SiT: Exploring Flow and Diffusion-based Generative Models with Scalable Interpolant Transformers

Anonymous CVPR submission

Paper ID 1444

Abstract

We present Scalable Interpolant Transformers (SiT), a family of generative models built on the backbone of Diffusion Transformers (DiT). The interpolant framework, which allows for connecting two distributions in a more flexible way than standard diffusion models, makes possible a modular study of various design choices impacting generative models built on dynamical transport: using discrete vs. continuous time learning, deciding the model to learn, choosing the interpolant connecting the distributions, and deploying a deterministic or stochastic sampler. By carefully introducing the above ingredients, SiT surpasses DiT uniformly across model sizes on the conditional ImageNet 256x256 benchmark using the exact same backbone, number of parameters, and GFLOPs. By exploring various diffusion coefficients, which can be tuned separately from learning, SiT achieves an FID-50K score of 2.06.

1. Introduction

Contemporary success in image generation has come from a combination of algorithmic advances, mediated by methods such as score-based diffusion and denoising diffusion probabilistic models, coupled with progress in scaling neural network models and data, as well as improvements in model architecture. State-of-the-art diffusion models [20, 54] proceed by incrementally transforming data into Gaussian noise as prescribed by an iterative stochastic process, which can be specified either in discrete or continuous time. At an abstract level, this corruption process can be viewed as forming a connection between the data distribution and a Gaussian distribution, and defines a time-dependent distribution that is iteratively smoothed from the original data distribution into a Gaussian. Diffusion-based generative models learn to reverse this corruption process, which corresponds to pushing noise samples backwards along this connection. The objects learned to perform this transformation are conventionally either a discrete-time denoiser [20] or a model of the score of the distribution connecting the data and the Gaussian [54], though alternatives of these choices exist [22, 45].

Model	Params(M)	Training Steps	FID ↓
DiT-S	33	400K	68.4
SiT-S	33	400K	**57.6**
DiT-B	130	400K	43.5
SiT-B	130	400K	**33.5**
DiT-L	458	400K	23.3
SiT-L	458	400K	**18.8**
DiT-XL	675	400K	19.5
SiT-XL	675	400K	**17.2**
DiT-XL	675	7M	9.6
SiT-XL	675	7M	**8.6**
DiT-XL (cfg=1.5)	675	7M	2.27
SiT-XL (cfg=1.5)	675	7M	**2.06**

Table 1. **Scalable Interpolant Transformers.** We systematically vary the following aspects of a generative model: **discrete vs. continuous time**, **model prediction**, **interpolant**, **choice of sampler**. The resulting Scalable Interpolant Transformer (SiT) model, under identical training compute, consistently outperforms the Diffusion Transformer (DiT) in generating 256×256 ImageNet images. All models employ a patch size of 2. In this work we ask the question: *What is the source of the performance gain?*

The neural network architectures used to represent these objects have been shown to perform well on a variety of tasks. While diffusion models were originally built upon a U-Net backbone [20, 43], recent work has highlighted that architectural advances in vision such as the vision transformer [17] can be incorporated into the standard diffusion model pipeline to improve performance [39]. The aims of [39] were to push improvements on the model side of the duality of model and method.

Orthogonally, significant research effort has gone into exploring the structure of the noising process, which has been shown to lead to performance benefits [26]. Yet, many

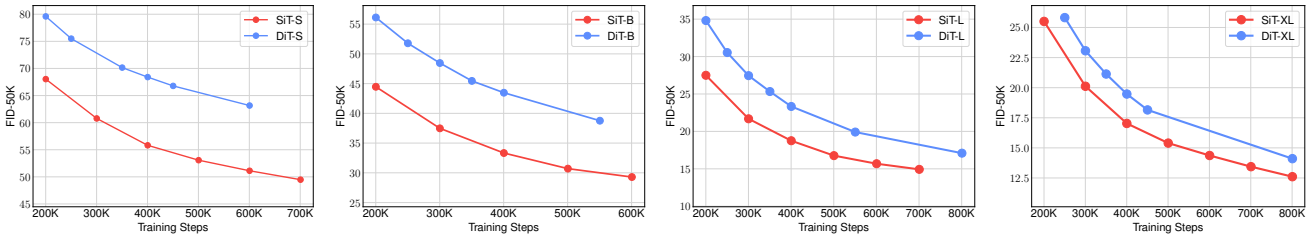

Figure 1. **SiT observes improvement in FID across all model sizes.** We show FID-50K over training iterations for both DiT and SiT models. All results are produced by a 250 steps SDE sampler. Across all model sizes, SiT converges faster.

Figure 2. We show selected samples from our largest SiT-XL models trained on ImageNet [44] at 256×256 resolution with cfg = 4.0.

of these efforts do not move past the notion of passing data samples through a diffusion process, which is a restricted definition of a connection between the data and the Gaussian. Here, we use the recently-introduced framework of *stochastic interpolants* to explore the effect of the connection at large scale.

Intuitively, we expect that the difficulty of the *learning problem* can be related to both the specific connection chosen and the object that is learned. Our aim is to clarify these design choices, in order to simplify the learning problem and improve performance. To glean where potential benefits arise in the learning problem, we start with the denoising diffusion probability model (DDPM) and sweep through adaptations of both: (i) which object to learn, and (ii) which interpolant to choose to reveal best practices.

In addition to the learning problem, there is a *sampling problem* that must be solved at inference time. It has been acknowledged for diffusion models that sampling can be accomplished either deterministically or stochastically [53], and the choice of sampling method can be made after learning with a fixed model. Yet, the diffusion coefficient functions used for stochastic sampling algorithms are typically presented as intrinsically tied to the forward noising process, which need not be the case in general.

Throughout this paper, we will explore how the design of the interpolant and the use of the resulting model as either a deterministic or a stochastic sampler can impact performance. Along the way, we carefully consider how each move away from a standard denoising diffusion model impacts performance. In summary, our **main contributions** are:

- By moving from **discrete to continuous time**, changing the **model prediction**; **interpolant**, and the **choice of sampler**, we observe a consistent improvement over the Diffusion Transformer (DiT).
- We systematically study where these improvements come from by addressing these factors one by one: learning on continuous time; learning a *velocity model* as compared to a *score model*; changing the interpolant connecting the the two distributions; and using the velocity in an SDE sampler with precise scale of noise.
- We show that the SDE for the interpolant can be instantiated using just a model of the velocity field, which we use to push the performance of these methods beyond previous results.

2. SiT: Scalable Interpolant Transformers

We begin by recalling the main ingredients for building flow-based and diffusion-based generative models.

2.1. Flows and diffusions

In recent years, a flexible class of generative models based on turning data $\mathbf{x}_* \sim p(\mathbf{x})$ into noise $\boldsymbol{\varepsilon} \sim \mathsf{N}(0, \mathbf{I})$ have been introduced. These models use the time-dependent process

$$\mathbf{x}_t = \alpha_t \mathbf{x}_* + \sigma_t \boldsymbol{\varepsilon}, \tag{1}$$

where α_t is a decreasing function of t and σ_t is an increasing function of t. Stochastic interpolants and other flow matching methods [1, 2, 33, 34] define the process (1) on $t \in [0, 1]$, and set $\alpha_0 = 1$, $\alpha_1 = 0$, $\sigma_0 = 0$, and $\sigma_1 = 1$ so that $\mathbf{x}_t$ interpolates exactly between $\mathbf{x}_*$ at time $t = 0$ and ϵ and time $t = 1$. By contrast, score-based diffusion models set α_t indirectly through choice of a stochastic differential equation (SDE) whose equilibrium distribution is $\mathsf{N}(0, \mathbf{I})$. Moreover, they consider the process $\mathbf{x}_t$ on an interval $[0, T]$ with T large enough that $\mathbf{x}_T$ is approximately Gaussian.

Probability flow. Common to both stochastic interpolants and score-based diffusion models is the observation that the process $\mathbf{x}_t$ can be sampled dynamically using either a probability flow ordinary differential equation (ODE) or an SDE. More precisely, the probability distribution $p_t(\mathbf{x})$ of $\mathbf{x}_t$ coincides with the distribution of the probability flow ODE

$$\dot{\mathbf{X}}_t = \mathbf{v}(\mathbf{X}_t, t) \tag{2}$$

where the velocity $\mathbf{v}(\mathbf{x}, t)$ is given by the conditional expectation,

$$\begin{aligned} \mathbf{v}(\mathbf{x}, t) &= \mathbb{E}[\dot{\mathbf{x}}_t | \mathbf{x}_t = \mathbf{x}], \\ &= \dot{\alpha}_t \mathbb{E}[\mathbf{x}_* | \mathbf{x}_t = \mathbf{x}] + \dot{\sigma}_t \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}]. \end{aligned} \tag{3}$$

Equation (3) is derived in the appendix. By solving the probability flow ODE (2) backwards in time from $\mathbf{X}_T = \boldsymbol{\varepsilon} \sim \mathsf{N}(0, \mathbf{I})$, we can generate samples from the data distribution $p(\mathbf{x})$. Therefore, (2) is a flow-based generative model.

Reverse-time SDE. The probability distribution $p_t(\mathbf{x})$ of $\mathbf{x}_t$ also coincides with the distribution of the reverse-time SDE [3]

$$d\mathbf{X}_t = \mathbf{v}(\mathbf{X}_t, t)dt + \frac{1}{2} w_t \mathbf{s}(\mathbf{X}, t)dt + \sqrt{w_t} d\bar{\mathbf{W}}_t \tag{4}$$

where $\bar{\mathbf{W}}_t$ is a reverse-time Wiener process, $w_t > 0$ is a time-dependent diffusion coefficient, $\mathbf{v}(\mathbf{x}, t)$ is the velocity defined in (3), and where $\mathbf{s}(\mathbf{x}, t) = \nabla \log p_t(\mathbf{x})$ is the score. Similar to $\mathbf{v}$, this score is given by the conditional expectation

$$\mathbf{s}(\mathbf{x}, t) = -\sigma_t^{-1} \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}] \tag{5}$$

This equation is also derived in the appendix. By solving the reverse SDE (4) backwards in time from $\mathbf{X}_T = \boldsymbol{\varepsilon} \sim \mathsf{N}(0, \mathbf{I})$ we can generate samples from the data distribution $p(\mathbf{x})$. Hence, (2) is a diffusion-based generative model.

Design choices. Score-based diffusion models typically tie the choice of α_t, σ_t, and w_t in (4) to the drift and diffusion coefficients used in the forward SDE that generates $\mathbf{x}_t$ (see (10) below). The stochastic interpolant framework shows that there is more flexibility in the choice of α_t, σ_t, and w_t, which is consistent with the fact $\mathbf{x}_t$ does not need to be generated from a forward SDE. Below, we will exploit this flexibility to construct generative models that outperform score-based diffusion models on standard benchmarks.

2.2. Estimating the score and the velocity

Practical use of the probability flow ODE (2) and the reverse-time SDE (4) as generative models relies on our ability to estimate the velocity $\mathbf{v}(\mathbf{x}, t)$ and score $\mathbf{s}(\mathbf{x}, t)$ fields that enter these equations. The key observation made in score-based diffusion models is that the score can be estimated parametrically as $\mathbf{s}_\theta(\mathbf{x}, t)$ using the loss

$$\mathcal{L}_{\mathrm{s}}(\theta) = \int_0^T \mathbb{E}[\|\sigma_t \mathbf{s}_\theta(\mathbf{x}_t, t) + \boldsymbol{\varepsilon}\|^2] \mathrm{d}t. \tag{6}$$

This loss can be derived by using (5) along with standard properties of the conditional expectation. Similarly, the velocity in (3) can be estimated parametrically as $\mathbf{v}_\theta(\mathbf{x}, t)$ via the loss

$$\mathcal{L}_{\mathrm{v}}(\theta) = \int_0^T \mathbb{E}[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \dot{\alpha}_t \mathbf{x}_* - \dot{\sigma}_t \boldsymbol{\varepsilon}\|^2] \mathrm{d}t. \tag{7}$$

We note that any time-dependent weight can be included under the integrals in both (6) and (7). These weight factors are key in the context of score-based models when T becomes large [29]; in contrast, with stochastic interpolants where $T = 1$ without any bias, these weights are less important. For this reason, we do not make use of weights in our numerical experiments.

Model prediction. Only one of the two quantities $\mathbf{s}_\theta(\mathbf{x}, t)$ and $\mathbf{v}_\theta(\mathbf{x}, t)$ needs to be estimated in practice. This follows from the constraint

$$\begin{aligned} \mathbf{x} &= \mathbb{E}[\mathbf{x}_t | \mathbf{x}_t = \mathbf{x}], \\ &= \alpha_t \mathbb{E}[\mathbf{x}_* | \mathbf{x}_t = \mathbf{x}] + \sigma_t \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}], \end{aligned} \tag{8}$$

which can be used to re-express the score (5) in terms of the velocity (3) as

$$\mathbf{s}(\mathbf{x}, t) = \sigma_t^{-1} \frac{\alpha_t \mathbf{v}(\mathbf{x}, t) - \dot{\alpha}_t \mathbf{x}}{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}. \tag{9}$$

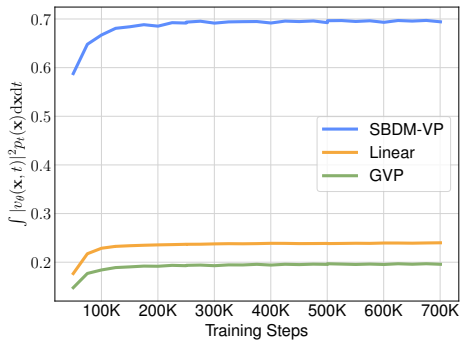

Figure 3. **Path length.** The path length $\mathcal{C}(v) = \int_0^1 \mathbb{E}[|\mathbf{v}(\mathbf{x}_t, t)|^2]\mathrm{d}t$ arising from the velocity field at different training steps for the various models considered: SBDM (VP), the linear interpolant, and the general VP (GVP) interpolant; approximated by 10000 datapoints.

We will use this relation to specify our **model prediction**. Conversely, we can also express $\mathbf{v}(\mathbf{x}, t)$ in terms of $\mathbf{s}(\mathbf{x}, t)$. In our experiments, we typically learn the velocity field $\mathbf{v}(\mathbf{x}, t)$ and use it to express the score $\mathbf{s}(\mathbf{x}, t)$ when using an SDE for sampling.

Note that $\dot{\alpha}_t < 0$ and $\dot{\sigma}_t > 0$ so that the denominator of (9) is never zero. By contrast, σ_t vanishes at $t = 0$, so that the σ_t^{-1} term in (9) is singular there[1]. This suggests the choice $w_t = \sigma_t$ in (4) to cancel this singularity, which we will explore in the numerical experiments.

2.3. Specifying the interpolating process

Score-based diffusion. In score-based diffusion models, the choice of α_t and σ_t in (1) is typically determined by the choice of the forward SDE used to define this process, though recent work has tried to reconsider this [29]. For example, if we use the standard variance-preserving (VP) SDE

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta_t \mathbf{x}_t \mathrm{d}t + \sqrt{\beta_t}\mathrm{d}\mathbf{W}_t \tag{10}$$

for some $\beta_t > 0$, it can be shown (see the Appendix) that the solution to (10) has the same probability distribution $p_t(\mathbf{x})$ as the process $\mathbf{x}_t$ defined in (1) for the choice

$$\text{VP:}\ \ \alpha_t = e^{-\frac{1}{2}\int_0^t \beta_s \mathrm{d}s}, \qquad \sigma_t = \sqrt{1 - e^{-\int_0^t \beta_s \mathrm{d}s}}. \tag{11}$$

The only design flexibility in (11) comes from the choice of β_t in (10), because α_t and σ_t are both determined by it. For example, setting $\beta_t = 1$ leads to $\alpha_t = e^{-t}$ and $\sigma_t = \sqrt{1 - e^{-2t}}$. This choice necessitates taking the final time T sufficiently large to reduce the bias induced by the fact that the solution to the SDE (10) only converges to the noise $\boldsymbol{\varepsilon} \sim \mathsf{N}(0, \mathbf{I})$ as $t \to \infty$.

[1]We remark that $\mathbf{s}(\mathbf{x}, t)$ can be shown to be non-singular at $t = 0$ analytically if the data distribution $p(\mathbf{x})$ has a smooth density [2], though this singularity will appear in numerical implementations in general.

General interpolants. In the stochastic interpolant framework, the process (1) is defined explicitly and without any reference to a forward SDE, which creates more flexibility in the choice of α_t and σ_t. Specifically, any choice satisfying: i) $\alpha_t^2 + \sigma_t^2 > 0$ for all $t \in [0, 1]$, ii) α_t and σ_t are differentiable, iii), $\alpha_1 = \sigma_0 = 0$, and iv) $\alpha_0 = \sigma_1 = 1$ gives a process that interpolates without bias between $\mathbf{x}_{t=0} = \mathbf{x}_*$ and $\mathbf{x}_{t=1} = \boldsymbol{\varepsilon}$. In our numerical experiments, we exploit this design flexibility to test, in particular, the choices

$$\begin{aligned} &\text{LIN:} && \alpha_t = 1 - t, && \sigma_t = t, \\ &\text{GVP:} && \alpha_t = \cos(\tfrac{1}{2}\pi t), && \sigma_t = \sin(\tfrac{1}{2}\pi t), \end{aligned} \tag{12}$$

where GVP is the general VP which has constant variance across time for any endpoint distributions with the same variance. We note that the velocity and score fields $\mathbf{v}(\mathbf{x}, t)$ and $\mathbf{s}(\mathbf{x}, t)$ entering (2) and (4) depend on the choice of α_t and σ_t, which must be specified before learning. This is in contrast to the diffusion coefficient $w(t)$, as we now describe.

2.4. Specifying the diffusion coefficient

As stated earlier, in score-based diffusion models, the diffusion coefficient used in the reverse SDE (4) is usually taken to be the same as the one in the forward SDE (10). That is, we set $w_t = \beta_t$. In the stochastic interpolant framework, this choice is again subject to greater flexibility, and *any* $w_t > 0$ can be used. Interestingly, this choice can be made *after* learning, as it does not affect the velocity $\mathbf{v}(\mathbf{x}, t)$ or the score $\mathbf{s}(\mathbf{x}, t)$. This is unlike α_t and σ_t, which both impact $\mathbf{v}(\mathbf{x}, t)$ and $\mathbf{s}(\mathbf{x}, t)$, and therefore need to be chosen prior to learning. In our experiments, we exploited this flexibility by considering the choices listed in Table 2.

2.5. Time-discretization and link with DDPM

In practice, both the probability flow ODE (2) and the reverse-time SDE (4) need to be discretized in time. This allows us to make a link with Denoising Diffusion Probabilistic Models (DDPM), which we discuss next.

Assuming that we discretize time using a grid $0 = t_0 < t_1 < t_2 < \ldots < t_N = T$, the process (1) can be evaluated at each grid point, $\mathbf{x}_{t_i} = \alpha_{t_i}\mathbf{x}_* + \sigma_{t_i}\boldsymbol{\varepsilon}$, and both the velocity and the score can be estimated on these points via the losses

$$\mathcal{L}_{\mathrm{s}}^N(\theta) = \sum_{i=0}^N \mathbb{E}[\|\sigma_{t_i}\mathbf{s}_\theta(\mathbf{x}_{t_i}, t_i) + \boldsymbol{\varepsilon}\|^2], \tag{13}$$

$$\mathcal{L}_{\mathrm{v}}^N(\theta) = \sum_{i=0}^N \mathbb{E}[\|\mathbf{v}_\theta(\mathbf{x}_{t_i}, t_i) - \dot{\alpha}_{t_i}\mathbf{x}_* - \dot{\sigma}_{t_i}\boldsymbol{\varepsilon}\|^2]. \tag{14}$$

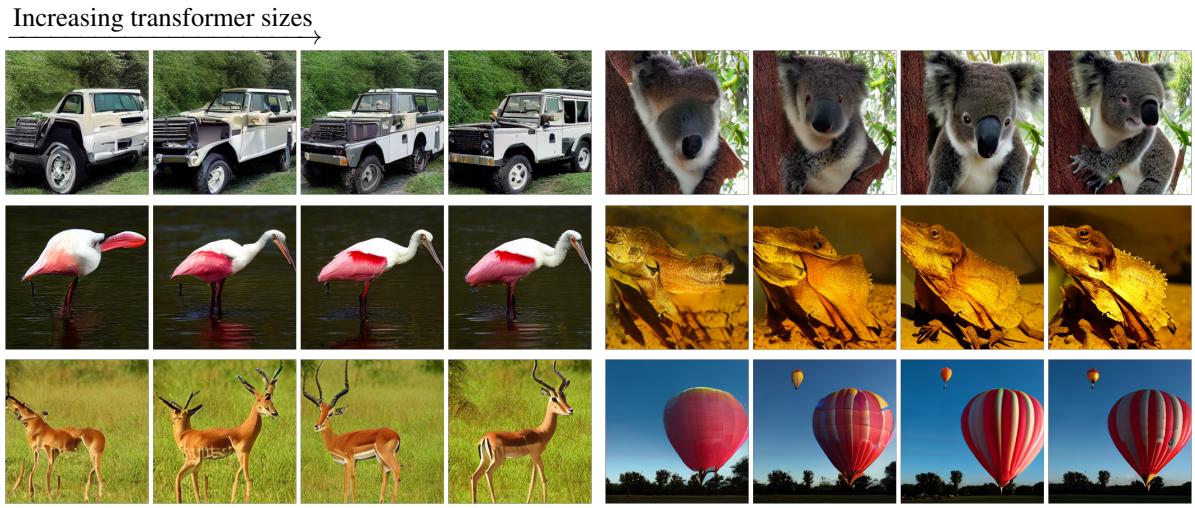

Figure 4. **Increasing transformer size increases sample quality**. *Best viewed zoomed-in*. We sample from all 4 of our SiT model (SiT-S, SiT-B, SiT-L and SiT-XL) after 400K training steps using the same latent noise and class label.

Expression for w_t
$\beta_t = -\sigma_t(\dot{\sigma}_t - 2\frac{\sigma_t\dot{\alpha}_t}{\alpha_t})$
σ_t
$1-t$
$\sin^2(\pi t)$
$(\cos(\pi t)+1)^2$
$(\cos(\pi t)-1)^2$

Table 2. **Diffusion coefficients.** In this work, we consider several different time-dependent diffusion coefficients w_t. These can be specified after learning to maximize performance.
The choice $w_t = \beta_t$ given in the first row corresponds to using the expression for SBDM diffusion coefficient that is coupled to the corruption connecting the Gaussian and the data. This is derived from Eq. (11).
The choice $w_t = \sigma_t$ in the second row is used to eliminate the singularity at $t = 0$ following the explanation at the end of Sec. 2.2.
In other choices we experiment with removing diffusivity at different time in sampling.

Moreover, the learned $\mathbf{s}_\theta(\mathbf{x}, t_i)$ and $\mathbf{v}_\theta(\mathbf{x}, t_i)$ are all that is needed to integrate the probability flow ODE (2) and the reverse-time SDE (4) on the same grid. The resulting procedure, in which we define $\mathbf{x}_t$ iteratively on the grid, is a generalization of DDPM. Starting from $\mathbf{x}_{t_0} = \mathbf{x}_*$, we set for $i \geq 0$,

$$\mathbf{x}_{t_{i+1}} = \sqrt{1 - h\beta_{t_i}}\mathbf{x}_{t_i} + \sqrt{h\beta_{t_i}}\boldsymbol{\varepsilon}_{t_i}, \tag{15}$$

where $h = t_{i+1} - t_i$ and where we assume that the grid is uniform. Because $\sqrt{1 - h\beta_{t_i}} = 1 - \frac{1}{2}h\beta_{t_i} + o(h)$, it is easy to see that (15) is a consistent time-discretization of the forward SDE (10). Our results show that it is not necessary to specify the time discretized process $\mathbf{x}_{t_i}$ using (15), but instead we can directly use (1).

2.6. Interpolant Transformer Architecture

The backbone architecture and capacity of generative models are also crucial for producing high-quality samples. In order to eliminate any confounding factor and focus on our exploration, we strictly follow the standard Diffusion Transformer (DiT) [39] and its configurations. This way, we can also test the scalability of our model across various model sizes. Here we briefly introduce the model design. It is computationally intense to process high-resolution images in generative modeling. Latent diffusion models (LDMs) [42] address this by first downsampling images into a smaller latent embedding space using an encoder E, and then training a diffusion model on $z = E(x)$. New images are created by sampling z from the model and decoding it back to images using a decoder $x = D(z)$. SiT is also a latent generative model and we use the same pre-trained VAE encoder and decoder models originally used in Stable Diffusion [42]. SiT processes a spatial input z (shape $32 \times 32 \times 4$ for $256 \times 256 \times 3$ images) by first 'patchifying' it into T linearly embedded tokens of dimension d. We always use a patch size of 2 in these models as they achieve the best sample quality. It then applies standard ViT [17] sine-cosine positional embeddings to these tokens. We use a series of N SiT transformer blocks, each with hidden dimension d. Our configurations – SiT-S,B,L,XL – vary in model size (parameters) and compute (Flops), allowing for a model scaling analysis. For class-conditional generation on ImageNet, we use the AdaLN-Zero block [39] to process additional conditional information. Configuration details are listed in Table 3.

Model	Layers N	Hidden size d	Heads
SiT-S	12	384	6
SiT-B	12	768	12
SiT-L	24	1024	16
SiT-XL	28	1152	16

Table 3. **Details of SiT models.** We follow DiT [39] for the Small (S), Base (B), Large (L) and XLarge (XL) model configurations.

3. Experiments

To provide a more detailed answer to the question raised in Tab. 1 and make a fair comparison between DiT and SiT, we gradually transition from a DiT model to a SiT model in the following four subsections. Throughout all of our experiments, we uses a DiT-B model at 400K training steps as our backbone. For the ODE solver, we adopt a fixed Heun integrator; for the SDE solver, we used an Euler-Maruyama integrator. With both solver choices we limit the NFE to be 250 to match the evaluation steps used in DDPM. All the numbers presented in the table are FID-50K scores evaluated on the training set.

3.1. Discrete to Continuous Time

To understand the role of continuous-time versus discrete-time models, we study discrete-time DDPM against continuous-time SBDM-VP with estimation of the score. The results are presented in Table 4, where we find a marginal improvement in FID when going to a continuous-time score.

	Model	*Objective*	DDIM / ODE
DDPM	Noise	$\mathcal{L}_{\mathrm{s}}^{N}$	44.2
SBDM-VP	Score	$\mathcal{L}_{\mathrm{s}}$	**43.6**

Table 4. **DDPM vs. SBDM.** We find that SBDM-VP with a score model parameterization produces a lower FID than DDPM with a noise model parameterization.

3.2. Model parameterizations

To clarify the role of the model parameterization in the context of SBDM-VP, we now compare learning (i) a score model using (6), (ii) a weighted score model (see appendix), or (iii) a velocity model using (7). The results are shown in Table 5, where we find that we obtain a significant performance improvement by learning a weighted score model or a velocity model. For the SBDM-VP model class, the weighted score performs best.

3.3. Choice of interpolant

Section 2 highlights that there are many possible ways to build a connection between the data distribution and a Gaussian by varying the choice of α_t and σ_t in the definition of the interpolant (1). To understand the role of this choice, we now study the benefits of moving away from the commonly-used SBDM-VP setup. We consider learning a velocity model $\mathbf{v}(\mathbf{x}, t)$ with the linear and GVP interpolants presented in (12), which make the connection between the Gaussian and the data distribution exact in finite time. We benchmark these models against the SBDM-VP case by comparing FIDs in Table 6, where we find that both the GVP and LIN interpolants obtain significantly improved performance. One possible explanation for this observation is given in Fig. 3, where we see that the path length (transport cost) is reduced when changing from SBDM-VP to GVP or LIN. Numerically, we also note that for SBDM-VP, $\dot{\sigma}_t = \beta_t e^{-\int_0^t \beta_s ds}/(2\sigma_t)$, which becomes singular as $t \to 0$: this can pose numerical difficulties inside $\mathcal{L}_{\mathrm{v}}$. This issue does not appear with the GVP and LIN interpolants.

Interpolant	*Model*	*Objective*	ODE
SBDM-VP	$\mathcal{L}_{\mathrm{s}}$	Score	43.6
SBDM-VP	$\mathcal{L}_{\mathrm{s}_\lambda}$	Score	**39.1**
SBDM-VP	$\mathcal{L}_{\mathrm{v}}$	Velocity	39.8

Table 5. **Effect of model parameterization.** We compare learning a score, to a weighted score, to a velocity model in SBDM. We find best performance with a weighted score, which performs closely to learning the velocity. The weighted score is given in the appendix.

Interpolant	*Model*	*Objective*	FID
SBDM-VP	$\mathcal{L}_{\mathrm{v}}$	Velocity	39.1
LIN	$\mathcal{L}_{\mathrm{v}}$	Velocity	34.8
GVP	$\mathcal{L}_{\mathrm{v}}$	Velocity	**34.6**

Table 6. **Effect of interpolant.** We compare SBDM-VP to two choices of interpolant, both of which lead to significant performance improvements.

3.4. Deterministic vs stochastic sampling

Using the SBDM diffusion coefficient. As shown in Sec. 2, given a learned model, we can sample using either the probability flow equation (2) or an SDE (4). In SBDM we conventionally take as diffusion coefficient $w_t = \beta_t$. Since from (11) β_t satisfies the relation $\beta_t = -\sigma_t(\dot{\sigma}_t - 2\frac{\sigma_t\dot{\alpha}_t}{\alpha_t})$, when using interpolants, we begin by testing $w_t = -\sigma_t(\dot{\sigma}_t - 2\frac{\sigma_t\dot{\alpha}_t}{\alpha_t})$ for several choices of α_t and β_t in (1) (i.e. we tie the choice of diffusion coefficient to what is used in SBDM).

Our results are shown in Tab. 8, where we find a significant performance improvement by sampling with an SDE over the ODE (SDE sampling uses 250 function evaluations),

Interpolant	*Model*	*Objective*	$w_t = \beta_t$	$w_t = \sigma_t$	$w_t = \sin^2(\pi t)$
SBDM-VP	velocity	$\mathcal{L}_v$	37.8	38.7	39.2
	score	$\mathcal{L}_{s_\lambda}$	35.7	37.1	37.7
GVP	velocity	$\mathcal{L}_v$	**32.9**	33.4	33.6
	score	$\mathcal{L}_s$	38.0	33.5	33.2
Linear	velocity	$\mathcal{L}_v$	33.6	33.5	33.3
	score	$\mathcal{L}_s$	41.0	35.3	34.4

Table 7. **Evaluation of our SDE samplers.** All results in the table are FID-50K scores produced by a SiT-B model at 400K training steps. When learning a velocity, we write the score for use in the SDE (4) using (9). The last three columns specify different diffusion coefficients w_t detailed in Tab. 2. To make the SBDM-VP competitive when learning a score, we use a weighted score given in the appendix, as per the remarks below (7).

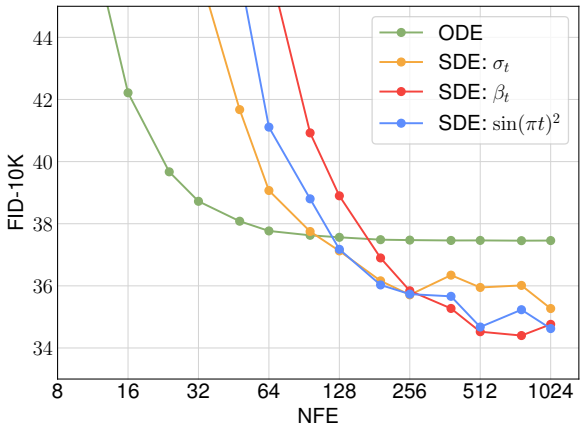

Figure 5. **Comparison of ODE and SDE w/ choices of diffusion coefficients.** We evaluate each sampler using the SiT-B model trained with LIN interpolant, learning the $\mathbf{v}(\mathbf{x}, t)$, at 400K training steps.

Interpolant	*Model*	*Objective*	ODE	SDE
SBDM-VP	$\mathcal{L}_v$	Velocity	39.1	37.8
LIN	$\mathcal{L}_v$	Velocity	34.8	33.6
GVP	$\mathcal{L}_v$	Velocity	34.6	**32.9**

Table 8. **ODE vs. SDE, SBDM diffusion.** For the fixed SBDM diffusion coefficient, we find improved performance with SDE-based sampling (FID-50k).

in line with the bounds given in [2]. We note that under a fixed computational budget, the performance of ODE and SDE integrators may differ, as shown in Fig. 5. The ODE converges faster with fewer number of functions evaluations, while the SDE is capable of reaching a much lower final FID score when given a larger computational budget.

Tunable diffusion coefficient. Motivated by the improved performance of SDE sampling, we now consider the effect of tuning the diffusion coefficient in a manner that is distinct from the choices made in SBDM, as detailed in Tab. 2. As shown in Tab. 7, we find that the optimal choice for sampling is both model prediction and interpolant dependent. We picked the three functions with the best performance and sweep through all different combinations of our model prediction and interpolant, and present the result in Tab. 7

We note that the influences of different diffusion coefficients can vary across different model sizes. Empirically, we observe the best choice for our SiT-XL is a velocity model with LIN interpolant and sampled with σ_t coefficient.

3.5. Classifier-free guidance

Classifier-free guidance (CFG) often leads to improved performance for score-based models, and a natural question is if it can also be used for the velocity-based models we advocate for here. In this section, we give a concise justification for doing so, and then empirically show that the drastic gains in performance for DiT case carry across to SiT.

Guidance for a velocity field means that: (i) that the velocity model $\mathbf{v}_\theta(\mathbf{x}, t; \mathbf{y})$ takes class labels y during training, where y is occasionally masked with a null token $\emptyset$; and (ii) during sampling the velocity used is $\mathbf{v}^\zeta_\theta(\mathbf{x}, t; \mathbf{y}) = \zeta\mathbf{v}_\theta(\mathbf{x}, t; \mathbf{y}) + (1 - \zeta)\mathbf{v}_\theta(\mathbf{x}, t; \emptyset)$ for a fixed $\zeta > 0$. In the appendix, we show that this corresponds to sampling the density $p(\mathbf{x}_t)p(\mathbf{y}|\mathbf{x}_t)^\zeta$. This follows from (9), which shows that velocities and scores are linearly related, so that a combination of velocities implies a combination of scores. Given this observation, one can leverage the usual argument for classifier-free guidance for score-based models [19].

Empirically, we find that classifier-free guidance leads to a drastic improvement in quantitative performance. For a CFG scale of $\omega = 1.5$, DiT-XL (675 M parameters) sees an improvement in FID from a 9.6 (non-CFG) down to 2.27 (CFG). For SiT-XL sampled with an SDE, the improvement is similar, from 8.6 to 2.06 (Tab. 1 and Tab. 9). This shows that SiT benefits from the same training and sampling choices explored previously, and can surpass DiTs performance in each training setting, not only with respect

Class-Conditional ImageNet 256x256					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep[7]	6.95	7.36	171.4	**0.87**	0.28
StyleGAN-XL[46]	2.30	**4.02**	265.12	0.78	0.53
Mask-GIT[9]	6.18	-	182.1	-	-
ADM[15]	10.94	6.02	100.98	0.69	0.63
ADM-G, ADM-U	3.94	6.14	215.84	0.83	0.53
CDM[21]	4.88	-	158.71	-	-
RIN[25]	3.42	-	182.0	-	-
Simple Diffusion(U-Net)[22]	3.76	-	171.6	-	-
Simple Diffusion(U-ViT, L)	2.77	-	211.8	-	-
VDM++[29]	2.12	-	267.7	-	-
DiT-XL(cfg = 1.5)[39]	2.27	4.60	278.24	0.83	0.57
SiT-XL(cfg = 1.5, ODE)	2.15	4.60	258.09	0.81	0.60
SiT-XL(cfg = 1.5, SDE:σ_t)	**2.06**	4.50	270.27	0.82	0.59

Table 9. **Benchmarking class-conditional image generation on ImageNet 256x256.** SiT-XL surpasses DiT-XL in FID when either of the samplers, ODE or SDE-based.

to model size, but also with respect to sampling choices.

4. Related Work

Transformers. The transformer architecture [57] has emerged as a powerful tool for application domains as diverse as vision [17, 38], language [59, 60], quantum chemistry [58], and biology [8]. Several works have built on the DiT and have made improvements by modifying the architecture to internally to include masked prediction layers [18, 61]; these choices are orthogonal to the transition from DiT to SiT studied in this work; they may be fruitfully combined in future work.

Training and Sampling in Diffusions. Diffusion models arose from [20, 51, 54] and have close historical relationship with denoising methods [23, 24, 50]. Various efforts have gone into improving the sampling algorithms behind these methods in the context of DDPM [52] and SBDM [26, 53]. Improved Diffusion ODE [62] studies several combinations of model parameterizations (velocity versus noise) and paths (VP versus linear) for sampling an ODE; they report best results for velocity model with smoother probability flow. They focus on lower dimensional experiments, benchmark with likelihoods, and don't consider the SDE sampling. Here, we explore the effects of changing between VP and linear paths, as well as score and velocity parameterizations, on large scale Imagenet-256 and document how these choices individually improve performance. We also provide an exploration of how FIDs change for a fixed choice of model and path, when we explore a family of available sampling algorithms, using not just the ODE associated with the velocity but a family of SDEs indexed by a choice of diffusion coefficients, which brings observations about the flexibility of sampling algorithms from [2] into practice.

Interpolants and flow matching. Velocity field parameterizations using the linear path were also studied in [33, 34], and were generalized to the manifold setting in [4]. A tradeoff in bounds on the KL divergence between the target distribution and the distribution predicted by the model arises when considering using an SDE to sample vs an ODE [2], where it is shown that minimizing the objectives presented in this work controls KL for SDEs, but not for ODEs. Error bounds for SDE-based sampling with score-based diffusion models are studied in [10, 11, 31, 32]. Error bounds on the ODE are also explored in [5, 12], in addition to the Wasserstein bounds provided in [1].

Related works of [33, 34] also consider learning deterministic velocity models with the linear interpolant, though the treatment of the score and the SDE is given in [2]. Recent works try to improve the sampling [41, 55] by learning a mini-batch optimal coupling between the Gaussian and data distribution, though it is unclear if these auxiliary actions significantly impact performance in high dimensions. Various works also consider learning a stochastic bridge connecting the two distributions [14, 35, 40, 49].

Diffusion in Latent Space. Generative modeling in latent space [42, 56] is a tractable approach for modeling high-dimensional data. The approach has been applied beyond images to video generation [6], which is a yet-to-be explored and promising application area for velocity trained models. [13] also train velocity models in the latent space of the pretrained Stable Diffusion VAE. They show promising results for the DiT-B architecture with a final FID-50K of 4.46; their study was one motivation for the investigations in this work on which aspects of these models contribute to the gains in performance over DiT. The pretrained VAEs in this work were trained on the LAION dataset [47, 48].

5. Conclusion

In this work, we have presented Scalable Interpolant Transformers, a simple and powerful framework for image generation tasks. Within the framework, we explored the tradeoffs between a number of key design choices: the choice of a continuous or discrete-time model, the choice of interpolant, the choice of model prediction, and the choice of diffusion coefficient. We highlighted the advantages and disadvantages of each choice and demonstrated how careful decisions can lead to significant performance improvements.

References

[1] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants, 2023. 3, 8

[2] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023. 3, 4, 7, 8, 1, 5, 6

[3] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. 3

[4] Heli Ben-Hamu, Samuel Cohen, Joey Bose, Brandon Amos, Maximilian Nickel, Aditya Grover, Ricky T. Q. Chen, and Yaron Lipman. Matching normalizing flows and probability paths on manifolds. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, pages 1749–1763. PMLR, 2022. 8

[5] Joe Benton, George Deligiannidis, and Arnaud Doucet. Error bounds for flow matching methods, 2023. 8

[6] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 8

[7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019. 8

[8] Abel Chandra, Laura Tünnermann, Tommy Löfstedt, and Regina Gratz. Transformer-based deep learning for predicting protein properties in the life sciences. *eLife*, 12:e82819, 2023. 8

[9] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer, 2022. 8

[10] Hongrui Chen, Holden Lee, and Jianfeng Lu. Improved analysis of score-based generative modeling: User-friendly bounds under minimal smoothness assumptions. *ICML*, 2022. 8

[11] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations*, 2023. 8

[12] Sitan Chen, Giannis Daras, and Alex Dimakis. Restoration-degradation beyond linear diffusions: A non-asymptotic analysis for DDIM-type samplers. In *Proceedings of the 40th International Conference on Machine Learning*, pages 4462–4484. PMLR, 2023. 8

[13] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023. 8

[14] Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. In *Advances in Neural Information Processing Systems*, pages 17695–17709. Curran Associates, Inc., 2021. 8

[15] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. 8, 4, 7

[16] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion, 2022. 6

[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, 2021. 1, 5, 8

[18] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. *arXiv preprint arXiv:2303.14389*, 2023. 8

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 7, 4

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 1, 8

[21] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation, 2021. 8

[22] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images, 2023. 1, 8

[23] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. 8

[24] Aapo Hyvärinen. Sparse code shrinkage: Denoising of non-gaussian data by maximum likelihood estimation. *Neural Computation*, 11(7):1739–1768, 1999. 8

[25] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation, 2023. 8

[26] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. 1, 8

[27] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021. 6

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 6

[29] Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation, 2023. 3, 4, 8

[30] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2023. 6

[31] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence for score-based generative modeling with polynomial complexity. *arXiv:2206.06227*, 2022. 8

[32] Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence of score-based generative modeling for general data distributions. In *Proceedings of The 34th International Conference on Algorithmic Learning Theory*, pages 946–985, 2023. 8

[33] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. 3, 8

[34] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. 3, 8

[35] Xingchao Liu, Lemeng Wu, Mao Ye, and Qiang Liu. Let us build bridges: Understanding and extending diffusion generative models, 2022. 8

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 6

[37] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022. 6

[38] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image Transformer. *arXiv:1802.05751*, 2018. 8

[39] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. 1, 5, 6, 8

[40] Stefano Peluchetti. Non-denoising forward-time diffusions, 2022. 8

[41] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings, 2023. 8

[42] 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. 5, 8

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 1

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 2

[45] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. 1

[46] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets, 2022. 8

[47] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021. 8

[48] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 8

[49] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching, 2023. 8

[50] Eero P. Simoncelli and Edward H. Adelson. Noise removal via bayesian wavelet coring. In *Proceedings of 3rd IEEE International Conference on Image Processing*, pages 379–382 vol.1, 1996. 8

[51] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, Lille, France, 2015. PMLR. 8

[52] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 8

[53] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models, 2021. 2, 8, 3

[54] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. 1, 8, 3, 6

[55] 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, 2023. 8

[56] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space, 2021. 8, 3

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *arXiv:1706.03762*, 2017. 8

[58] Ingrid von Glehn, James S. Spencer, and David Pfau. A Self-Attention Ansatz for Ab-initio Quantum Chemistry. *arXiv:2211.13672*, 2023. 8

[59] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning Deep Transformer Models for Machine Translation. *arXiv:1906.01787*, 2019. 8

[60] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. *arXiv:2007.14062*, 2021. 8

[61] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *arXiv preprint arXiv:2306.09305*, 2023. 8

[62] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Improved techniques for maximum likelihood estimation for diffusion odes, 2023. 8

SiT: Exploring Flow and Diffusion-based Generative Models with Scalable Interpolant Transformers

Supplementary Material

A. Proofs

In all proofs below, we use $\cdot$ for dot product and assume all bold notations ($\mathbf{x}$, $\boldsymbol{\varepsilon}$, etc.) are real-valued vectors in $\mathbb{R}^d$. Most proofs are derived from Albergo et al. [2].

A.1. Proof of the probability flow ODE (2) with the velocity in Eq. (3).

Consider the time-dependent probability density function (PDF) $p_t(\mathbf{x})$ of $\mathbf{x}_t = \alpha_t \mathbf{x}_* + \sigma_t \boldsymbol{\varepsilon}$ defined in Eq. (1). By definition, its characteristic function $\hat{p}_t(\mathbf{k}) = \int_{\mathbb{R}^d} e^{i\mathbf{k}\cdot\mathbf{x}} p_t(\mathbf{x}) \mathrm{d}\mathbf{x}$ is given by

$$\hat{p}_t(\mathbf{k}) = \mathbb{E}[e^{i\mathbf{k}\cdot\mathbf{x}_t}] \tag{16}$$

where $\mathbb{E}$ denotes expectation over $\mathbf{x}_*$ and $\boldsymbol{\varepsilon}$. Taking time derivative on both sides, and using the tower property of conditional expectation, we have

$$\begin{aligned}
\partial_t \hat{p}_t(\mathbf{k}) &= i\mathbf{k} \cdot \mathbb{E}[\dot{\mathbf{x}}_t e^{i\mathbf{k}\cdot\mathbf{x}_t}] && (17)\\
&= i\mathbf{k} \cdot \mathbb{E}_{\mathbf{x}\sim p_t}[\mathbb{E}[\dot{\mathbf{x}}_t e^{i\mathbf{k}\cdot\mathbf{x}_t} | \mathbf{x}_t = \mathbf{x}]] && (18)\\
&= i\mathbf{k} \cdot \mathbb{E}_{\mathbf{x}\sim p_t}[\mathbb{E}[(\dot{\alpha}_t \mathbf{x}_* + \dot{\sigma}_t \boldsymbol{\varepsilon}) e^{i\mathbf{k}\cdot\mathbf{x}_t} | \mathbf{x}_t = \mathbf{x}]] && (19)\\
&= i\mathbf{k} \cdot \mathbb{E}_{\mathbf{x}\sim p_t}[\mathbb{E}[(\dot{\alpha}_t \mathbf{x}_* + \dot{\sigma}_t \boldsymbol{\varepsilon}) | \mathbf{x}_t = \mathbf{x}] e^{i\mathbf{k}\cdot\mathbf{x}}] && (20)\\
&= i\mathbf{k} \cdot \mathbb{E}_{\mathbf{x}\sim p_t}[\mathbf{v}(\mathbf{x}, t) e^{i\mathbf{k}\cdot\mathbf{x}}] && (21)
\end{aligned}$$

where $\mathbf{v}(\mathbf{x}, t) = \mathbb{E}[(\dot{\alpha}_t \mathbf{x}_* + \dot{\sigma}_t \boldsymbol{\varepsilon}) | \mathbf{x}_t = \mathbf{x}] = \dot{\alpha}_t \mathbb{E}[\mathbf{x}_* | \mathbf{x}_t = \mathbf{x}] + \dot{\sigma}_t \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}]$ is the velocity defined in Eq. (3). Explicitly, Eq. (21) reads

$$\partial_t \int_{\mathbb{R}^d} e^{i\mathbf{k}\cdot\mathbf{x}} p_t(\mathbf{x}) \mathrm{d}\mathbf{x} = i\mathbf{k} \cdot \int_{\mathbb{R}^d} \mathbf{v}(\mathbf{x}, t) e^{i\mathbf{k}\cdot\mathbf{x}} p_t(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{22}$$

from which we deduce

$$\int_{\mathbb{R}^d} e^{i\mathbf{k}\cdot\mathbf{x}} \partial_t p_t(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_{\mathbb{R}^d} \mathbf{v}(\mathbf{x}, t) \cdot \nabla_{\mathbf{x}} [e^{i\mathbf{k}\cdot\mathbf{x}}] p_t(\mathbf{x}) \mathrm{d}\mathbf{x} = -\int_{\mathbb{R}^d} \nabla_{\mathbf{x}} \cdot [\mathbf{v}(\mathbf{x}, t) p_t(\mathbf{x})] e^{i\mathbf{k}\cdot\mathbf{x}} \mathrm{d}\mathbf{x} \tag{23}$$

where $\nabla_{\mathbf{x}} \cdot [\mathbf{v} p_t] = \sum_{i=1}^d \frac{\partial}{\partial x_i} [v_i p_t]$ is the divergence operator and we used integration by parts to get the second equality. By the properties of Fourier transform, Eq. (23) implies that $p_t(\mathbf{x})$ satisfies the transport equation

$$\partial_t p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \cdot (\mathbf{v}(\mathbf{x}, t) p_t(\mathbf{x})) = 0. \tag{24}$$

Solving this equation by the method of characteristic leads to probability flow ODE (2).

A.2. Proof of the SDE (4)

We show that the SDE (4) has marginal density $p_t(\mathbf{x})$ with any choice of $w_t \geq 0$. To this end, recall that solution to the SDE

$$d\mathbf{X}_t = [\mathbf{v}(\mathbf{X}_t, t) + \frac{1}{2} w_t \mathbf{s}(\mathbf{X}_t, t)] dt + \sqrt{w_t} d\bar{\mathbf{W}}_t$$

has a PDF that satisfies the Fokker-Planck equation

$$\partial_t p_t(\mathbf{x}) = -\nabla_{\mathbf{x}} \cdot \left([\mathbf{v}(\mathbf{x}, t) + \frac{1}{2} w_t \mathbf{s}(\mathbf{x}, t)] p_t(\mathbf{x}) \right) + \frac{1}{2} w_t \Delta_{\mathbf{x}} p_t(\mathbf{x}) \tag{25}$$

where $\Delta_{\mathbf{x}}$ is the Laplace operator defined as $\Delta_{\mathbf{x}} = \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}} = \sum_{i=0}^{d} \frac{\partial^2}{\partial x_i^2}$. Reorganizing the equation and usng the definition of the score $\mathbf{s}(\mathbf{x}, t) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = p_t^{-1}(\mathbf{x}) \nabla_{\mathbf{x}} p_t(\mathbf{x})$, we have

$$\partial_t p_t(\mathbf{x}) = \underbrace{-\nabla_{\mathbf{x}} \cdot [\mathbf{v}(\mathbf{x}, t) p_t(\mathbf{x})]}_{= \partial_t p_t(\mathbf{x}) \text{ by Eq. (24)}} - \frac{1}{2} w_t \nabla_{\mathbf{x}} \cdot [\underbrace{\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) p_t(\mathbf{x})}_{= \nabla_{\mathbf{x}} p_t(\mathbf{x})}] + \frac{1}{2} w_t \Delta_{\mathbf{x}} p_t(\mathbf{x}) \tag{26}$$

$$\implies \qquad 0 = -\frac{1}{2} w_t \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}} p_t(\mathbf{x}) + \frac{1}{2} w_t \Delta_{\mathbf{x}} p_t(\mathbf{x}) \tag{27}$$

By definition of Laplace operator, the last equation holds for any $w_t \geq 0$.

A.3. Proof of the expression for the score in Eq. (5)

We show that $\mathbf{s}(\mathbf{x}, t) = -\sigma_t^{-1} \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}]$. Letting $\hat{f}(\mathbf{k}, t) = \mathbb{E}[\boldsymbol{\varepsilon} e^{i \sigma_t \mathbf{k} \cdot \boldsymbol{\varepsilon}}]$, we have

$$\hat{f}(\mathbf{k}, t) = -\frac{i}{\sigma_t} \nabla_{\mathbf{k}} \mathbb{E}[e^{i \sigma_t \mathbf{k} \cdot \boldsymbol{\varepsilon}}] \tag{28}$$

Since $\boldsymbol{\varepsilon} \sim \mathsf{N}(0, \mathbf{I})$, we can compute the expectation explicitly to obtain

$$\hat{f}(\mathbf{k}, t) = -\frac{i}{\sigma_t} (\nabla_{\mathbf{k}} e^{-\frac{1}{2} \sigma_t^2 |\mathbf{k}|^2}) \tag{29}$$

$$= i \sigma_t \mathbf{k} e^{-\frac{1}{2} \sigma_t^2 |\mathbf{k}|^2} \tag{30}$$

Since $\mathbf{x}_*$ and $\boldsymbol{\varepsilon}$ are independent random variable, we have

$$\mathbb{E}[\boldsymbol{\varepsilon} e^{i \mathbf{k} \cdot \mathbf{x}_t}] = \hat{f}(\mathbf{k}, t) \mathbb{E}[e^{i \alpha_t \mathbf{k} \cdot \mathbf{x}_*}] = i \sigma_t \mathbf{k} \underbrace{e^{-\frac{1}{2} \sigma_t^2 |\mathbf{k}|^2} \mathbb{E}[e^{i \alpha_t \mathbf{k} \cdot \mathbf{x}_*}]}_{\text{combine this}} = i \sigma_t \mathbf{k} \hat{p}_t(\mathbf{k}) \tag{31}$$

where $\hat{p}_t(\mathbf{k})$ is the characteristic function of $\mathbf{x}_t = \alpha_t \mathbf{x}_* + \sigma_t \boldsymbol{\varepsilon}$, defined in Eq. (16). The left hand-side of this equation can also be written as:

$$\mathbb{E}[\boldsymbol{\varepsilon} e^{i \mathbf{k} \cdot \mathbf{x}_t}] = \int_{\mathbb{R}^d} \mathbb{E}[\boldsymbol{\varepsilon} e^{i \mathbf{k} \cdot \mathbf{x}_t} | \mathbf{x}_t = \mathbf{x}] p_t(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{32}$$

$$= \int_{\mathbb{R}^d} \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}] e^{i \mathbf{k} \cdot \mathbf{x}} p_t(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{33}$$

whereas the right hand-side is

$$i \sigma_t \mathbf{k} \hat{p}_t(\mathbf{k}) = i \sigma_t \mathbf{k} \int_{\mathbb{R}^d} e^{i \mathbf{k} \cdot \mathbf{x}} p_t(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{34}$$

$$= \sigma_t \int_{\mathbb{R}^d} \nabla_{\mathbf{x}} [e^{i \mathbf{k} \cdot \mathbf{x}}] p_t(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{35}$$

$$= -\sigma_t \int_{\mathbb{R}^d} e^{i \mathbf{k} \cdot \mathbf{x}} \nabla_{\mathbf{x}} p_t(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{36}$$

$$= -\sigma_t \int_{\mathbb{R}^d} e^{i \mathbf{k} \cdot \mathbf{x}} \mathbf{s}(\mathbf{x}, t) p_t(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{37}$$

where we used integration by parts to get the third equality, and again the definition of the score to get the last. Comparing Eq. (33) and Eq. (37) we deduce that, when $\sigma_t \neq 0$,

$$\mathbf{s}(\mathbf{x}, t) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = -\sigma_t^{-1} \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}] \tag{38}$$

A.4. Proof of Eq. (9)

We note that there exists a straightforward connection between $\mathbf{v}(\mathbf{x}, t)$ and $\mathbf{s}(\mathbf{x}, t)$. From Eq. (1), we have

$$\begin{aligned}
\mathbf{v}(\mathbf{x}, t) &= \dot{\alpha}_t \mathbb{E}[\mathbf{x}_* | \mathbf{x}_t = \mathbf{x}] + \dot{\sigma}_t \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}] && (39)\\
&= \dot{\alpha}_t \mathbb{E}[\frac{\mathbf{x}_t - \sigma_t \boldsymbol{\varepsilon}}{\alpha_t} | \mathbf{x}_t = \mathbf{x}] + \dot{\sigma}_t \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}] && (40)\\
&= \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} + (\dot{\sigma}_t - \frac{\dot{\alpha}_t \sigma_t}{\alpha_t}) \mathbb{E}[\boldsymbol{\varepsilon} | \mathbf{x}_t = \mathbf{x}] && (41)\\
&= \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} + (\dot{\sigma}_t - \frac{\dot{\alpha}_t \sigma_t}{\alpha_t})(-\sigma_t \mathbf{s}(\mathbf{x}, t)) && (42)\\
&= \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} - \lambda_t \sigma_t \mathbf{s}(\mathbf{x}, t) && (43)
\end{aligned}$$

where we defined

$$\lambda_t = \dot{\sigma}_t - \frac{\dot{\alpha}_t \sigma_t}{\alpha_t}. \tag{44}$$

Given Eq. (43) is linear in terms of $\mathbf{s}$, reverting it will lead to Eq. (9).

Note that we can also plug Eq. (43) into the loss $\mathcal{L}_\mathbf{v}$ in Eq. (7) to deduce that

$$\begin{aligned}
\mathcal{L}_\mathrm{v}(\theta) &= \int_0^T \mathbb{E}[\| \underbrace{\frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x}}_{\text{Expand by } \mathbf{x}_t = \alpha_t \mathbf{x}_* + \sigma_t \boldsymbol{\varepsilon}} + \lambda_t(-\sigma_t \mathbf{s}_\theta(\mathbf{x}_t, t)) - \dot{\alpha}_t \mathbf{x}_* - \dot{\sigma}_t \boldsymbol{\varepsilon}\|^2] \mathrm{d}t && (45)\\
&= \int_0^T \mathbb{E}[\|\dot{\alpha}_t \mathbf{x}_* + \frac{\dot{\alpha}_t \sigma_t}{\alpha_t}\boldsymbol{\varepsilon} + \lambda_t(-\sigma_t \mathbf{s}_\theta(\mathbf{x}_t, t)) - \dot{\alpha}_t \mathbf{x}_* - \dot{\sigma}_t \boldsymbol{\varepsilon}\|^2] \mathrm{d}t && (46)\\
&= \int_0^T \mathbb{E}[\|\lambda_t(-\sigma_t \mathbf{s}_\theta(\mathbf{x}_t, t)) - \lambda_t \boldsymbol{\varepsilon}\|^2] \mathrm{d}t && (47)\\
&= \int_0^T \lambda_t^2 \mathbb{E}[\|\sigma_t \mathbf{s}_\theta(\mathbf{x}_t, t) + \boldsymbol{\varepsilon}\|^2] \mathrm{d}t && (48)\\
&\equiv \mathcal{L}_{\mathrm{s}_\lambda}(\theta) && (49)
\end{aligned}$$

which defines the weighted score objective $\mathcal{L}_{\mathrm{s}_\lambda}(\theta)$. This observation is consistent with the claim made in Kingma and Gao [29] that the score objective with different monotonic weighting functions coincides with losses for different model parameterizations. In Appendix B we show that λ_t corresponds to the square of the maximum likelihood weighting proposed in Song et al. [53] and Vahdat et al. [56].

B. Connection with Score-based Diffusion

As shown in Song et al. [54], the reverse-time SDE from Eq. (10) is

$$\mathrm{d}\mathbf{X}_t = [-\frac{1}{2}\beta_t \mathbf{X}_t - \beta_t \mathbf{s}(\mathbf{X}_t, t)]\mathrm{d}t + \sqrt{\beta_t}\mathrm{d}\bar{\mathbf{W}}_t \tag{50}$$

Let us show this SDE is Eq. (4) for the specific choice $w_t = \beta_t$. To this end, notice that the solution $\mathbf{X}_t$ to Eq. (50) for the initial condition $\mathbf{X}_{t=0} = \mathbf{x}_*$ with $\mathbf{x}_*$ fixed is Gaussian distributed with mean and variance given respectively by

$$\begin{aligned}
\mathbb{E}[\mathbf{X}_t] &= e^{-\frac{1}{2}\int_0^t \beta_s \mathrm{d}s}\mathbf{x}_* \equiv \alpha_t \mathbf{x}_* && (51)\\
\mathrm{var}[\mathbf{X}_t] &= 1 - e^{-\int_0^t \beta_s \mathrm{d}s} \equiv \sigma_t^2 && (52)
\end{aligned}$$

Using Eq. (43), the velocity of the score-based diffusion model can therefore be expressed as

$$\begin{aligned}
\mathbf{v}(\mathbf{x}, t) &= -\frac{1}{2}\beta_t \mathbf{x} + (-\frac{1}{2}\beta_t(1 - e^{-\int_0^t \beta_s \mathrm{d}s}) - \frac{1}{2}\beta_t e^{-\int_0^t \beta_s \mathrm{d}s})\mathbf{s}(\mathbf{x}, t) && (53)\\
&= -\frac{1}{2}\beta_t \mathbf{x} - \frac{1}{2}\beta_t \mathbf{s}(\mathbf{x}, t) && (54)
\end{aligned}$$

If we plug Eq. (54) into Eq. (4)[2] with $w_t = \beta_t$, we arrive at Eq. (50). In addition we see that $2\lambda_t\sigma_t$ is precisely β_t, making λ_t^2 correspond to the square of maximum likelihood weighting.

A useful observation for choosing velocity versus noise model. We see that in the velocity model, all of the path-dependent terms (α_t, σ_t) are inside the squared loss, and in the score model, the terms are pulled out (apart from the necessary σ_t in score matching loss) and get squared due to coming out of the norm. So which is better depends on the interpolant. In the paper we see that for SBDM-VP, due to the blowing up behavior of $\dot{\sigma}_t$ near $t = 0$, both $\mathcal{L}_{\text{v}}$ and $\mathcal{L}_{\text{s}_\lambda}$ are unstable.

Yet, shown in Tab. 5, we observed better performance with $\mathcal{L}_{\text{s}_\lambda}$ for SBDM-VP, as the blowing up λ_t near $t = 0$ will compensate for the diminishing gradient inside the squared norm, where $\mathcal{L}_{\text{v}}$ would simply experience gradient explosion resulted from $\dot{\sigma}_t$. The behavior is different for the LIN and GVP interpolant, where the source of instability is α_t^{-1} near $t = 1$. We note $\mathcal{L}_{\text{v}}$ is stable since α_t^{-1} gets cancelled out inside the squared norm, while in $\mathcal{L}_{\text{s}_\lambda}$ it remains in λ_t outside the norm.

C. Sampling with Guidance

Let $p_t(\mathbf{x}|\mathbf{y})$ be the density of $\mathbf{x}_t = \alpha_t\mathbf{x}_* + \sigma_t\boldsymbol{\varepsilon}$ conditioned on some extra variable $\mathbf{y}$. By argument similar to the one given in Appendix A.1, it is easy to see that $p_t(\mathbf{x}|\mathbf{y})$ satisfies the transport equation (compare Eq. (24))

$$\partial_t p_t(\mathbf{x}|\mathbf{y}) + \nabla_{\mathbf{x}} \cdot (\mathbf{v}(\mathbf{x}, t|\mathbf{y})p_t(\mathbf{x}, |\mathbf{y})) = 0, \tag{55}$$

where (compare Eq. (3))

$$\mathbf{v}(\mathbf{x}, t|\mathbf{y}) = \mathbb{E}[\dot{\mathbf{x}}_t|\mathbf{x}_t = \mathbf{x}, \mathbf{y}] = \dot{\alpha}_t\mathbb{E}[\mathbf{x}_*|\mathbf{x}_t = \mathbf{x}, \mathbf{y}] + \dot{\sigma}_t\mathbb{E}[\boldsymbol{\varepsilon}|\mathbf{x}_t = \mathbf{x}, \mathbf{y}] \tag{56}$$

Proceeding as in Appendix A.3 and Appendix A.4, it is also easy to see that the score $\mathbf{s}(\mathbf{x}, t|\mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y})$ is given by (compare Eq. (5))

$$\mathbf{s}(\mathbf{x}, t|\mathbf{y}) = -\sigma_t^{-1}\mathbb{E}[\boldsymbol{\varepsilon}|\mathbf{x}_t = \mathbf{x}, \mathbf{y}] \tag{57}$$

and that $\mathbf{v}(\mathbf{x}, t|\mathbf{y})$ and $\mathbf{s}(\mathbf{x}, t|\mathbf{y})$ are related via (compare Eq. (43))

$$\mathbf{v}(\mathbf{x}, t|\mathbf{y}) = \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} - \lambda_t\sigma_t\mathbf{s}(\mathbf{x}, t|\mathbf{y}) \tag{58}$$

Consider now

$$\begin{aligned}
\mathbf{s}^\zeta(\mathbf{x}, t|\mathbf{y}) &\equiv (1-\zeta)\mathbf{s}(\mathbf{x}, t) + \zeta\mathbf{s}(\mathbf{x}, t|\mathbf{y}) && (59)\\
&= \nabla \log p_t(\mathbf{x}) - \zeta\nabla \log p_t(\mathbf{x}) + \zeta\nabla \log p_t(\mathbf{x}|\mathbf{y}) && (60)\\
&= \nabla \log p_t(\mathbf{x}) - \zeta\nabla \log p_t(\mathbf{x}) + \Big(\zeta\nabla \log p_t(\mathbf{y}|\mathbf{x}) + \zeta\nabla \log p_t(\mathbf{x})\Big) && (61)\\
&= \nabla \log p_t(\mathbf{x}) + \zeta\nabla \log p_t(\mathbf{y}|\mathbf{x}) && (62)\\
&= \nabla \log[p_t(\mathbf{x})p_t^\zeta(\mathbf{y}|\mathbf{x})] && (63)
\end{aligned}$$

where we have used the fact $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ that follows from $p_t(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p_t(\mathbf{y}|\mathbf{x})p_t(\mathbf{x})$, and ζ to be some constant greater than 1. Eq. (63) shows that using the score mixture $\mathbf{s}^\zeta(\mathbf{x}, t|\mathbf{y}) = (1-\zeta)\mathbf{s}(\mathbf{x}, t) + \zeta\mathbf{s}(\mathbf{x}, t|\mathbf{y})$, and the velocity mixture associated with it, namely,

$$\begin{aligned}
\mathbf{v}^\zeta(\mathbf{x}, t|\mathbf{y}) &= (1-\zeta)\mathbf{v}(\mathbf{x}, t) + \zeta\mathbf{v}(\mathbf{x}, t|\mathbf{y}) && (64)\\
&= \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} - \lambda_t\sigma_t[(1-\zeta)\mathbf{s}(\mathbf{x}, t) + \zeta\mathbf{s}(\mathbf{x}, t|\mathbf{y})] && (65)\\
&= \frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x} - \lambda_t\sigma_t\mathbf{s}^\zeta(\mathbf{x}, t|\mathbf{y}), && (66)
\end{aligned}$$

allows one to to construct generative models that sample the tempered distribution $p_t(\mathbf{x}_t)p_t^\zeta(\mathbf{y}|\mathbf{x}_t)$ following classifier guidance [15]. Note that $p_t(\mathbf{x})p_t^\zeta(\mathbf{y}|\mathbf{x}) \propto p_t^\zeta(\mathbf{x}|\mathbf{y})p_t^{1-\zeta}(\mathbf{x})$, so we can also perform classifier free guidance sampling [19]. Empirically, we observe significant performance boost by applying classifier free guidance, as showed in Tab. 1.

[2]We note this SDE is defined in forward time; for reverse time we have $\mathrm{d}\mathbf{X}_t = \mathbf{v}(\mathbf{X}_t, t)\mathrm{d}t - \frac{1}{2}w_t\mathbf{s}(\mathbf{X}, t)\mathrm{d}t + \sqrt{w_t}\mathrm{d}\bar{\mathbf{W}}_t$

Model	Training Steps(K)	FID↓	sFID↓	IS↑	Precision↑	Recall↑
SiT-S	400	58.97 / 57.64	8.95 / 9.05	23.34 / 24.78	0.40 / 0.41	0.59 / 0.60
SiT-B	400	34.64 / 33.45	6.59 / 6.46	41.53 / 43.71	0.52 / 0.53	0.64 / 0.63
SiT-L	400	20.01 / 18.79	5.31 / 5.29	67.76 / 72.02	0.62 / 0.64	0.64 / 0.64
SiT-XL	400	18.04 / 17.19	5.17 / 5.07	73.90 / 76.52	0.63 / 0.65	0.64 / 0.63
SiT-XL	7000	9.35 / 8.61	6.38 / 6.32	126.06 / 131.65	0.67 / 0.68	0.68 / 0.67

Table 1. **FID-50K scores produced by ODE and SDE.** We demonstrate the comparison between ODE and SDE across all of our model sizes. All statistics are produced without classifier free guidance. Each cell in the table is showing [ODE results] / [SDE results]. We note the better performances of SDE observed in all model sizes are in line with the bounds given in [2], and that ODE has its advantage in lower NFE region, as shown in Fig. 5

D. Sampling with ODE and SDE

In the main body of the paper, we used a second order Heun integrator for solving the ODE in Eq. (2) and a first order Euler-Maruyama integrator for solving the SDE in Eq. (4). We summarize all results in Tab. 1, and present the implementations below.

Algorithm 1 Deterministic Heun Sampler

procedure HEUNSAMPLER($\mathbf{v}_\theta(\mathbf{x}, t, \mathbf{y}), t_{i\in\{0,\cdots,N\}}, \alpha_t, \sigma_t$)
  sample $\mathbf{x}_0 \sim \mathsf{N}(0, \mathbf{I})$ ▷ Generate initial sample
  $\Delta t \leftarrow t_1 - t_0$ ▷ Determine fixed step size
  for $i \in \{0, \cdots, N-1\}$ **do**
    $\mathbf{d}_i \leftarrow \mathbf{v}_\theta(\mathbf{x}_i, t_i, \mathbf{y})$
    $\tilde{\mathbf{x}}_{i+1} \leftarrow \mathbf{x}_i + \Delta t \mathbf{d}_i$ ▷ Euler Step at t_i
    $\mathbf{d}_{i+1} \leftarrow \mathbf{v}_\theta(\tilde{\mathbf{x}}_{i+1}, t_{i+1}, \mathbf{y})$
    $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \frac{\Delta t}{2}[\mathbf{d}_i + \mathbf{d}_{i+1}]$ ▷ Explicit trapezoidal rule at t_{i+1}
  end for
  return $\mathbf{x}_N$
end procedure

Algorithm 2 Stochastic Euler-Maruyama Sampler

procedure EULERSAMPLER($\mathbf{v}_\theta(\mathbf{x}, t, \mathbf{y}), w_t, t_{i\in\{0,\cdots,N\}}, T, \alpha_t, \sigma_t$)
  sample $\mathbf{x}_0 \sim \mathsf{N}(0, \mathbf{I})$ ▷ Generate initial sample
  $\mathbf{s}_\theta \leftarrow$ **convert** from $\mathbf{v}_\theta$ following Appendix A.4 ▷ Obtain $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ in Eq. (4)
  $\Delta t \leftarrow t_1 - t_0$ ▷ Determine fixed step size
  for $i \in \{0, \cdots, N-1\}$ **do**
    sample $\varepsilon_i \sim \mathsf{N}(0, \mathbf{I})$
    $\mathrm{d}\varepsilon_i \leftarrow \varepsilon_i * \sqrt{\Delta t}$
    $\mathbf{d}_i \leftarrow \mathbf{v}_\theta(\mathbf{x}_i, t_i, \mathbf{y}) + \frac{1}{2} w_{t_i} \mathbf{s}_\theta(\mathbf{x}_i, t_i, \mathbf{y})$ ▷ Evaluate drift term at t_i
    $\bar{\mathbf{x}}_{i+1} \leftarrow \mathbf{x}_i + \Delta t \mathbf{d}_i$
    $\mathbf{x}_{i+1} \leftarrow \bar{\mathbf{x}}_{i+1} + \sqrt{w_{t_i}} \mathrm{d}\varepsilon_i$ ▷ Evaluate diffusion term at t_i
  end for
  $h \leftarrow T - t_N$ ▷ Last step size; T denotes the time where $\mathbf{x}_T = \mathbf{x}_*$
  $\mathbf{d} \leftarrow \mathbf{v}_\theta(\mathbf{x}_N, t_N, \mathbf{y}) + \frac{1}{2} w_{t_N} \mathbf{s}_\theta(\mathbf{x}_N, t_N, \mathbf{y})$
  $\mathbf{x} \leftarrow \mathbf{x}_N + h * \mathbf{d}$ ▷ Last step; output noiseless sample without diffusion
  return $\mathbf{x}$
end procedure

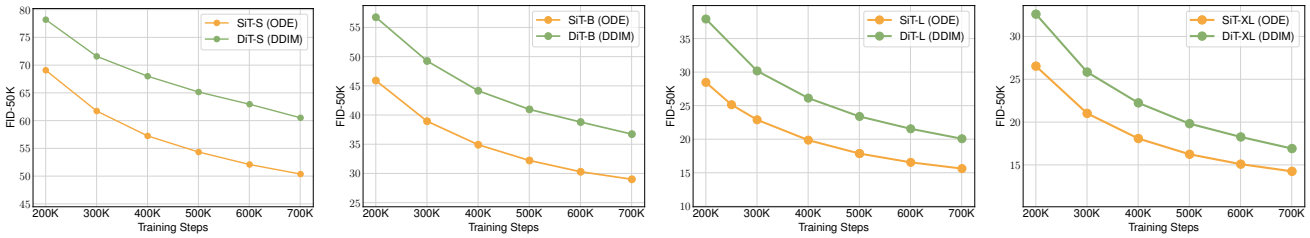

Figure 1. **SiT observes improvement in FID across all model sizes.** We show FID-50K over training iterations for both DiT and SiT models. Across all model sizes, SiT converges faster. We acknowledge this is *not directly an apples-to-apples comparison*. This is because DDIM is essentially a discrete form of the first-order Euler's method, whereas in sampling SiT, we employ the second-order Heun's method. Nevertheless, both the SiT and DiT results are produced by a deterministic sampler with a 250 NFE.

It is feasible to use either a velocity model $\mathbf{v}_\theta$ or a score model $\mathbf{s}_\theta$ in applying the above two samplers. If learning the score for the deterministic Heun sampler, we could always convert the learned $\mathbf{s}_\theta$ to $\mathbf{v}_\theta$ following Appendix A.4. However, as there exists potential numerical instability (depending on interpolants) in $\dot{\sigma}_t$, α_t^{-1} and λ_t, it's recommended to learn $\mathbf{v}_\theta$ in sampling with deterministic sampler instead of $\mathbf{s}_\theta$. For the stochastic sampler, it's required to have both $\mathbf{v}_\theta$ and $\mathbf{s}_\theta$ in integration, so we always need to convert from one (either learning velocity or score) to obtain the other. Under this scenario, the numerical issue from Appendix A.4 can only be avoided by clipping the time interval near $t = 0$. Empirically we found clipping the interval by $h = 0.04$ and doing a long last step from $t = 0.04$ to 0 can greatly benefit the performance. A detailed summary of sampler configuration is provided in Appendix E.

Additionally, we could replace $\mathbf{v}_\theta$ and $\mathbf{s}_\theta$ by $\mathbf{v}_\theta^\zeta$ and $\mathbf{s}_\theta^\zeta$ presented in Appendix C as inputs of the two samplers and enjoy the performance improvements coming along with guidance. As guidance requires evaluating both conditional and unconditional model output in a single step, it will impose twice the computational cost when sampling.

We also note that our models are compatible with more advanced samplers [30, 37]. We do not include the evaluations of those samplers in our work for the sake of direct comparison with the DDPM model, and we leave the investigation of potential performance improvements to future work.

Comparison between DDPM and Euler-Maruyama We primarily investigate and report the performance comparison between DDPM and Euler-Maruyama samplers. We set our Euler sampler's number of steps to be 250 to match that of DDPM during evaluation. This comparison is made direct and fair, as the DDPM method can be viewed as a discretized version of Euler's method.

Comparison between DDIM and Heun We also investigate the performance difference produced by deterministic samplers between DiT and our models. In Fig. 1, we show the FID-50K results for both DiT models sampled with DDIM and SiT models sampled with Heun. We note that this is not directly an apples-to-apples comparison, as DDIM can be viewed as a discretized version of the first order Euler's method, while we use the second order Heun's method in sampling SiT models, due to the large discretization error with Euler's method in continuous time. Nevertheless, we control the NFEs for both DDIM (250 sampling steps) and Heun (250 NFE).

E. Additional Implementation Details

We implemented our models in JAX following the DiT PyTorch codebase by Peebles and Xie [39][3], and referred to Albergo et al. [2][4], Song et al. [54][5], and Dockhorn et al. [16][6] for our implementation of the Euler-Maruyama sampler. For the Heun sampler, we directly used the one from `diffrax` [27][7], a JAX-based numerical differential equation solver library.

Training configurations We trained all of our models following identical structure and hyperparameters retained from DiT [39]. We used AdamW [28, 36] as optimizer for all models. We use a constant learning rate of 1×10^{-4} and a batch size

[3]https://github.com/facebookresearch/DiT
[4]https://github.com/malbergo/stochastic-interpolants
[5]https://github.com/yang-song/score_sde
[6]https://github.com/nv-tlabs/CLD-SGM
[7]https://github.com/patrick-kidger/diffrax

of 256. We used random horizontal flip with probability of 0.5 in data augmentation. *We did not tune the learning rates, decay/warm up schedules, AdamW parameters, nor use any extra data augmentation or gradient clipping during training*. Our largest model, SiT-XL, trains at approximately 6.8 iters/sec on a TPU v4-64 pod following the above configurations. This speed is slightly faster compared to DiT-XL, which trains at 6.4 iters/sec under identical settings.

Sampling configurations We maintain an exponential moving average (EMA) of all models weights over training with a decay of 0.9999. All results are sampled from the EMA checkpoints, which is empirically observed to yield better performance. We summarize the start and end points of our deterministic and stochastic samplers with different interpolants below, where each t_0 and t_N are carefully tuned to optimize performance and avoid numerical instability during integration.

Interpolant	*Model*	*Objective*	Heun		Euler-Maruyama	
			t_0	t_N	t_0	t_N
SBDM-VP	velocity	$\mathcal{L}_{\mathrm{v}}$	1	1e-5	1	4e-2
	score	$\mathcal{L}_{\mathrm{s}_\lambda}$	1	1e-5	1	4e-2
GVP	velocity	$\mathcal{L}_{\mathrm{v}}$	1	0	1	4e-2
	score	$\mathcal{L}_{\mathrm{s}}$	1 - 1e-5	0	1 - 1e-3	4e-2
LIN	velocity	$\mathcal{L}_{\mathrm{v}}$	1	0	1	4e-2
	score	$\mathcal{L}_{\mathrm{s}}$	1 - 1e-5	0	1 - 1e-3	4e-2

Table 2. Sampler configurations

FID calculation We calculate FID scores between generated images (10K or 50K) and all available real images in ImageNet training dataset. We observe small performance variations between TPU-based FID evaluation and GPU-based FID evaluation (ADM's TensorFlow evaluation suite [15][8]). To ensure consistency with the basline DiT, we sample all of our models on GPU and obtain FID scores using the ADM evaluation suite.

[8]https://github.com/openai/guided-diffusion/tree/main/evaluations

F. Additional Visual results

Figure 2. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”macaw”(88)

Figure 3. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”arctic fox”(279)

Figure 4. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”loggerhead turtle”(33)

Figure 5. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”balloon”(417)

CVPR
#1444

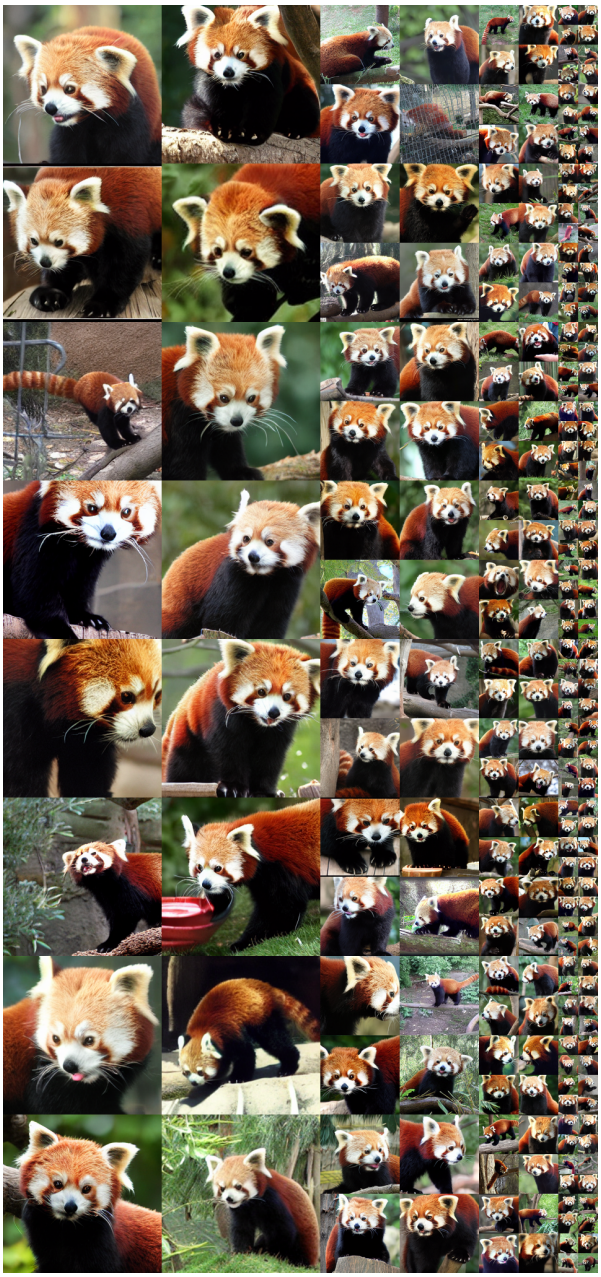

Figure 6. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”red panda”(387)

Figure 7. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”geyser”(974)

Figure 8. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”volcano”(980)

Figure 9. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”golden retriever”(207)

Figure 10. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ” ice cream”(928)

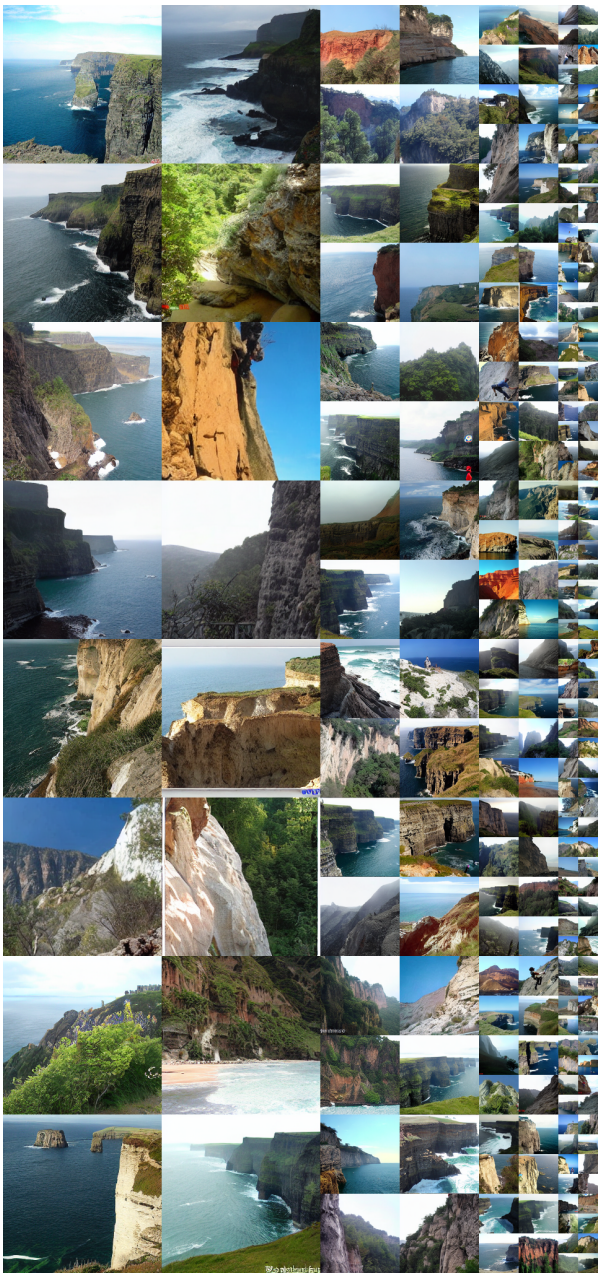

Figure 11. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”cliff”(972)

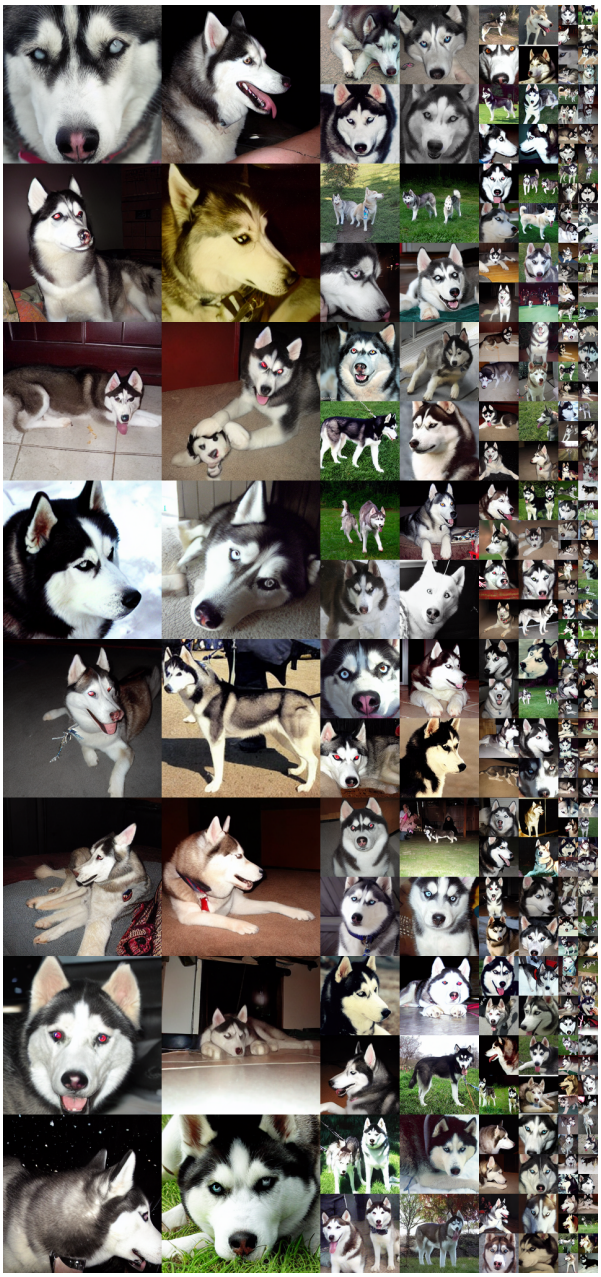

Figure 12. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”husky”(250)

Figure 13. **Uncurated** 256 × 256 **SiT-XL samples**.
Classifier-free guidance scale = 4.0
Class label = ”valley”(979)