data from  $\mathcal{N}(\pm 1.0, 0.05)$ ,  $\mathcal{N}(\pm 0.5, 0.05)$ , and  $\mathcal{N}(\pm 0.1, 0.05)$  respectively. The denoising diffusion process starts from left to right. For each dataset, we train a vanilla conditional diffusion model and a decomposed version, *i.e.*, an unconditional diffusion model and a classifier. We generate 20k samples (10k for each class) from both sides of Eq. (6) with the same initial noises and compute the absolute differences for each step in the denoising diffusion process. This plot shows the average as well as the standard deviation for the difference. Apparently, the classifier guidance decomposition doesn't hold with equality.

$Z$  is a normalizing factor independent of  $\mathbf{x}_t$ .  $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$  is an unconditional denoising diffusion process and  $p_\theta(c|\mathbf{x}_t)$  is a classifier used to predict the probability that  $\mathbf{x}_t$  aligns with the conditioning information  $c$ .

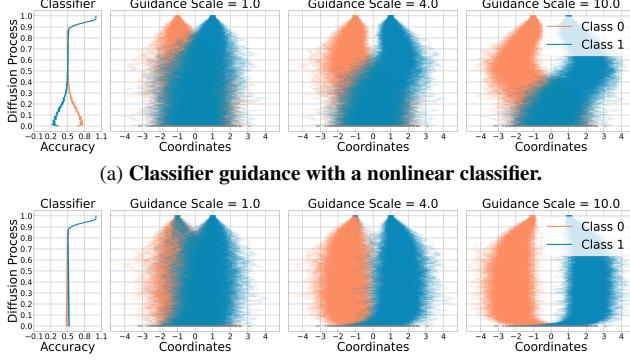
When revisiting the derivation for Eq. (6), we find the key step boils down to the following definition appearing as Eq. (30) in [16]'s Appendix H:

$$\hat{q}(\mathbf{x}_{t+1}|\mathbf{x}_t, c) \triangleq q(\mathbf{x}_{t+1}|\mathbf{x}_t). \quad (7)$$

Note,  $\hat{q}(\mathbf{x}_{t+1}|\mathbf{x}_t, c)$  is a newly-defined conditional forward diffusion process.

At a high level, Eq. (7) tries to convert any *conditional* forward diffusion process, *i.e.*,  $\hat{q}(\mathbf{x}_{t+1}|\mathbf{x}_t, c)$ , into an *unconditional* forward diffusion process, *i.e.*,  $q(\mathbf{x}_{t+1}|\mathbf{x}_t)$ . Based on Eq. (7), Dhariwal and Nichol [16] derive that the reverse process of  $\hat{q}(\mathbf{x}_{t+1}|\mathbf{x}_t, c)$  can be decomposed into a combination of an unconditional denoising diffusion process and a classifier prediction as in Eq. (6) (or Eq. (2) and Eq. (60) in [16]). For details, please refer to Appendix H in [16].

However, it is questionable whether Eq. (7) always holds: why should a conditional denoising process behave identical to an unconditional one? If Eq. (7) does not hold everywhere, the two sides in Eq. (6) may differ too. Indeed, our experiments on synthetic 1D data verify our suspicion as shown in Fig. 1. Furthermore, not only does the vanilla conditional model (left side of Eq. (6)) behave differently from the proposed decomposition (right side of Eq. (6)), but different instantiations of the classifier  $p_\theta(c|\mathbf{x}_t)$  will produce significantly divergent behaviors as well. The “guidance scale = 1” plots in Fig. 2a and 2b clearly illustrate this. **Large classifier guidance scale  $w$**  introduced by Dhariwal and Nichol [16] will amplify the difference demonstrated



(b) **Classifier guidance with a linear classifier.** For the second plot, *i.e.*, if the guidance scale is 1.0: the trajectories for Class 0 (the orange lines) behave similarly to those of Class 1 (blue lines) with a nontrivial portion of incorrect generations. Those lines are not easy to see as they overlap with the Class 1 curves. Please refer to Fig. S8 for class-wise visualization.

**Figure 2. Classifier guidance behavior is dominated by the classifier.** We apply denoising diffusion models with classifier guidance on a 1D dataset with data from  $\mathcal{N}(\pm 1.0, 0.05)$ . The classifiers in Fig. 2a and 2b differ. The denoising diffusion process for all plots starts from the bottom to the top. In Fig. 2a and 2b, the first plot demonstrates the classifier’s accuracy on a validation set for each class through the diffusion process, *i.e.*,  $p_\theta(c|\mathbf{x}_t)$  in Eq. (6), while the remaining three plots display the diffusion trajectories with different guidance scales. We observe: 1) classifier guidance essentially pushes the diffusion process away from the classifier’s decision boundary that is around the origin; and 2) different classifiers can produce entirely different trajectories (Fig. 2a vs. 2b). Since we use the same initial noise and the same unconditional diffusion model, *i.e.*,  $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$  in Eq. (6), for all plots, differences are solely due to the classifier.

above. Specifically, Dhariwal and Nichol [16] suggest increasing the impact of the classifier with  $w > 1$  and sampling from a distribution that is skewed towards high classifier confidence:

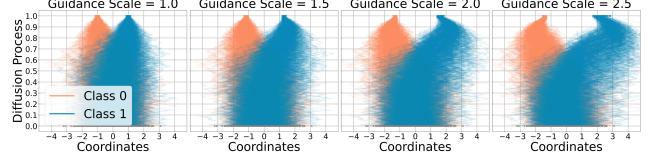
$$\mathbf{x}_t \sim p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}) p_\theta(c|\mathbf{x}_t)^w. \quad (8)$$

Similar to Eq. (5), Dhariwal and Nichol [16] show that Eq. (8) can be re-formulated such that  $\mathbf{x}_t$  can be sampled via predicting the following noise  $\tilde{\epsilon}_\theta(\mathbf{x}_t, t, c) \triangleq$

$$\epsilon_\theta(\mathbf{x}_t, t) - w \cdot \sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log p_\theta(c|\mathbf{x}_t), \quad (9)$$

where  $\epsilon_\theta(\mathbf{x}_t, t)$  is the corresponding noise estimator for the unconditional denoising diffusion process  $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$ . As shown in Fig. 2, classifier guidance’s behaviors are dominated by the characteristics of the classifier.

It is worth noticing that classifier guidance with larger and larger guidance scales continuously distorts the straight-like denoising diffusion trajectories to *push them away from the classifier’s decision boundary*. In other words, the goal of *conditional* generation via classifier guidance is achieved by avoiding those areas in which the classifier is uncertain. This explains why a large guidance scale



**Figure 3. Classifier-free guidance distorts denoising diffusion trajectories.** We apply denoising diffusion models with classifier-free guidance on a 1D dataset composed of data from  $\mathcal{N}(\pm 1.0, 0.05)$ . The denoising diffusion process for all plots starts from the bottom to the top. We use the same trained model as well as the same initial noise for all plots. The trajectory differences are solely caused by different guidance scales. Similar to Fig. 2, classifier-free guidance with large guidance scale distorts the trajectories for vanilla conditional models (the first plot) to push them away from the dataset decision boundary, *i.e.*, the origin. Different scales in Fig. 2 and this figure arise from classifier guidance and classifier-free guidance’s differing sensitivities. Here, scale=2.5 distorts trajectories significantly, while Fig. 2’s scale=4 causes minor changes. We hypothesize that classifier-free guidance’s greater sensitivity stems from its training with conditioning dropout.

can produce high-fidelity images compared to results generated with a low guidance scale, *e.g.*, Fig. 3 in [16]. The reason is that a low guidance scale is not strong enough to move the diffusion trajectories away from areas on the data distribution manifold where different conditional information intersect. Due to the entanglement, these areas naturally form the decision boundary for a *well-trained* classifier. With a large guidance scale and a well-trained classifier, classifier guidance can completely avoid ambiguous areas on the image manifold and generate results *unambiguously* aligned with the conditional information.

The obvious next question: can this reasoning for classifier guidance be generalized to classifier-free guidance?

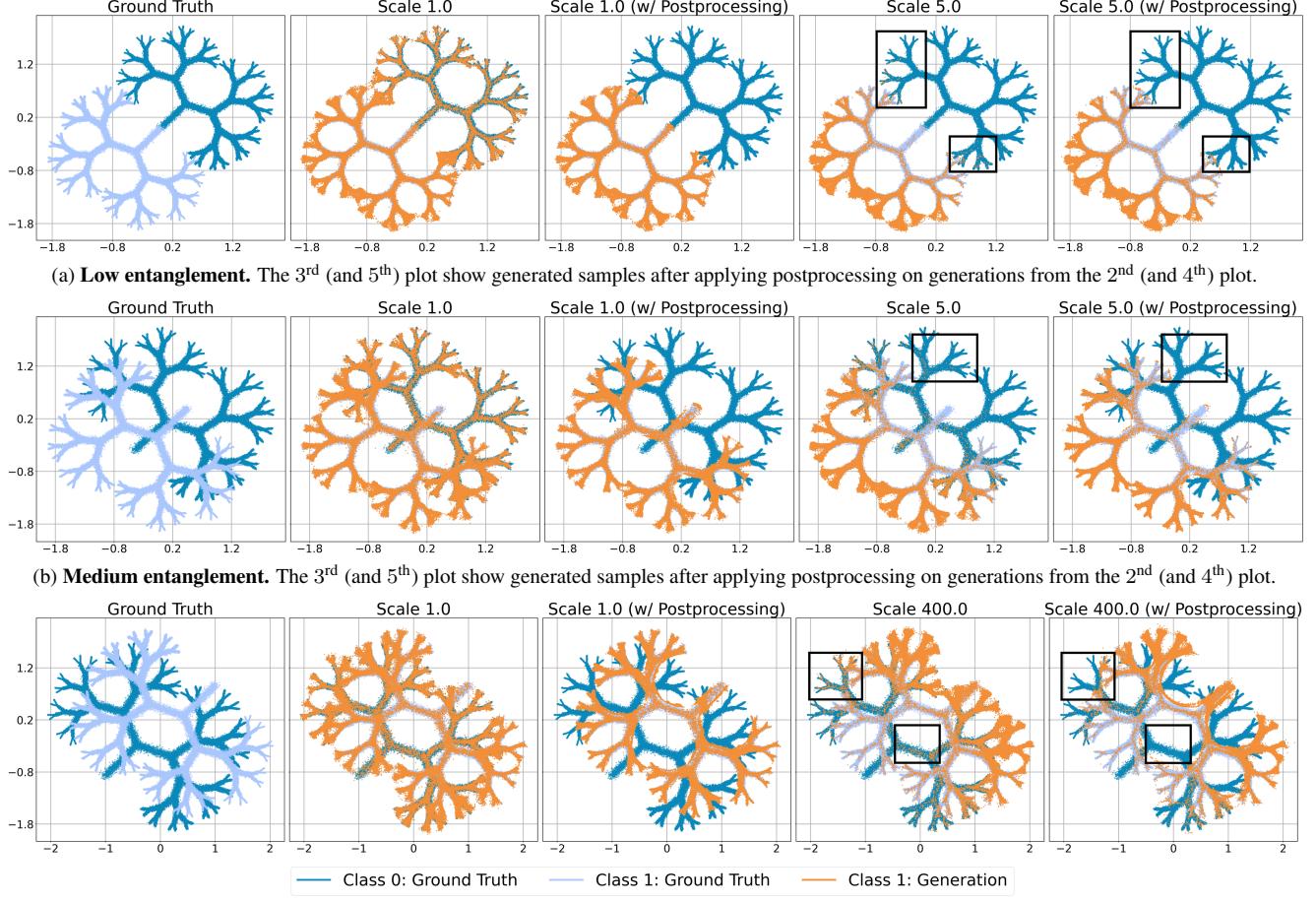
### 3.3. Classifier-Free Guidance Revisited

Classifier-free guidance was introduced to eliminate the reliance on a separate classifier [24]. Intuitively, with Bayes rule, we have  $p(c|\mathbf{x}_t) = p(c)p(\mathbf{x}_t|c)/p(\mathbf{x}_t)$ . Consequently,  $\nabla_{\mathbf{x}_t} \log p_\theta(c|\mathbf{x}_t)$  can be decomposed as  $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|c) - \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$ , where  $p(c)$  disappears as it is independent of  $x_t$ . When substituting this into Eq. (9) and considering Eq. (5), we have  $\tilde{\epsilon}_\theta(\mathbf{x}_t, t, c) =$

$$\epsilon_\theta(\mathbf{x}_t, t) + w \cdot (\epsilon_\theta(\mathbf{x}_t, t, c) - \epsilon_\theta(\mathbf{x}_t, t)). \quad (10)$$

The effect of classifier guidance can be achieved by training two noise estimators for both conditional ( $\epsilon_\theta(\mathbf{x}_t, t, c)$ ) and unconditional ( $\epsilon_\theta(\mathbf{x}_t, t)$ ) denoising diffusion processes respectively. In practice, Ho and Salimans [24] propose to only train one conditional denoising diffusion model but randomly drop out the conditioning information  $c$  during training to mimic the unconditional process.

One may notice that Eq. (6), *i.e.*, Eq. (2) in [16], *lays the foundation for classifier-free guidance* [24]. We want to know whether this connection can be used to show that



(c) **High entanglement.** The 3<sup>rd</sup> (and 5<sup>th</sup>) plot show generated samples after applying postprocessing on generations from the 2<sup>nd</sup> (and 4<sup>th</sup>) plot.

**Figure 4. Classifier guidance with flow-matching based postprocessing (Sec. 3.4) on 2D fractal data.** To demonstrate that the proposed postprocessing benefits sample generation across various levels of entanglements, we intentionally train a *linear* classifier which struggles to provide a correct signal  $p_\theta(c|x_i)$  in Eq. (6) as we increase the entanglement of both classes from Fig. 4a to 4c. After training, all three classifiers’ decision boundaries roughly align with the diagonal from top-left to bottom-right. For each level of entanglement, we generate two sets of samples, one with guidance scale  $w = 1$  and the other with a large scale ( $w = 5$  or  $400$ ) that pushes generations away from the classifier’s decision boundary. Then for each set, we train a corresponding rectified flow. The postprocessing step always improves the generated samples to match the real data, especially on boundaries between the two classes. For scenarios where scale equals  $1$ , before postprocessing, a large portion of the generations fall into the incorrect category as the classifier signal is not strong enough. With the proposed postprocessing step, we observe correct generations aligned with the conditioning information. For large scales, though the generations are generally correct as the signal from the classifier starts to dominate the generation process, there are still outliers as highlighted. The proposed postprocessing significantly corrects these low-quality generations while not altering already-high-quality generations. For a clear visualization, we only display the generation for one class. Please refer to the appendix for the other class.

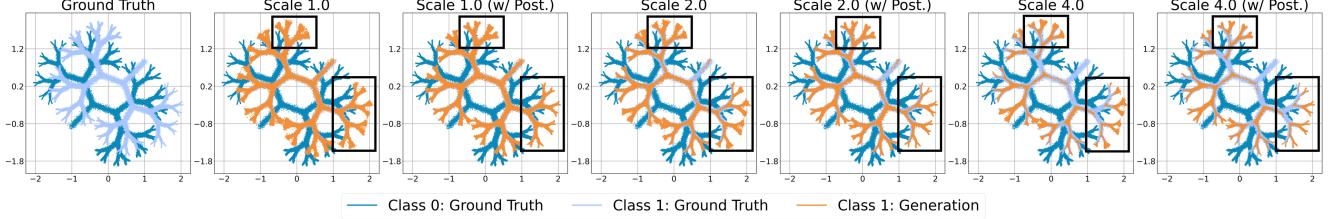
classifier-free guidance inherits the characteristics of classifier guidance. Namely, does classifier-free guidance also try to push the diffusion trajectories away from the data’s decision boundary? Note, classifier-free guidance does not directly involve any explicit classifier. However, based on our discussion in Sec. 3.2, a well-trained classifier’s decision boundary naturally aligns with the data’s decision boundary. Experiments on 1D synthetic datasets provide an affirmative answer as shown in Fig. 3.

Based on the understanding of the connection between classifier guidance and classifier-free guidance, we hypothesize that an approach which improves the generation qual-

ity of classifier guidance can also increase the fidelity of classifier-free guidance. We thus propose a generic framework from a classifier-centric perspective in the following. It focuses on improving low-quality generations around the decision boundary.

### 3.4. Generation Postprocessing with Flow Matching

Let  $\mathcal{X}_{\text{real}}$  refer to a set of samples from the real data distribution while  $\mathcal{X}$  denotes a set of generated samples from a generative model. Note,  $\mathcal{X}$  is agnostic to the specific choice of generation strategy, *e.g.*, it does not matter whether  $\mathcal{X}$  was produced with classifier guidance or classifier-free guid-



**Figure 5. Classifier-free guidance with flow-matching based postprocessing (Sec. 3.4) on 2D fractal data.** The level of entanglement is the same as that in Fig. 4c. Each plot with ‘Post.’ in the title displays generations after applying postprocessing on generations from the corresponding previous plot. We observe that the postprocessing step continuously improves the fidelity of the generations via moving outlier samples back to the real data distribution as the leaf branches become much sharper, regardless of the guidance scale we use.

ance. Our goal is to move the distribution underlying  $\mathcal{X}$  closer to the distribution represented by  $\mathcal{X}_{\text{real}}$ . However, since  $\mathcal{X}$  is certainly not an isotropic Gaussian distribution, denoising diffusion models are not suitable. Instead, we can however train a rectified flow [40, 41]  $v_\theta$  that transfers data from  $\mathcal{X}$  to  $\mathcal{X}_{\text{real}}$ . Intuitively, numerically integrating the velocity vector field  $v_\theta$  from time  $t = 0$  to time  $t = 1$  to solve an ordinary differential equation (ODE) adjusts the samples from  $\mathcal{X}$  towards  $\mathcal{X}_{\text{real}}$ . To learn the velocity vector field  $v_\theta$  we propose to optimize:

$$\min_{v_\theta} \int_0^1 \mathbb{E}_{\mathcal{X}} [\|(\hat{\mathbf{x}} - \text{NN}(\hat{\mathbf{x}}, \mathcal{X}_{\text{real}})) - v_\theta(\hat{\mathbf{x}}_t, c, t)\|^2] dt, \quad (11)$$

$$\text{where } \hat{\mathbf{x}} \sim \mathcal{X}, \hat{\mathbf{x}}_t = (1-t) \cdot \hat{\mathbf{x}} + t \cdot \text{NN}(\hat{\mathbf{x}}, \mathcal{X}_{\text{real}}). \quad (12)$$

Here  $\text{NN}(\hat{\mathbf{x}}, \mathcal{X}_{\text{real}})$  represent the nearest neighbor sample for  $\hat{\mathbf{x}}$  in the real data set  $\mathcal{X}_{\text{real}}$ .

We emphasize the use of NN in Eq. (11), which differs from classic rectified flow formulations. Importantly, the use of NN automatically balances between 1) already-high-quality generations; and 2) low-quality generations. If  $\hat{\mathbf{x}}$  is already a high-fidelity generation, *i.e.*, close to  $\mathcal{X}_{\text{real}}$ ,  $\hat{\mathbf{x}} - \text{NN}(\hat{\mathbf{x}}, \mathcal{X}_{\text{real}})$  will be extremely small, providing a negligible learning signal. In contrast, generations that are far away from the real data, *e.g.*, those that are misguided by incorrect signals from the classifier, will be dragged toward the real distribution. Based on our study in Sec. 3.2 and Sec. 3.3, the training will focus on generations around decision boundaries as they exhibit strong learning signals. In practice, inspired by [56], we do not always use the nearest neighbor  $\text{NN}(\hat{\mathbf{x}}, \mathcal{X}_{\text{real}})$ . Instead, we first find top- $k$  nearest neighbors and randomly select one from the top- $k$  as the target during each training iteration. Injecting this randomness provides more opportunities to avoid local optima.

After training the postprocessing flow, conditional generation involves two steps: 1) sampling from the original denoising diffusion model  $p_\theta(\mathbf{x}_0|c)$  in Eq. (3) to obtain a sample  $\hat{\mathbf{x}}_0$ ; and 2) running an ODE solver over the time interval  $[0, 1]$  to solve  $d\mathbf{z}_t/dt = v_\theta(\mathbf{z}_t, c, t)$  numerically while starting from  $\mathbf{z}_0 = \hat{\mathbf{x}}_0$ . The ODE solver output  $\mathbf{z}_1$  will be our final generation. Here, we use  $\hat{\mathbf{x}}_0$  to highlight the output of the base generative model and  $\mathbf{z}_t$  to emphasize that postprocessing is based on a separate flow matching procedure.

**Table 1. Nearest neighbor distance between generations and ground truth for Fig. 5.** We report the average nearest neighbor (NN) distance ( $\times 10^{-5}$ ) for 20k generations from the same initial noise in the format of  $A/B/C$ , *i.e.*, NN distance before postprocessing (A) / postprocessing with **nearest** (B) / postprocessing with random sampling from 20 candidates (C). Our approach (B or C) moves the generations closer to the real data. B vs. C verifies that random sampling is better than always choosing the nearest.

Guidance Scale	$w = 1$	$w = 2$	$w = 4$
Class 0	5.95 / 3.11 / 1.57	5.31 / 1.70 / 1.01	11.7 / 2.05 / 1.12
Class 1	12.4 / 3.25 / 1.70	6.09 / 1.88 / 1.00	6.51 / 1.41 / 0.79

The proposed postprocessing is related to autoguidance [30], which guides the model training with a bad version of itself. Autoguidance moves samples in the direction given by the difference between an inferior version and the current model. The direction *approximates* the desired change from the currently learned distribution to the real data distribution. The approximation is noisy, needing careful search ([30]’s Sec. 4 “mismatched degradations”). In contrast, we provide another perspective: we acknowledge the issue in generations from a pre-trained denoising diffusion model with a *specific* guidance scale. We then use a flow matching model to mitigate the issues in the generative model’s learned distribution, *i.e.*, we propose a form of *cascading* [59]. Such a framework alleviates the need to search for an optimal guidance scale to produce results aligned with the conditional information (or maybe there does not even exist such a scale due to the limited capabilities of a pre-trained model) since it can boost performance with arbitrary guidance scales as we will show.

## 4. Experiments

Please see Appendix C for all implementation details.

### 4.1. 2D Fractal

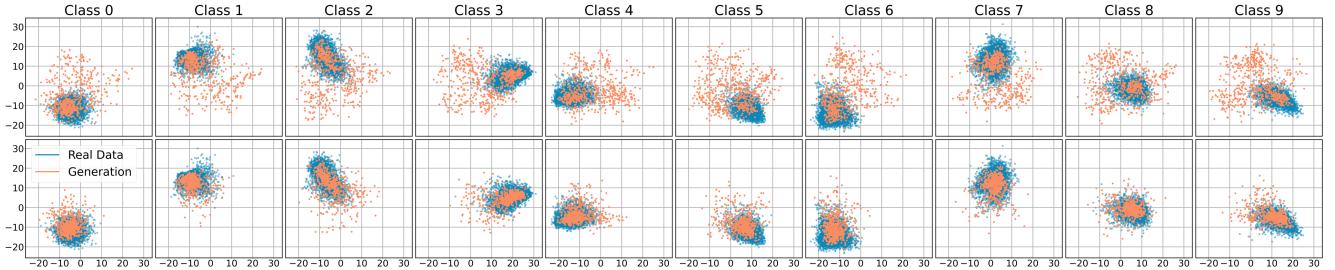
We first verify our analysis and the flow matching based postprocessing step on a synthetic 2D fractal dataset consisting of two classes. The dataset is represented by a mixture of Gaussians, similar to the dataset used by Karras et al. [30]. Please refer to the appendix for details. As displayed in “Ground Truth” plots in Fig. 4, *this synthetic dataset provides an easy way to control the level of entanglement*



(a) Generations before postprocessing step.



(b) Generations after postprocessing step.



(c) Postprocessing moves low-fidelity generations closer to the real data distribution. We train a classifier (not the one for classifier guidance) that achieves almost perfect accuracy on the validation split. A principal component analysis (PCA) is built on MNIST training data's features extracted from the classifier. With the fitted PCA, we transform 6k (600 for each digit) generations accordingly and visualize the first two components. The top and bottom rows correspond to Fig. 6a and 6b respectively. The postprocessing moves the learned distribution (orange clusters) closer to the real one (blue clusters).

Figure 6. Classifier guidance (scale 1.0) with flow-matching based postprocessing (Sec. 3.4) on MNIST. Fig. 6a and 6b share the same initial noises for corresponding cells and conditioning information from top to bottom row is the digit 0 to 9. The flow matching based postprocessing clearly improves the alignment between generations and conditioning, which is verified by Fig. 6c as well.



(a) Generations before postprocessing step.



(b) Generations after postprocessing step.

Figure 7. Classifier-free guidance (scale 10.0) with flow-matching based postprocessing (Sec. 3.4) on MNIST. As expected, a large guidance scale deteriorates generation quality and digits are blurry (Fig. 7a). Our postprocessing recovers the fidelity as shown in Fig. 7b.

among data from different classes. Our analysis in Sec. 3 reveals that both classifier guidance and classifier-free guidance achieve conditional generations by pushing the samples away from the decision boundaries. These boundaries are complex if the data is highly entangled. Consequently, this complexity will affect the quality of generations, providing a great testbed to study our proposed approach.

We qualitatively illustrate the effects of our proposed approach for classifier guidance and classifier-free guidance in Fig. 4 and Fig. 5 respectively. We choose top-20 nearest neighbors when training the rectified flow in all experiments if not specified otherwise. As can be seen clearly, the proposed postprocessing step greatly enhances the gen-

eration quality no matter what kind of guidance or guidance scale we use. Specifically, before postprocessing, some generated samples do not align with the conditional information, *i.e.*, they either fall into areas for the other class or are outliers far from the real data. Since these issues usually occur around the class decision boundary, the improvement of generation verifies our hypothesis in Sec. 3.4: postprocessing can improve low-quality generations around the boundary. Quantitatively, Tab. 1 verifies the effectiveness of our approach for Fig. 5.

Additionally, according to Eq. (10), when using a scale of 1 for classifier-free guidance, we essentially sample from a pure conditional model, *i.e.*,  $p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1}, c)$  on Eq. (6) left

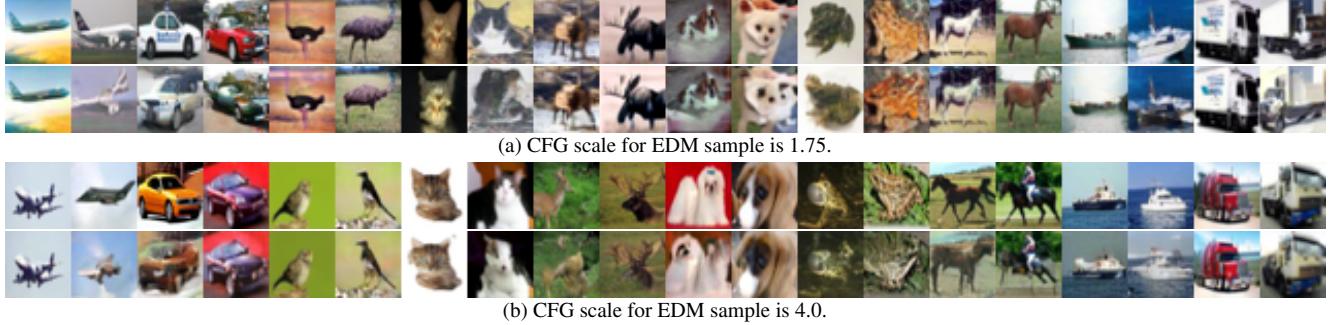


Figure 8. Tab. 3 results. Top and bottom for each subplot displays generations before and after postprocessing respectively.

**Table 2. One-for-All-Scales postprocessing model for classifier-free guidance on CIFAR-10 (base model as flow matching).** We report conditional FID evaluation computed on 50k generations (lower is better). We train a generalized flow matching based post-processing model on data sampled with guidance scales 1.00, 2.00, and 3.00 and apply it to samples with scales that are not seen during training. We ablate the distance space where nearest neighbor NN is computed, namely 1) the raw RGB space (Pixel); 2) the feature for the `CLS` token output by DINOv2 [44] (DINOv2 `CLS`); and 3) the average of the patch features (DINOv2 Patch). We also ablate the number of top- $k$  nearest neighbors (NN) used during training of the postprocessing model. Please refer to Sec. 4.4 for details. As can be seen clearly, DINOv2 patch feature space yields the best performance, and using more nearest neighbors is beneficial. Our proposed postprocessing step consistently improves the FID score for most guidance scales. Postprocessing is abbreviated as “Post.” and **best FID** is highlighted.

Post.	NN Space	top- $k$ for NN	CFG Scale for Samples Before Postprocessing						
			1.25	1.50	1.75	2.25	2.50	2.75	
1	✗	–	–	6.151	13.10	21.14	35.77	41.58	46.37
2	✓	Pixel	20	13.12	13.49	15.70	22.55	25.96	28.95
3	✓	DINOv2 <code>CLS</code>	20	9.456	9.851	12.17	19.37	22.97	26.48
4	✓	DINOv2 Patch	20	6.961	8.545	11.22	17.27	20.19	23.32
5	✓	DINOv2 Patch	40	6.252	8.035	11.20	17.48	20.10	22.69

**Table 3. One-for-All-Scales postprocessing model for classifier-free guidance on CIFAR-10 (base model as EDM [29]).** The setup for training the postprocessing model (post.) follows Tab. 2 Row 5. Results confirm: 1) CIFAR-10 has low entanglement as larger guidance causes worse FID; 2) postprocessing boosts diversity for large CFG scales and stabilizes performances across various scales. See visualizations in Fig. 8.

Post.	CFG Scale for Samples Before Post.							
	1.75	2.25	2.50	2.75	3.25	3.5	3.75	4.0
✗	5.096	8.016	9.402	10.75	13.23	14.31	15.37	16.34
✓	5.718	5.821	5.936	6.176	6.675	6.957	7.202	7.490

side. The comparison between two plots of “Scale 1.0” in Fig. 4c and Fig. 5 corroborates our analysis in Sec. 3.2 that the two sides of Eq. (6) are generally not equal.

## 4.2. MNIST

Our proposed postprocessing step improves the fidelity of generations for both classifier guidance and classifier-free guidance on real-world MNIST [38] data as shown in Fig. 6 and Fig. 7. Our denoising diffusion and rectified flow models are based on a UNet [51] similar to the one used by Dhariwal and Nichol [16].

## 4.3. One-for-All-Scales Postprocessing Model

Above, we show results using a dedicated flow matching model which was trained on a set of samples  $\mathcal{X}$  generated with a *specific* guidance scale. A natural question remains: is it possible to develop a *generalized* postprocessing model that can be trained only once and then applied to all samples regardless of their guidance scale? We provide an affirmative answer in the appendix Sec. A.

## 4.4. CIFAR-10

We further verify our postprocessing on image synthesis via Fréchet Inception Distance (FID) [23] on CIFAR-10 [37]. As mentioned in Sec. 3.4, our postprocessing method only requires a set of samples and real data and is agnostic to the base generation strategies. Thus, we conduct experiments on two base generative models:

**Base model as flow matching.** We train the base model following one of the state-of-the-art flow-matching-based generative models [56] on CIFAR-10. Tab. 2 provides the quantitative results. We find that 1) CIFAR-10 has low entanglement, and 2) our postprocessing increases the quality of generations for high guidance scales. See Appendix B for qualitative results and a detailed analysis.

**Base model as EDM [29].** EDM is one of the state-of-the-art diffusion-based generative models on CIFAR-10. However, the pre-trained EDM model lacks conditioning dropout, making it incompatible with CFG. We re-train a CFG-compatible EDM, verifying correctness with FID 1.850 (ours with CFG scale 1.0) vs. 1.849 (pre-trained). Postprocessing stabilizes the generation qualities across various guidance scales as shown in Tab. 3 and Fig. 8.

## 5. Conclusion

We carry out an empirical study aiming to understand classifier-free guidance from a classifier-centric perspective. Our analysis reveals that both classifier-free guidance and classifier guidance push the denoising diffusion process away from the data’s decision boundaries, where low-quality generations usually occur. To improve low-quality generations around the decision boundary, we propose a flow matching based postprocessing step, and verify its effectiveness on various datasets.

**Acknowledgements:** Work supported in part by NSF grants 2008387, 2045586, 2106825, MRI 1725729, and NIFA award 2020-67021-32799.

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 Technical Report. *ArXiv*, 2023. [1](#) [2](#)
- [2] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A Learning Algorithm for Boltzmann Machines. *Cognitive Science*, 1985. [1](#) [2](#)
- [3] Jyoti Aneja, Alexander G. Schwing, Jan Kautz, and Arash Vahdat. A Contrastive Learning Approach for Training Variational Autoencoder Priors. In *NeurIPS*, 2021. [2](#)
- [4] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian J. Goodfellow, and Augustus Odena. Discriminator Rejection Sampling. In *ICLR*, 2019. [2](#)
- [5] M. Bauer and Andriy Mnih. Resampled Priors for Variational Autoencoders. In *AISTATS*, 2019. [2](#)
- [6] Arwen Bradley and Preetum Nakkiran. Classifier-Free Guidance is a Predictor-Corrector. *ArXiv*, 2024. [1](#) [2](#)
- [7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *ArXiv*, 2018. [1](#) [2](#)
- [8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *NeurIPS*, 2020. [2](#)
- [9] George Casella, Christian P Robert, and Martin T Wells. Generalized Accept-Reject Sampling Schemes. *Lecture notes-monograph series*, 2004. [2](#)
- [10] Tong Che, Ruixiang Zhang, Jascha Narain Sohl-Dickstein, H. Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling. In *NeurIPS*, 2020. [2](#)
- [11] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating Gradients for Waveform Generation. In *ICLR*, 2021. [2](#)
- [12] Muthu Chidambaram, Khashayar Gatmiry, Sitan Chen, Holden Lee, and Jianfeng Lu. What Does Guidance Do? A Fine-Grained Analysis in a Simple Setting. In *NeurIPS*, 2024. [1](#) [2](#)
- [13] Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. CFG++: Manifold-Constrained Classifier Free Guidance for Diffusion Models. *ArXiv*, 2024. [2](#)
- [14] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision Transformers Need Registers. *ArXiv*, 2023. [11](#)
- [15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. [2](#)
- [16] Prafulla Dhariwal and Alexander Nichol. Diffusion Models Beat GANs on Image Synthesis. In *NeurIPS*, 2021. [1](#) [2](#) [3](#) [4](#) [8](#) [16](#)
- [17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. [11](#)
- [18] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvassy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The FAISS Library. *ArXiv*, 2024. [15](#)
- [19] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 Herd of Models. *ArXiv*, 2024. [1](#) [2](#)
- [20] Federico A. Galatolo, Mario Giovanni C. A. Cimino, and Gigliola Vaglini. Generating Images from Caption and Vice Versa via CLIP-Guided Generative Latent Space Search. In *International Conference on Image Processing and Vision Engineering*, 2021. [2](#)
- [21] Aditya Grover, Ramki Gummadi, Miguel Lázaro-Gredilla, Dale Schuurmans, and Stefano Ermon. Variational Rejection Sampling. In *AISTATS*, 2018. [2](#)
- [22] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *ICML*, 2020. [15](#)
- [23] 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. In *NeurIPS*, 2017. [8](#) [11](#)
- [24] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS Workshop*, 2021. [1](#) [2](#) [4](#)
- [25] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020. [1](#) [3](#)
- [26] Susung Hong, Gyuseong Lee, Wooseok Jang, and Sung Wook Kim. Improving Sample Quality of Diffusion Models Using Self-Attention Guidance. In *ICCV*, 2023. [1](#)
- [27] Martin Jankowiak and Du Phan. Reparameterized variational rejection sampling. In *AISTATS*, 2023. [2](#)
- [28] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *CVPR*, 2018. [1](#) [2](#)
- [29] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *NeurIPS*, 2022. [8](#)
- [30] Tero Karras, Miika Aittala, Tuomas Kynkänniemi, Jaakkko Lehtinen, Timo Aila, and Samuli Laine. Guiding a Diffusion Model with a Bad Version of Itself. In *NeurIPS*, 2024. [6](#) [15](#)
- [31] Dongjun Kim, Yeongmin Kim, Wanmo Kang, and Il-Chul Moon. Refining Generative Process with Discriminator Guidance in Score-based Diffusion Models. In *ICML*, 2022. [1](#) [2](#)

- [32] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ArXiv*, 2014. 16
- [33] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*, 2018. 2
- [34] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014. 2
- [35] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. In *NeurIPS*, 2021. 1
- [36] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *ICLR*, 2021. 2
- [37] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009. 8, 11
- [38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998. 8
- [39] Shanchuan Lin and Xiao Yang. Diffusion Model with Perceptual Loss. *ArXiv*, 2024. 2
- [40] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *ICLR*, 2023. 2, 6
- [41] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *ICLR*, 2023. 2, 6, 11
- [42] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2017. 15, 16
- [43] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The journal of Chemical Physics*, 1953. 2
- [44] Maxime Oquab, Timothée Darzet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINov2: Learning Robust Visual Features without Supervision. *ArXiv*, 2023. 8, 11, 14
- [45] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam M. Shazeer, Alexander Ku, and Dustin Tran. Image Transformer. In *ICML*, 2018. 2
- [46] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. In *ICCV*, 2021. 2
- [47] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*, 2023. 1, 2
- [48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021. 2
- [49] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. In *NeurIPS*, 2019. 2
- [50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *CVPR*, 2022. 1, 2
- [51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015. 8
- [52] Seyedmorteza Sadat, Jakob Buhmann, Derek Bradley, Otmar Hilliges, and Romann M. Weber. Cads: Unleashing the diversity of diffusion models through condition-annealed sampling. In *ICLR*, 2024. 2
- [53] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *NeurIPS*, 2019. 1, 3
- [54] Yang Song, Jascha Narain Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*, 2021. 2, 3
- [55] Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. SOAR: Improved Indexing for Approximate Nearest Neighbor Search. In *NeurIPS*, 2023. 15
- [56] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and Generalizing Flow-Based Generative Models With Minibatch Optimal Transport. *TMLR*, 2024. 6, 8, 11
- [57] Ryan D. Turner, Jane Hung, Yunus Saatci, and Jason Yosinski. Metropolis-Hastings Generative Adversarial Networks. In *ICML*, 2018. 2
- [58] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 2011. 1, 3
- [59] Paul Viola and Michael Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *CVPR*, 2001. 6
- [60] Yuchen Wu, Minshuo Chen, Zihao Li, Mengdi Wang, and Yuting Wei. Theoretical Insights for Diffusion Guidance: A Case Study for Gaussian Mixture Models. *ArXiv*, 2024. 2
- [61] Mengfei Xia, Nan Xue, Yujun Shen, Ran Yi, Tieliang Gong, and Yong-Jin Liu. Rectified Diffusion Guidance for Conditional Generation. *ArXiv*, 2024. 1, 2

# Studying Classifier(-Free) Guidance From a Classifier-Centric Perspective

## Supplementary Material

This supplementary material is structured as follows:

1. Sec. A discusses a generalized postprocessing model and its evaluation on 2D fractal and MNIST data;
2. Sec. B provides details for our proposed approach on the CIFAR-10 dataset;
3. Sec. C describes implementation details;
4. Sec. D discusses limitations;
5. Sec. E provides more visualizations.

### A. One-for-All-Scales Postprocessing Model

In the main paper, specifically in Fig. 4, 5, 6, and 7, we show results using a dedicated flow matching model which was trained on a set of samples  $\mathcal{X}$  generated with a *specific* guidance scale. A natural next question: is it possible to develop a *generalized* postprocessing model that can be trained only once and then applied to all samples regardless of their guidance scale?

We conduct experiments and answer this question in the affirmative. Specifically, we show results for 2D fractal data in Fig. S1, while Fig. S2 and Fig. S3 provide results for MNIST. We emphasize that the amount of training data is identical, *i.e.*, the generalized model is exposed to the same amount of training data as the individually-trained model. From the visualized results we observe that even in this fair setup, the generalized model performs on-par or even better than individually-trained models. We hypothesize that this is because generations with different scales essentially share a similar underlying distribution, as classifier guidance and classifier-free guidance both push the denoising trajectories away from the decision boundaries. With a combined set of samples, the model is exposed to a diverse but coherent dataset, facilitating model generalization.

### B. Experiments on CIFAR-10

#### B.1. Base Model as Flow Matching

As mentioned in Sec. 4.4, we verify our proposed postprocessing approach on image synthesis using CIFAR-10 [37]. Here, we study how our proposed postprocessing step works in conjunction with a base generative model that is based on flow matching, *i.e.*, both our base and postprocessing generative model are flow-matching-based models. For this, we train a rectified flow [41] based conditional generative model on CIFAR-10. Following the procedure for a one-for-all-scales postprocessing model discussed in Appendix A, we compose a training set of 50k generations sampled with a guidance scale of 1.0, 2.0, and 3.0. Each scale contributes to one-third of the training data. With this

data, we train a generalized postprocessing model and evaluate it using Fréchet Inception Distance (FID) [23] computed with generations with unseen guidance scales. All generations are obtained using the adaptive solver Dopr5 following the suggestions of Tong et al. [56].

Recall that our model training requires to find the nearest neighbor from real data as detailed in Eq. (11). The space for nearest neighbor computation is hence an important design choice. Here we study two possible candidates: 1) directly using the raw RGB pixel space; 2) using the feature embedding space of recently developed pre-trained foundation models. For the latter, we employ DINOv2 [44] pre-trained with registers [14]. Due to the ViT structure [17] used in DINOv2, we can either use the information encoded in the CLS token to search for the nearest neighbor or we can take an average of the features corresponding to each patch token. The quantitative results are summarized in Tab. 2. We observe that the model trained with aggregated patch feature performs best (Row 4 vs. Row 2 and 3).

One interesting observation in Tab. 2 is that generations with a guidance scale of 1.25 before the postprocessing (Row 1, FID of 6.151) perform slightly better than those after postprocessing (Row 4, FID of 6.961). We hypothesize that this indicates that CIFAR-10 has low entanglement around the decision boundary for the underlying data distribution. Indeed, the ten classes, namely plane, car, bird, cat, deer, dog, frog, horse, ship, and truck are easily distinguishable. The guidance scale of 1.25 is small enough such that generations will not be pushed too far away from the decision boundaries, covering a reasonably large area on the real data manifold. The nearest neighbor search can shrink the diversity of the generations, making the learned distribution after the postprocessing less aligned with the real distribution. Consequently, the FID score after postprocessing gets penalized since FID essentially reflects the alignment between distributions. To verify, we train a new postprocessing model using the DINOv2 patch features but with more nearest neighbors (Sec. 3.4). The better-performing Row 5, especially for the scale of 1.25, *i.e.*, 6.252 in Row 5 vs. 6.961 in Row 4, confirms our hypothesis: with more nearest neighbors, we effectively reduce the level of diversity shrinkage, resulting in a better FID.

Note, except for the scale of 1.25, the performance comparison between Row 5 vs. 4 for other larger guidance scales is on par. This further corroborates our hypothesis: based on our classifier-centric analysis, for large scales, the generations are pushed away from the decision boundaries, which inherently harms the diversity of the generations. Therefore, more nearest neighbors will not change the level of

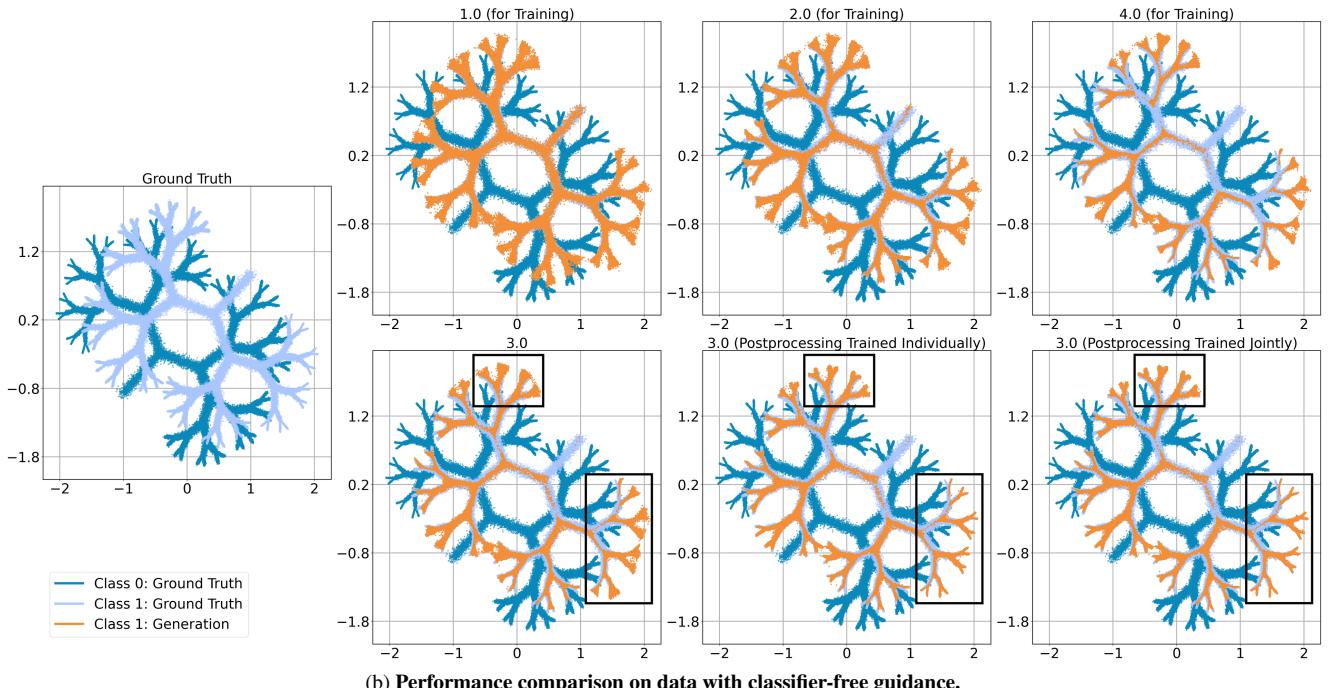
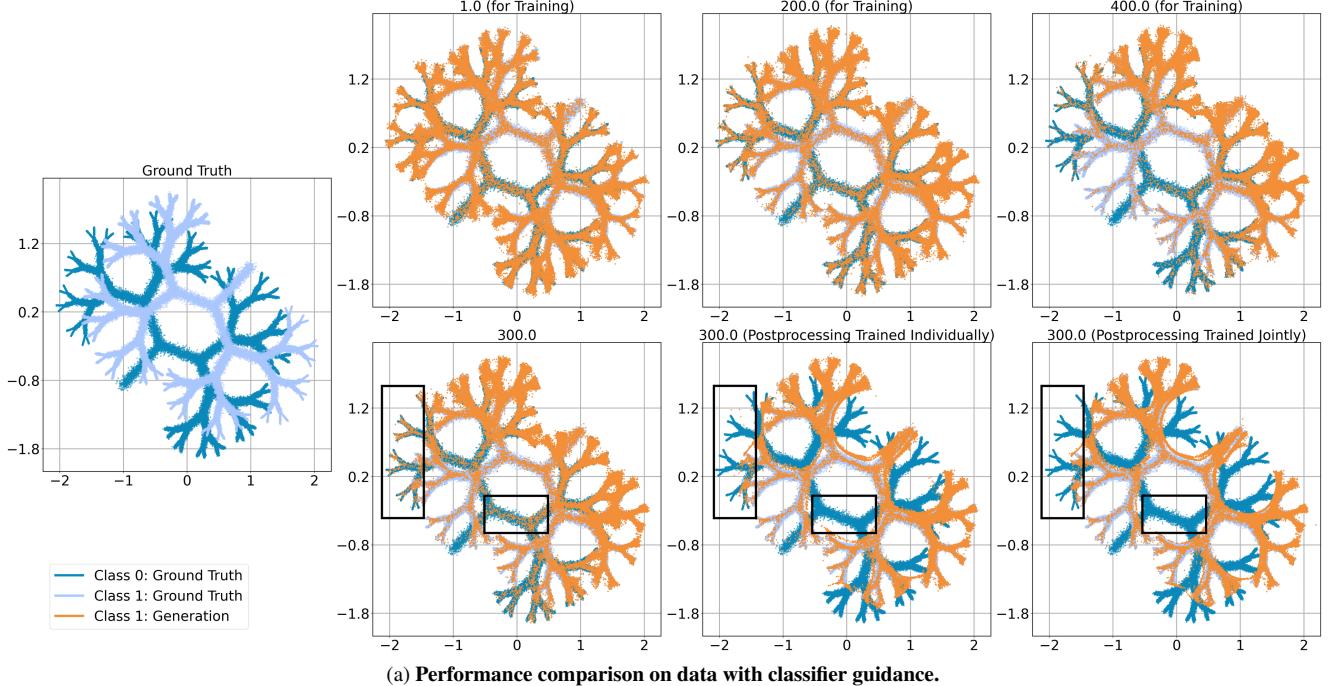


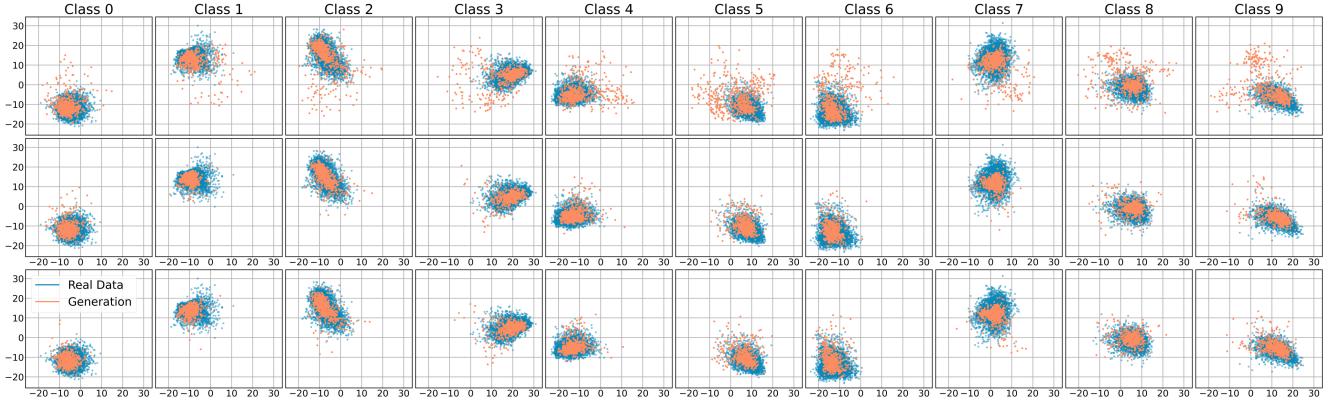
Figure S1. **One-for-All-Scales postprocessing model (Sec. 3.4) on 2D fractal data.** We train a *generalized* postprocessing model that can be applied to samples generated by various guidance scales. In Fig. S1a and S1b, the top row displays the training data set composed of samples generated with three different scales. Note that for clear visualization, we only show one class of data. The generalized model is trained on two classes of samples. The bottom row in Fig. S1a and S1b compare the performance between *individually-trained* and *generalized* postprocessing model. We want to emphasize that *we keep the amount of training data identical for both post-processing models such that the comparison is fair*. Concretely, for the individually-trained model, we generate 100k points with the specific guidance scale. For the generalized model, we generate 100k /  $K$  samples for each of the  $K$  different guidance scales. As can be seen clearly, the generalized postprocessing model performs well, demonstrating the potential of a One-for-All-Scales postprocessing model.



(a) Before postprocessing.

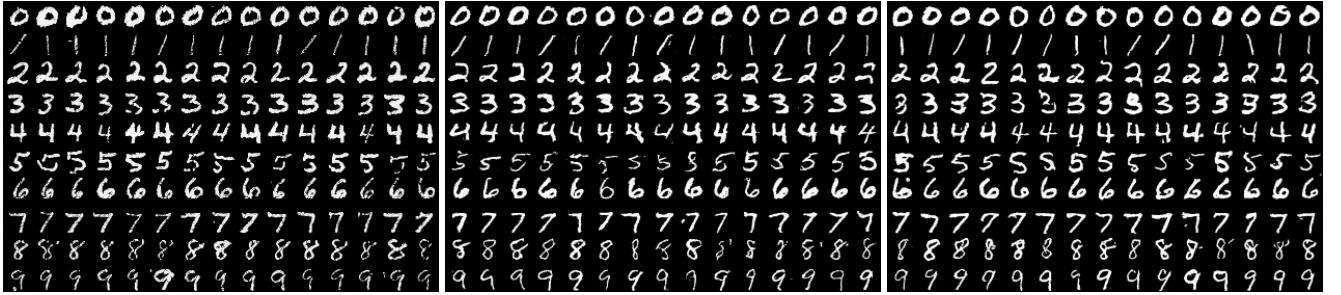
(b) Postprocessing; individually-trained model.

(c) Postprocessing; jointly-trained model.



(d) PCA comparison for generalized postprocessing model. We show PCA results with a procedure similar to that in Fig. 6c. The 1<sup>st</sup>, 2<sup>nd</sup>, and 3<sup>rd</sup> rows correspond to Fig. S2a, S2b, and S2c respectively. The postprocessing moves the learned distribution (orange clusters) closer to the real one (blue clusters). The generalized model performs on par with the model that is trained with generations from the specific guidance scale.

**Figure S2. One-for-All-Scales postprocessing model (Sec. 3.4) for classifier guidance on MNIST.** Here we show the performance comparison for different postprocessing models on generations with a scale of 3.0. Fig. S2a, S2b, and S2c share the same initial noises for corresponding cells and conditioning information from top to bottom row is the digit 0 to 9. We want to emphasize that *we keep the amount of training data identical for both post-processing models such that the comparison is fair*. Concretely, for the individually-trained model, we generate 6k images with the specific guidance scale of 3.0. For the generalized model, we generate 2k samples each with scales 1.0, 5.0, and 10.0 respectively. As can be seen clearly, the generalized model performs well even though it has not been exposed to samples generated with the guidance scale of 3.0 (also verified by Fig. S2d).



(a) Before postprocessing.

(b) Postprocessing; individually-trained model.

(c) Postprocessing; jointly-trained model.

**Figure S3. One-for-All-Scales postprocessing model (Sec. 3.4) for classifier-free guidance on MNIST.** Here we show the performance comparison for different postprocessing models on generations with a scale of 8.0. Fig. S3a, S3b, and S3c share the same initial noises for corresponding cells and conditioning information from top to bottom row is the digit 0 to 9. We want to emphasize that *we keep the amount of training data identical for both post-processing models such that the comparison is fair*. Concretely, for the individually-trained model, we generate 6k images with the specific guidance scale of 8.0. For the generalized model, we generate 2k samples each with scales 1.0, 5.0, and 10.0 respectively. Both individually-trained and generalized postprocessing models recover the fidelity.

diversity a lot, resulting in a similar FID.

Generation results are illustrated in Fig. S4. Importantly,

our postprocessing model indeed increases the quality of the generations on this real-world image synthesis task for most



**Figure S4. One-for-All-Scales postprocessing model (Sec. 3.4) for classifier-free guidance on CIFAR-10.** Here we show the qualitative results of the flow matching based postprocessing model trained with top-40 nearest neighbor computed in DINOv2 [44] patch feature space, which corresponds to Row 5 in Tab. 2. For each plot, the left and right figures share the same initial noises for corresponding cells and conditioning information from top to bottom row is the class ID 0 to 9, *i.e.*, plane, car, bird, cat, deer, dog, frog, horse, ship, and truck. Note, the training dataset for our postprocessing model does not contain samples from these guidance scales. Our proposed postprocessing indeed increases the diversity of the generations especially when the guidance is large, *e.g.*, those from a guidance scale of 2.5 or 2.75.

guidance scales.

## B.2. Control of Postprocessing Model

Postprocessing controllability can be obtained via a CFG-compatible rectified flow model, *i.e.*, dropping out condi-

tions during training. See Tab. S1 and Fig. S5 for quantitative and qualitative results.

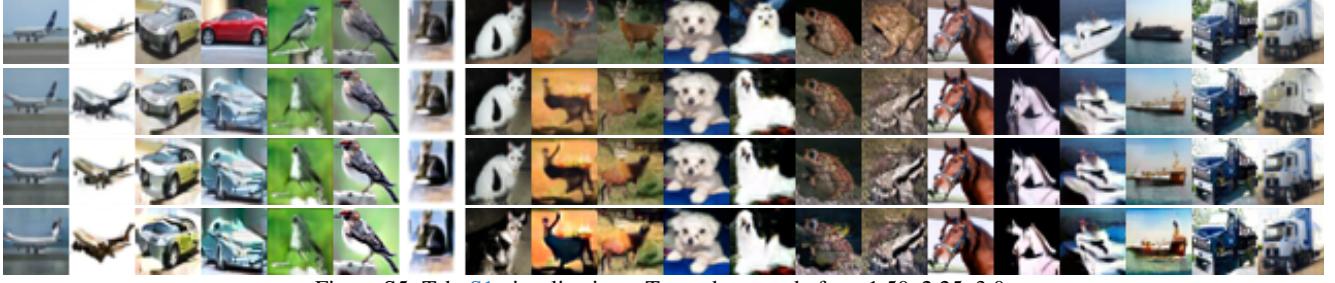


Figure S5. Tab. S1 visualizations. Top to bottom: before, 1.50, 2.25, 3.0.

Table S1. FID on CIFAR-10 for *CFG-compatible* postprocessing model. We run postprocessing with various CFG scales on samples generated from base model with CFG scale 2.75. Thus “Before” and “1.0” columns are same as Tab. 3 column “2.75”. See Fig. S5 for visualizations.

Before	1.0	1.25	1.50	1.75	2.0	2.25	2.50	2.75	3.0
10.75	6.176	6.842	7.394	7.856	8.361	8.918	9.545	10.29	11.27

## C. Implementation Details

### C.1. Top- $k$ Nearest Neighbor (NN) Search

There are several great tools [22, 55] for accelerating the nearest neighbor (NN) search required in Eq. (11). In this work, we use FAISS [18] for all our top- $k$  nearest neighbor computations.

### C.2. 2D Fractal Construction

The fractal data, *e.g.*, the data presented in Fig. 4 and Fig. 5, is synthesized similarly to the one used by Karras et al. [30] (Appendix C in [30]). The fractal data consists of 2D points sampled from a mixture of Gaussians, each situated to form branches.

Concretely, we first define how to conduct a branch split. For each class, the corresponding fractal is essentially a binary tree. The  $i$ -th branch is defined by the starting position  $\mathbf{s}_i \in \mathbb{R}^2$ , length  $l_i \in \mathbb{R}$ , and orientation  $o_i \in [0, 2\pi]$ . To obtain the child branch, *e.g.*, the  $k$ -th branch where  $k \in \{2i, 2i + 1\}$ , we follow the procedure:

$$\mathbf{s}_k = \mathbf{s}_i + l_i \cdot (\cos(o_i), \sin(o_i))^T, \quad (S1)$$

$$l_k = l_i \cdot (1 - 0.4 \cdot \xi), \quad \xi \sim U(0.5, 0.8), \quad \text{and} \quad (S2)$$

$$o_k = o_i + (-1)^{k+1} \cdot \pi \cdot \left( \frac{1}{2.8 \cdot \exp(\frac{\lfloor \log_2 k \rfloor}{4})} + \xi_1 \cdot \xi_2 \right),$$

$$\text{where } \xi_1 \sim \text{Bernoulli}(0.5), \quad \xi_2 \sim U(0, 0.05). \quad (S3)$$

We repeat the above branch split 6 times, *i.e.*, creating a binary tree with depth 6, resulting in  $2^0 + \dots + 2^6 = 127$  branches for each class. For both classes, we have  $\mathbf{s}_0 = \mathbf{0}$  and  $l_0 = 1.2$ . While class 0 uses  $o_0 = 0.25\pi$  and class 1 uses  $o_0 = 1.75\pi$ .

For each branch constructed above, we create 8 Gaussians whose means are uniformly positioned along the branch. For each class, for the  $i$ -th Gaussian from a total of  $127(\#\text{branches}) \times 8 = 1016$  Gaussians, we define

the covariance matrix by rotating the base covariance matrix  $\text{diag}(0.005 \cdot \exp(-\frac{i}{30}), 0.003 \cdot \exp(-\frac{i}{25}))$  using the corresponding branch’s orientation.

The final dataset is constructed by sampling points from the Gaussians defined above. Concretely, for each class, for the  $i$ -th branch, we sample  $\lfloor 1000 \times \exp(-\frac{i}{100}) \rfloor$  points from *each* of the 8 Gaussians on the branch.

### C.3. Models for 1D Gaussian

For 1D Gaussian data, both our denoising diffusion model for classifier guidance and classifier-free guidance are based on the structure illustrated in Fig. S6. We also construct the nonlinear classifier for the classifier guidance results shown in Fig. 2a using the structure illustrated in Fig. S6, but remove the class embedding block. The linear classifier used for the results shown in Fig. 2b is developed by feeding the concatenated data  $\mathbf{x}_t$  and diffusion step  $t$  to a linear layer.

We train both 1) linear and nonlinear classifiers; and 2) unconditional and conditional diffusion models using the optimizer AdamW [42] with a batch size of 4096 points and a learning rate of  $10^{-4}$ . The classifiers and diffusion models are trained for 50k and 100k steps respectively.

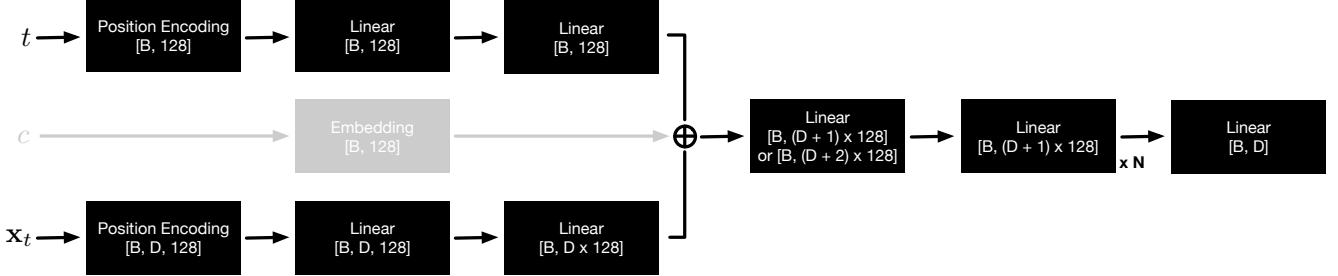
### C.4. Models for 2D Fractal

For the 2D fractal data, both our denoising diffusion model for classifier guidance and classifier-free guidance as well as the postprocessing model are based on the structure illustrated in Fig. S6. In all our 2D fractal experiments we use a *linear* classifier for classifier guidance, similar to the one discussed in Sec. C.3.

We train 1) the classifier; 2) unconditional and conditional diffusion models; and 3) the postprocessing model using the optimizer AdamW [42] with a batch size of 4096 points and a learning rate of  $10^{-4}$ . The training iterations for the classifier, the denoising diffusion models, and the postprocessing models are 30k, 100k, and 100k.

### C.5. Models for MNIST

In all our MNIST experiments we use a *linear* classifier for classifier guidance, similar to the one discussed in Sec. C.3. The input is a flattened image concatenated with the diffusion step. Our denoising diffusion models and the postprocessing model for classifier guidance and classifier-free



**Figure S6. MLP-based denoising diffusion model.** Here we provide details about the network structure for our MLP-based denoising diffusion model, which is used for experiments on 1D Gaussian data in Sec. 3 and on 2D fractal data in Sec. 4.1. In each block, the top row provides the layer’s name. The bottom row illustrates the tensor shape after the corresponding layer’s processing. Here,  $B$  and  $D$  stand for batch size and data channel respectively. For example,  $D$  equals 1 and 2 respectively for the data of 1D Gaussian and 2D fractal. We have the following position encoding function applied on *each channel* of the input data:  $\{\sin(2\pi \cdot u_0), \dots, \sin(2\pi \cdot u_{C-1}), \cos(2\pi \cdot u_0), \dots, \cos(2\pi \cdot u_{C-1})\}$ , where  $C = 128/2$  in our setup. We use  $u_i = x \cdot \exp(i \cdot \log_2(-\frac{10000}{C-1}))$ , where  $x$  is the value at the corresponding channel the position encoding is applied. Blocks with a grey background color are optional. Concretely, for classifier guidance, we train an *unconditional* model, which does not use the conditioning information  $c$ . In contrast, for classifier-free guidance, we enable the embedding layer to train a *conditional* model. Throughout our experiments, we repeat the second to last block four times, namely we have  $N = 4$ .

guidance are identical to the ones used by Dhariwal and Nichol [16].<sup>1</sup> We use the following configuration:

- `model_channels = 128;`
- `num_res_blocks = 2;`
- `attention_resolutions = (), i.e., empty;`
- `channel_mult = (1, 2).`

We train 1) the classifier; 2) unconditional and conditional diffusion models; and 3) the postprocessing models using the optimizer AdamW [42] with a batch size of 512 images and a learning rate of  $3 \cdot 10^{-4}$ . The training iterations for the classifier, the denoising diffusion models, and the postprocessing models are 100k, 2k, and 10k.

## C.6. Models for CIFAR-10

**In case of a flow matching base model**, our rectified flow models and the postprocessing model for classifier-free guidance are identical to the ones used by Dhariwal and Nichol [16].<sup>1</sup> We use the following configuration:

- `model_channels = 128;`
- `num_res_blocks = 2;`
- `attention_resolutions = (2,);`
- `channel_mult = (1, 2, 2, 2);`
- `num_heads = 4;`
- `num_head_channels = 64;`
- `dropout = 0.1.`

We train both 1) the conditional rectified flow model; and 2) the postprocessing model using the Adam [32] optimizer with a batch size of 128 images and a learning rate of  $2 \cdot 10^{-4}$  for 400k iterations. We store the exponential moving average of the model weights with a decay rate of 0.9999.

<sup>1</sup>[https://github.com/openai/guided-diffusion/blob/22e0df818350/guided\\_diffusion/unet.py#L396](https://github.com/openai/guided-diffusion/blob/22e0df818350/guided_diffusion/unet.py#L396)

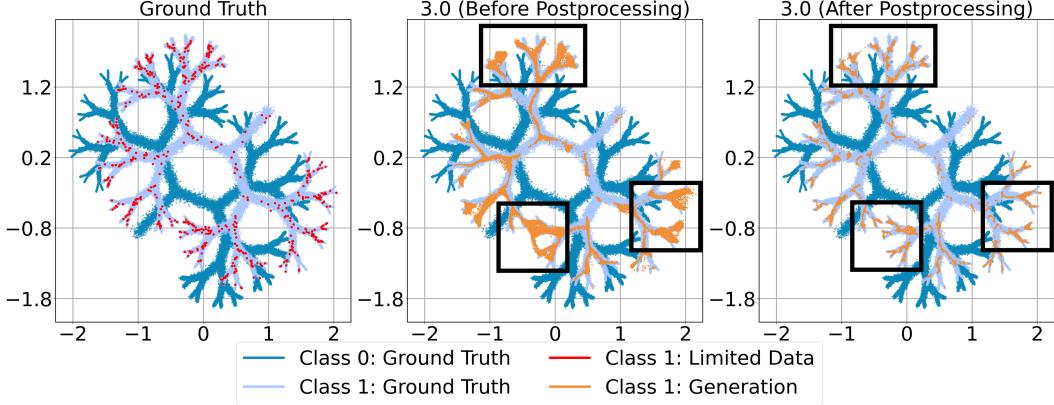
**In case of an EDM base model**, our postprocessing model for classifier-free guidance is also identical to the ones used by Dhariwal and Nichol [16].<sup>1</sup> We use the following configuration (with difference compared to previous setup highlighted):

- `model_channels = 128;`
- `num_res_blocks = 4;`
- `attention_resolutions = (2,);`
- `channel_mult = (1, 2, 2, 2);`
- `num_heads = 4;`
- `num_head_channels = 64;`
- `dropout = 0.1.`

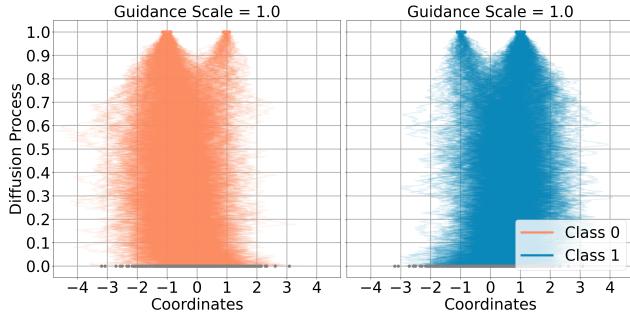
We train the postprocessing model using the Adam [32] optimizer with a batch size of 512 images and a learning rate of  $5 \cdot 10^{-4}$  for 400k iterations. We store the exponential moving average of the model weights with a decay rate of 0.9999. The primary goal of the highlighted changes is to maximize alignment between the capacities of our postprocessing model and the base EDM model while working within our limited computing resources. For example, increasing `num_res_blocks` expands our postprocessing model’s number of parameters from 34.10 M to 55.97 M, roughly matching the base EDM model’s 55.74 M parameters.

## D. Limitations

Regarding efficiency, since our postprocessing step runs another round of diffusion, inference time will be doubled when compared to the generation process without postprocessing. However, with more prevailing distillation techniques and faster samplers, we think the overhead can be largely mitigated.



**Figure S7. Postprocessing on limited data.** Given limited data (red points in 1st plot), postprocessing mitigates out-of-distribution generations (highlighted). We train a *generalized* postprocessing model on samples at CFG scales  $\{1.0, 2.0, 4.0\}$ , then apply it to samples at *unseen* scale 3.0.



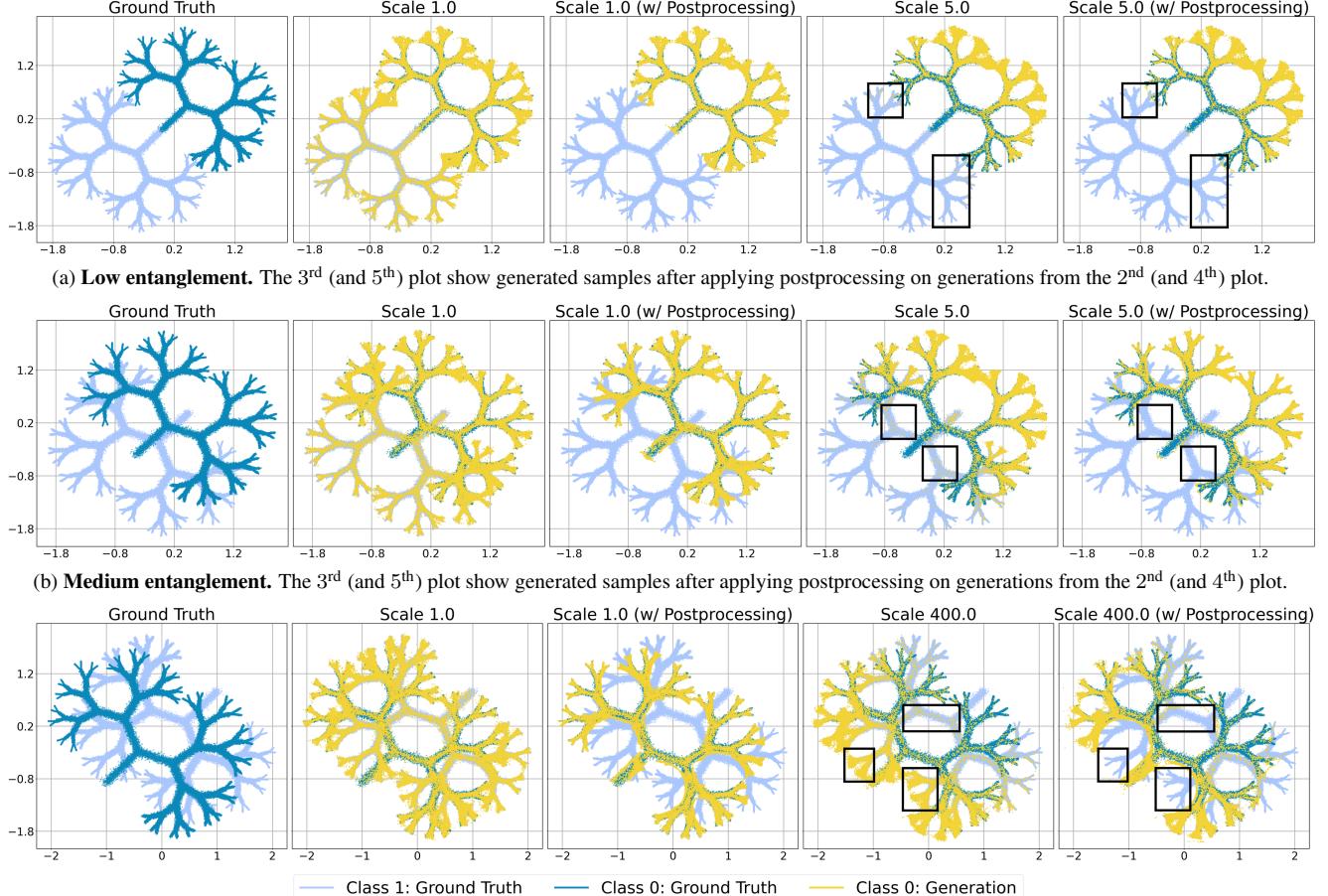
**Figure S8. Classifier guidance with a linear classifier (Fig. 2b) per-class visualization.**

## E. More Visualizations

Fig. S7 visualizes the effect of our postprocessing in cases where data is limited.

Fig. S8 visualizes the per-class denoising diffusion trajectories for the 2<sup>nd</sup> column in Fig. 2b. This shows that both classes have a nontrivial portion of incorrect generations.

In Fig. S9, we show results the 2D fractal results for the class which was not presented in Fig. 4. As shown in the main paper, our proposed postprocessing step improves the generation quality around the decision boundaries.



**Figure S9. Classifier guidance with flow-matching based postprocessing (Sec. 3.4) on 2D fractal data.** This figure is complementary to Fig. 4, showing generations for the other class that are not presented in Fig. 4. The postprocessing step always improves the generated samples to match the real data, especially on boundaries between the two classes. For scenarios where scale equals 1, before postprocessing, a large portion of the generations fall into the incorrect category as the classifier signal is not strong enough. With the proposed postprocessing step, we observe correct generations aligned with the conditioning information. For large scales, though the generations are generally correct as the signal from the classifier starts to dominate the generation process, there are still outliers as highlighted. The proposed postprocessing significantly corrects these low-quality generations while not altering already-high-quality generations.