SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. While computing time-reversals of SDEs is challenging in general, we show that in this situation the reverse-time SDE depends only on the time-dependent gradient field (score) of the corrupted data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in diffusion probabilistic modeling and score-based generative modeling, and allows for new sampling procedures. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, which enables exact likelihood computation, and improved sampling efficiency. Our framework also enables conditional generation with an unconditional model, as we demonstrate with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.2, and a competitive likelihood of 3.1 bits/dim.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density) at each noise level, and then uses Langevin dynamics to sample from a sequence of decreasing noise levels during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly involves computing scores at each noise level. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models have proven effective at generation of images (Song & Ermon, 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai et al., 2020). However, the practical performance of the two model classes can often be quite different, for reasons that are not understood. More generally, despite obvious surface similarities the relationship between SMLD and DDPM approaches is largely unexplored.

We address these open questions by unifying and generalizing both approaches through the lens of stochastic differential equations (SDEs). Instead of perturbing data with a finite number of noise distributions, we consider a continuum of them, evolving over time according to an Itô process. This process progressively diffuses a data point into random noise, and is given by a prescribed SDE that does not depend on the data. By reversing this process, we can smoothly mold random noise into data for sample generation. Crucially, this reverse process satisfies a reverse-time SDE (Anderson,

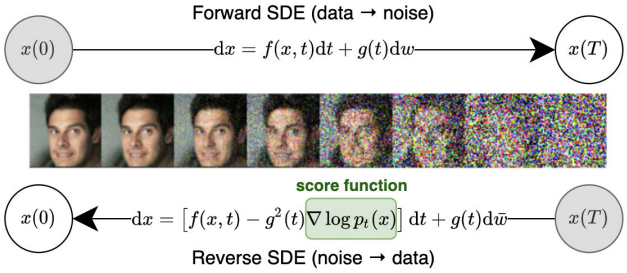

Figure 1: **Solving reverse-time SDEs yields score-based generative models.** Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_x \log p_t(x)$.

1982), which can be derived from the forward SDE as well as the scores of the marginal probability densities as a function of time. We can therefore approximate the reverse-time SDE by training a time-dependent neural network to estimate the scores, and then produce samples using numerical SDE solvers. Our key idea is summarized in Fig. 1.

Our proposed framework yields several theoretical and practical contributions:

Flexible sampling General-purpose SDE solvers can be employed to integrate the reverse-time SDE for sampling. In addition, we propose two special methods that are not applicable to general SDEs: (i) Predictor-Corrector (PC) samplers that combine numerical SDE solvers with score-based MCMC approaches, such as Langevin MCMC (Parisi, 1981; Grenander & Miller, 1994); and (ii) deterministic samplers based on probability flows. The former unifies and improves over existing sampling methods for score-based models. The latter allows for exact likelihood computation, efficient and adaptive sampling via black-box ODE solvers, and uniquely identifiable encoding.

Conditional generation We can generate conditional data samples with an unconditional score-based model, due to special properties of the conditional reverse-time SDE. This enables applications such as class-conditional generation, image inpainting, and colorization with a single time-dependent score-based model.

Unified picture The approaches of SMLD and DDPM can be unified into our framework as discretizations of different SDEs. Although Ho et al. (2020) has reported higher sample quality than Song & Ermon (2019; 2020), we show that with better architectures and our new sampling algorithms, the latter can achieve new state-of-the-art Inception and FID scores on CIFAR-10, as well as high-fidelity generation of 1024×1024 samples. This indicates that either SDE could be advantageous, and should be tuned jointly with other design choices.

2 Background

2.1 Denoising score matching with Langevin dynamics (SMLD)

Let $p_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \triangleq \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2\mathbf{I})$ be a perturbation kernel, and $p_\sigma(\tilde{\mathbf{x}}) \triangleq \int p_{\text{data}}(\mathbf{x}) p_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$, where $p_{\text{data}}(\mathbf{x})$ denotes the data distribution. Consider a sequence of positive noise scales $\sigma_{\text{min}} = \sigma_1 < \sigma_2 < \cdots < \sigma_N = \sigma_{\text{max}}$. Typically, σ_{min} is small enough such that $p_{\sigma_{\text{min}}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$, and σ_{max} is large enough such that $p_{\sigma_{\text{max}}}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma_{\text{max}}^2\mathbf{I})$. Song & Ermon (2019) propose to train a Noise Conditional Score Network (NCSN), denoted by $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$, with a weighted sum of denoising score matching objectives:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})} \left[\left\| \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \right\|_2^2 \right]. \tag{1}$$

Given sufficient data and model capacity, the optimal score-based model $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}, \sigma)$ matches $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$ almost everywhere for $\sigma \in \{\sigma_i\}_{i=1}^N$. For sampling, Song & Ermon (2019) run M steps of Langevin MCMC to get a sample for each $p_{\sigma_i}(\mathbf{x})$ sequentially:

$$\mathbf{x}_i^m = \mathbf{x}_i^{m-1} + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i^{m-1}, \sigma_i) + \sqrt{2\epsilon_i}\mathbf{z}_i^m, \quad m = 1, 2, \cdots, M, \tag{2}$$

where $\epsilon_i > 0$ is the step size, and $\mathbf{z}_i^m$ is standard normal. The above is repeated for $i = N, N-1, \cdots, 1$ in turn with $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{x} \mid \mathbf{0}, \sigma_{\text{max}}^2\mathbf{I})$ and $\mathbf{x}_i^0 = \mathbf{x}_{i+1}^M$ when $i < N$. As $M \to \infty$ and $\epsilon_i \to 0$ for all i, $\mathbf{x}_1^N$ becomes an exact sample from $p_{\sigma_{\text{min}}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$.

2.2 DENOISING DIFFUSION PROBABILISTIC MODELS (DDPM)

Sohl-Dickstein et al. (2015); Ho et al. (2020) consider a sequence of positive noise scales $0 < \beta_1, \beta_2, \cdots, \beta_N < 1$. For each training data point $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$, a discrete diffusion process $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_N\}$ is constructed such that $p(\mathbf{x}_i \mid \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \sqrt{1-\beta_i}\mathbf{x}_{i-1}, \beta_i \mathbf{I})$, with a resulting perturbation kernel $p_{\alpha_i}(\mathbf{x}_i \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_0, (1-\alpha_i)\mathbf{I})$, where $\alpha_i \triangleq \prod_{j=1}^{i}(1-\beta_j)$. The noise scales are prescribed such that $\mathbf{x}_N$ is an approximate sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. A variational diffusion process in the reverse direction is parameterized with $p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1}|\mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i-1}; \frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)), \beta_i \mathbf{I})$, and trained with a weighted variant of the evidence lower bound (ELBO):

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} (1-\alpha_i) \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})} [\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]. \tag{3}$$

After solving Eq. (3) to get the optimal model $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}, i)$, samples can be generated by starting from $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and following the estimated reverse diffusion process as below

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i \mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i)) + \sqrt{\beta_i}\mathbf{z}_i, \quad i = N, N-1, \cdots, 1. \tag{4}$$

We call this method *ancestral sampling*, since it amounts to doing ancestral sampling from the graphical model $\prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)$. The objective Eq. (3) described here is equivalent to L_{simple} in Ho et al. (2020), but we re-write it in a slightly different form to expose more similarity to Eq. (1). Like Eq. (1), Eq. (3) is also a mixture of denoising score matching (Vincent, 2011) objectives, which implies that the optimal model, $\mathbf{s}_{\boldsymbol{\theta}^*}(\tilde{\mathbf{x}}, i)$, matches the score of the perturbed data distribution, $\nabla_{\mathbf{x}} \log \int p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) p_{\text{data}}(\mathbf{x}) \mathrm{d}\mathbf{x}$. Moreover, the i-th mixture weights in Eq. (1) and Eq. (3) are both variances of corresponding perturbation kernels, *i.e.*, $p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})$ and $p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})$.

3 SCORE-BASED GENERATIVE MODELING WITH SDES

Perturbing data with multiple noise scales is key to the success of previous methods. We propose to generalize this idea further to an infinite number of noise scales, such that perturbed data distributions evolve according to an Itô process as the noise intensifies.

When using a total of N noise scales, each perturbation kernel $p_{\sigma_i}(\mathbf{x} \mid \mathbf{x}_0)$ of SMLD corresponds to the marginal distribution of $\mathbf{x}_i$ in the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_i, \quad i = 1, \cdots, N, \tag{5}$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and we have introduced $\sigma_0 = 0$ to simplify the equation. In the limit of $N \to \infty$, the Markov chain $\{\mathbf{x}_i\}_{i=1}^{N}$ becomes a continuous process $\mathbf{x}(t)$, $\{\sigma_i\}_{i=1}^{N}$ becomes a function $\sigma(t)$, and $\mathbf{z}(t)$ becomes a Gaussian process, where we use a continuous time variable $t \in [0, 1]$ for indexing, rather than an integer i. Let $\Delta t = \frac{1}{N-1}$, $\mathbf{x}\left(\frac{i-1}{N-1}\right) = \mathbf{x}_i$, $\sigma\left(\frac{i-1}{N-1}\right) = \sigma_i$, and $\mathbf{z}\left(\frac{i-1}{N-1}\right) = \mathbf{z}_i$. Eq. (5) becomes the following at $t \in \{0, \frac{1}{N-1}, \cdots, \frac{N-2}{N-1}\}$:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \sqrt{\sigma^2(t + \Delta t) - \sigma^2(t)}\, \mathbf{z}(t) \approx \mathbf{x}(t) + \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}\Delta t}\, \mathbf{z}(t),$$

where approximate equality holds when $\Delta t \ll 1$. In the limit of $\Delta t \to 0$, this converges to the SDE

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}, \tag{6}$$

where $\mathbf{w}$ is the standard Wiener process / Brownian motion. Likewise for the perturbation kernels $\{p_{\alpha_i}(\mathbf{x} \mid \mathbf{x}_0)\}_{i=1}^{N}$ of DDPM, the discrete Markov chain is

$$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_i, \quad i = 1, \cdots, N. \tag{7}$$

Let $\beta\left(\frac{i-1}{N-1}\right) = (N-1)\beta_i$. As $N \to \infty$, $\beta(\cdot)$ converges to a function over $[0, 1]$, and Eq. (7) converges to the following SDE (see analysis in Appendix B):

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}. \tag{8}$$

So far, we have demonstrated that the noise perturbations used in SMLD and DDPM both correspond to discretizations of SDEs. The SDE of Eq. (6) always gives a process with exploding variance when $t \to \infty$. In contrast, the SDE of Eq. (8) yields a process with bounded variance. In addition, the process has a constant unit variance for all $t \in [0, \infty)$ when $p(\mathbf{x}(0))$ has a unit variance (proof in Appendix B). Due to this difference, we hereafter refer to Eq. (6) as the Variance Exploding (VE) SDE, and Eq. (8) the Variance Preserving (VP) SDE.

In general, we can consider noise perturbations that evolve according to any SDE of the form:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}. \tag{9}$$

Here $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the *drift* coefficient of $\mathbf{x}(t)$, and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is a scalar function known as the *dispersion* coefficient of $\mathbf{x}(t)$. For simplicity we assume the dispersion coefficient is a scalar (intead of a $d \times d$ matrix) and does not depend on $\mathbf{x}$, but our theory can be generalized to hold in those cases (see Appendix A). In this work we follow the Itô interpretation of SDEs. Under some regularity conditions (Øksendal, 2003), the solution to an SDE is an Itô process $\mathbf{x}(t)$. We denote by $p_t(\mathbf{x})$ the (marginal) probability density of $\mathbf{x}(t)$, and use $p_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$ to denote the transition kernel from time s to time t.

3.1 Generating samples by reversing the SDE

We can use an Itô process to gradually diffuse an unknown distribution p_0, for which we have a dataset of i.i.d. samples, to a known prior distribution p_T, for which we have a tractable form to generate samples efficiently. For example, $p_T \approx \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ for the VE SDE (Eq. (6)), and $p_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$ for the VP SDE (Eq. (8)). By starting from samples of $\mathbf{x}(T) \sim p_T$ and reversing the process, we can obtain samples $\mathbf{x}(0) \sim p_0$. A remarkable result from Anderson (1982) states that the reverse of an Itô process is also an Itô process, running backwards in time and given by the reverse-time SDE:

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{10}$$

where $\bar{\mathbf{w}}$ is a standard Wiener process when time flows backwards from T to 0, and $\mathrm{d}t$ is an infinitesimal negative timestep. Once the score of each marginal distribution, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, is known for all t, we can derive the reverse Itô process from Eq. (10) and simulate it to sample from p_0.

3.2 Estimating scores of the SDE

The score of a distribution can be estimated by training a score-based model on samples with score matching (Hyvärinen, 2005). To estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can train a time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ via a straightforward continuous generalization to Eqs. (1) and (3):

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t \sim \mathcal{U}(0,T)}[\lambda(t)\mathbb{E}_{\mathbf{x}(0)\sim p_0(\mathbf{x})}\mathbb{E}_{\mathbf{x}(t)\sim p_{0t}(\mathbf{x}|\mathbf{x}(0))}[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\right\|_2^2]], \tag{11}$$

where $\lambda : [0, T] \to \mathbb{R}^+$ is a weighting function, and $\mathcal{U}(0, T)$ denotes a uniform distribution over $[0, T]$. With sufficient data and model capacity, score matching ensures that the optimal solution to Eq. (11), denoted by $\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}, t)$, equals $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ for almost all $\mathbf{x}$ and t. Note that Eq. (11) uses denoising score matching, but other score matching objectives, such as sliced score matching (Song et al., 2019a) and finite-difference score matching (Pang et al., 2020) are also applicable here.

In Eq. (11), the expectation with respect to p_0 can be estimated with the empirical average over the dataset. The expectation over $\mathbf{x}(t)$, however, requires sampling from the transition kernel $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. When the drift coefficient of the forward SDE is affine (true for both VE and VB SDEs), the transition kernel is $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); \mathbf{m}_{\mathbf{x}(0)}(t), \boldsymbol{\Sigma}_{\mathbf{x}(0)}(t))$, where $\mathbf{m}_{\mathbf{x}(0)}(t)$ and $\boldsymbol{\Sigma}_{\mathbf{x}(0)}(t)$ are often known in closed-form (Särkkä & Solin, 2019). In this case, we can sample from $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ efficiently by $\mathbf{x}(t) = \mathbf{m}_{\mathbf{x}(0)}(t) + \boldsymbol{\Sigma}_{\mathbf{x}(0)}(t)^{1/2}\mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For more general SDEs, we need to simulate the forward SDE to sample from $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$.

4 Solving the reverse SDE

After training a time-dependent score-based model, we can use it to construct the reverse-time SDE and then solve it with numerical approaches to generate samples from p_0.

Algorithm 1 PC sampling (VE SDE)

1: $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$
2: **for** $i = N-1$ **to** 0 **do**
3: $\quad \mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2)\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, \sigma_{i+1})$
4: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad \mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\,\mathbf{z}$
6: $\quad$ **for** $j = 1$ **to** M **do**
7: $\quad\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
8: $\quad\quad \mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i}\,\mathbf{z}$
9: **return** $\mathbf{x}_0$

Algorithm 2 PC sampling (VP SDE)

1: $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $i = N-1$ **to** 0 **do**
3: $\quad \mathbf{x}'_i \leftarrow (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1)$
4: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad \mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}}\,\mathbf{z}$ Predictor
6: $\quad$ **for** $j = 1$ **to** M **do** Corrector
7: $\quad\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
8: $\quad\quad \mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\,\mathbf{z}$
9: **return** $\mathbf{x}_0$

4.1 General-purpose numerical solvers

Numerical solvers provide approximate trajectories from SDEs. Many general-purpose methods exist for solving SDEs numerically, such as Euler-Maruyama and stochastic Runge-Kutta methods (Kloeden & Platen, 2013), which correspond to different discretizations of the stochastic dynamics. We can apply any of them to the reverse-time SDE for sample generation.

Ancestral sampling—the sampling method of DDPM (Eq. (4))—actually corresponds to one special discretization of the reverse-time VP SDE (Eq. (8)) (see Appendix C). In theory, other discretizations should be able to perform comparably or better. To verify this, we propose to apply the same discretization as the forward SDE to the reverse-time SDE (see Appendix C). The resulting samplers for SMLD and DDPM are given as the "predictor" part in Algorithms 1 and 2, and we call this family of methods the *reverse diffusion sampler*. As shown in Table 1, these sampling methods perform slightly better than the original ancestral sampling for both SMLD and DDPM models on CIFAR-10.

4.2 Predictor-corrector samplers

Unlike generic SDEs, we have additional information that can be used to improve solutions. Since we have a score-based model $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can employ score-based MCMC approaches, such as Langevin MCMC (Parisi, 1981; Grenander & Miller, 1994) or HMC (Neal et al., 2011) to sample from p_t directly, and correct the solution of a numerical SDE solver. At each time step, the numerical SDE solver gives an estimate of the sample at the next time step, playing the role of a "predictor". The score-based MCMC approach corrects the marginal distribution of the estimated sample, playing the role of a "corrector". We therefore name these hybrid sampling algorithms *Predictor-Corrector* (PC) methods.

When using the reverse diffusion sampling method as the predictor, and annealed Langevin MCMC as the corrector, we have Algorithms 1 and 2 for VE and VP SDEs respectively, where $\{\epsilon_i\}_{i=0}^{N-1}$ are step sizes for Langevin dynamics. Algorithms 1 and 2 generalize the original sampling methods of SMLD and DDPM: We can recover annealed Langevin dynamics for SMLD by removing the predictor part, or regain the reverse diffusion sampling method for DDPM by removing the corrector.

We train a score-based model on CIFAR-10 with the architecture in Ho et al. (2020) for both VE and VP perturbations (see Appendix F for additional experimental details). To demonstrate the compatibility of our new sampling algorithms to models trained with old approaches, we train the two models with the original SMLD and DDPM objectives (Eqs. (1) and (3)) with $N = 1000$, instead of our new continuous objective Eq. (11). We compare different samplers in Table 1 using the same number of function evaluations (NFE) for the score-based model, where we also include probability flow, a predictor to be discussed in Section 4.3. Observations include: (i) the reverse diffusion sampling method outperforms the ancestral sampling method; and (ii) PC samplers significantly outperform the corrector-only method, and can improve over predictor-only approaches for most cases without extra computation. In addition, we conduct a qualitative study to demonstrate the benefit of splitting computation between predictors and correctors. As an illustration, we train a score-based model on 256×256 LSUN bedroom (Yu et al., 2015) with the VE SDE using the continuous objective Eq. (11), and show qualitative results in Fig. 2.

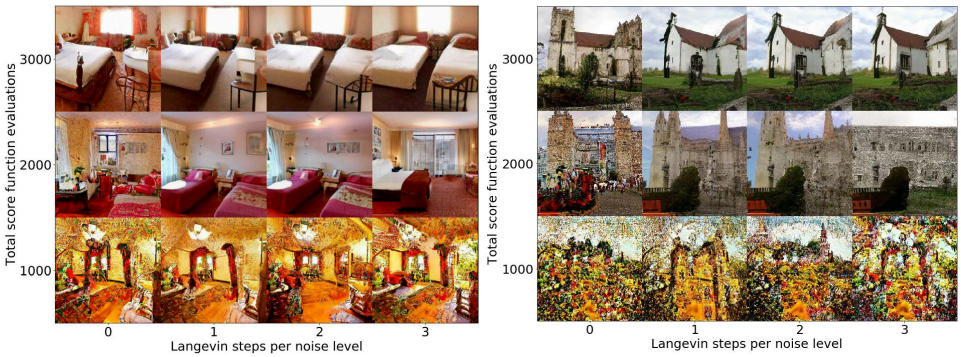

Figure 2: PC sampling for LSUN bedroom and church. The vertical axis corresponds to the total computation, and the horizontal axis represents the amount of computation allocated to the corrector. Samples are the best when computation is split between the predictor and corrector.

Table 1: A comparison of performance resulting from using different methods to sample from score-based generative models. Mean and standard deviation are reported over five sampling runs.

(a) VE SDE (SMLD)

Predictor	FID (w/o corrector)↓	FID (w/ corrector)↓
reverse diffusion	4.79 ± 0.07	3.62 ± 0.02
ancestral sampling	4.98 ± 0.06	3.54 ± 0.01
probability flow	15.41 ± 0.15	3.79 ± 0.04
-	-	39.2 ± 0.1

(b) VP SDE (DDPM)

Predictor	FID (w/o corrector)↓	FID (w/ corrector)↓
reverse diffusion	3.10 ± 0.03	3.18 ± 0.01
ancestral sampling	3.11 ± 0.03	3.21 ± 0.02
probability flow	3.25 ± 0.04	3.06 ± 0.03
-	-	52.92 ± 0.20

4.3 Score-based generative models are equivalent to neural ODEs

Score-based models enable another numerical method for solving the reverse-time SDE. For all Itô processes, there exists a corresponding *deterministic process* whose trajectories induce the same evolution of densities. This deterministic process satisfies an ODE (Maoutsa et al., 2020):

$$\mathrm{d}\mathbf{x} = \Big[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\Big]\mathrm{d}t, \tag{12}$$

which can be determined from the SDE once scores are known. As the score functions are typically parameterized by a neural network, this is an example of a neural ODE (Chen et al., 2018).

As with neural ODEs, we can produce samples by solving Eq. (12) from different final conditions $\mathbf{x}(T)$. Using a fixed discretization strategy we can generate competitive samples, especially when used in conjuction with correctors (Table 1, "probability flow sampler"). Using a black-box ODE solver (Dormand-Prince) allows us to explicitly trade-off accuracy for efficiency. With a larger error tolerance, the number of function evaluations can be reduced by over 90% without effecting the visual quality of samples (Fig. 3).

Exact likelihood computation Leveraging the connection to neural ODEs, we can compute the density defined by Eq. (12) via the instantaneous change of variables formula (Grathwohl et al., 2019). We report negative log-likelihoods (NLLs) measured in bits/dim in Table 2. We compute log-likelihoods on uniformly dequantized data, and only compare to models evaluated in the same way (excluding models evaluated with variational dequantization (Ho et al., 2019) or discrete data). Main results: (i) For the same DDPM model in Ho et al. (2020), we obtain better bits/dim compared to the upper bound given by ELBO, since our likelihoods are exact; (ii) Using the same architecture, we trained another DDPM model with our continuous objective Eq. (11) on uniformly dequantized data. This continuously-trained model (DDPM cont. in Table 2) has much lower bits/dim compared to the original one, setting a new record bits/dim on uniformly dequantized CIFAR-10.

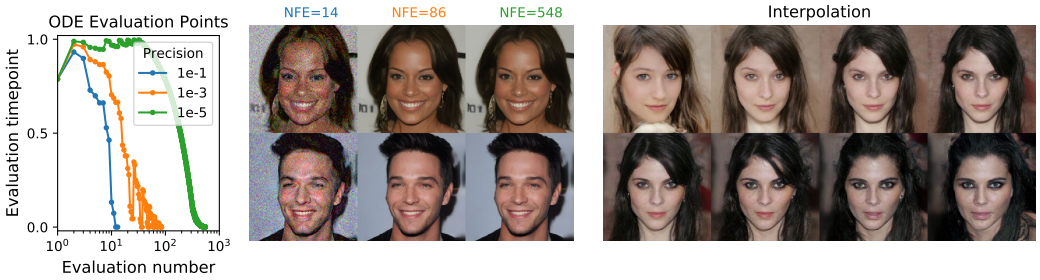

Figure 3: **Probability flow ODE enables efficient sampling** with adaptive step-sizes as the numerical precision is varied (left), and reduces the number of function evaluations (NFE) without harming quality (middle). The determinstic mapping from latents to images allows for interpolations (right).

Table 2: NLLs on CIFAR-10.

Model	NLL Test ↓	FID ↓
Glow (Kingma & Dhariwal, 2018)	3.35	-
MintNet (Song et al., 2019b)	3.32	-
Residual Flow (Chen et al., 2019)	3.28	46.37
FFJORD (Grathwohl et al., 2019)	3.40	-
DDPM (L) (Ho et al., 2020)	$\leqslant$ 3.70	13.51
DDPM (L_{simple}) (Ho et al., 2020)	$\leqslant$ 3.75	**3.17**
DDPM (probability flow)	3.39	3.77
DDPM cont. (probability flow)	**3.10**	4.02

Table 3: CIFAR-10 sample quality.

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	2.67	**10.06**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	3.26	9.74
NCSN (Song & Ermon, 2019)	25.32	8.87 $\pm$ 0.12
DDPM (Ho et al., 2020)	3.17	9.46 $\pm$ 0.11
NCSN++ (4 blocks/resolution)	2.45	9.73
NCSN++ cont. (4 blocks/resolution)	2.38	9.83
NCSN++ cont. (8 blocks/resolution)	**2.20**	**9.89**

Manipulating latent representations By integrating Eq. (12), we can encode any datapoint $\mathbf{x}(0)$ into a latent space $\mathbf{x}(T)$. Decoding can be achieved by integrating a corresponding ODE for the reverse-time SDE. As is done with other invertible models such as neural ODEs and normalizing flows (Dinh et al., 2016; Song et al., 2019b), we can manipulate this latent representation for image editing, such as interpolation, and temperature scaling (see Fig. 3 and Appendix E).

Uniquely identifable encoding Unlike all current invertible models , our representation is *uniquely identifiable*, meaning that with sufficient training data, model capacity, and optimization accuracy, the encoding for an input is uniquely determined by the data distribution (Roeder et al., 2020). This is because our forward SDE, Eq. (9), has no trainable parameters, and its associated probability flow ODE, Eq. (12), provides the same trajectories given perfectly estimated scores.

4.4 Architecture Choices

Although SMLD and DDPM can be viewed as different instantiations of a unified framework, there is a performance gap reported in previous papers. The best FID values of SMLD models on CIFAR-10 is 25.32 (Song & Ermon, 2019), whereas for DDPM it is 3.17 (Ho et al., 2020). With PC samplers and the same model architecture in Ho et al. (2020), the gaps can be reduced, but score-based models trained with the VE SDE still perform slightly worse (see Table 1). This raises the question of whether the variance preserving property of the VP SDE is always preferable or whether there is an interaction with model architecture.

To answer this question, we performed an extensive architecture search (see Appendix G). Our optimal architecture for the VE SDE, dubbed NCSN++, achieves an FID of 2.45 on CIFAR-10, whereas the optimal architecture for the VP SDE over the same architecture sweep obtains an FID of 2.88. This indicates that the VE SDE can be advantageous, and practitioners likely need to experiment with different SDEs for new domains and architectures.

Based on these improved architectures, we were able to further improve FID with the continuous training objective in Eq. (11) (FID 2.38), and increasing the number of residual blocks (FID 2.2, see Table 3 and Appendix G.2). This model sets new records for both inception score and FID on

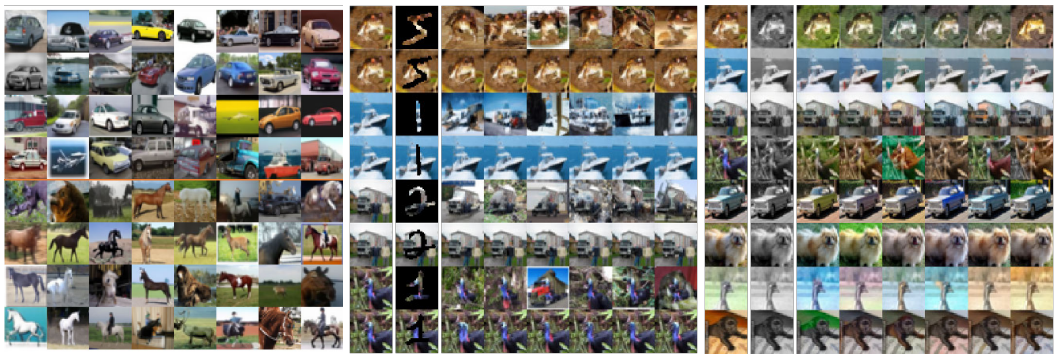

Figure 4: *Left*: Class-conditional samples. Top four rows are automobiles and bottom are horses. *Middle:* Inpainting. First column is the original image, second column is the masked image, remaining columns are sampled image completions. *Right:* Colorization. First column is the original image, second column is the grayscale image, remaining columns are colorizations of the grayscale image.

unconditional generation for CIFAR-10. Surprisingly, we can achieve better FID than the previous best conditional generation model with no labeled data. Finally, with all improvements together, we are able to obtain the first set of high-fidelity samples on CelebA-HQ 1024×1024 from score-based models (see Appendix G.3).

5 Controllable generation

The continuous structure of our framework allows us to not only produce data samples from p_0, but also from $p_0(\mathbf{x}(0) \mid \mathbf{y})$ if $p_t(\mathbf{y} \mid \mathbf{x}(t))$ is known. Given a forward SDE as in Eq. (9), we can sample from $p_t(\mathbf{x}(t) \mid \mathbf{y})$ by starting from $p_T(\mathbf{x}(T) \mid \mathbf{y})$ and solving a conditional reverse-time SDE:

$$\mathrm{d}\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - g(t)^2[\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})]\}\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}} \tag{13}$$

When $\mathbf{y}$ represents class labels, we can train a time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$ for class-conditional sampling. Since the forward SDE is tractable, we can easily create training data $(\mathbf{x}(t), \mathbf{y})$ for the time-dependent classifier by first sampling $(\mathbf{x}(0), \mathbf{y})$ from a dataset, and then sampling $\mathbf{x}(t) \sim p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. Afterwards, we may employ a mixture of cross-entropy losses over different time steps, like Eq. (11), to train the time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$. Class-conditional CIFAR-10 samples can be found in Fig. 4 (left). More details are in Appendix D.

Imputation is a special case of conditional sampling. Suppose we have an incomplete data point $\mathbf{y}$ where only some subset, $\Omega(\mathbf{y})$ is known. Imputation amounts to sampling from $p(\mathbf{x}(0) \mid \Omega(\mathbf{y}))$, which we can accomplish using an unconditional model (see Appendix D.2). Colorization is a special case of imputation, except that the known data dimensions are coupled. We can decouple these data dimensions with an orthogonal linear transformation, and perform imputation in the transformed space (details in Appendix D.3). Fig. 4 shows results for inpainting (middle), and colorization (right) achieved with a single unconditional time-dependent score model.

6 Discussion

We presented a framework for generative modeling by combining SDEs with score-based models. Our work enables a better understanding of existing approaches, several new sampling algorithms, exact likelihood computation, and brings new conditional generation abilities to the family of score-based generative models.

While our proposed sampling approaches improve results and enable more efficient sampling, they remain slower at sampling than GANs on the same datasets. Identifying ways of combining the stable learning of score-based generative models with the fast sampling of implicit models like GANs remains an important research direction.

REFERENCES

Brian D O Anderson. Reverse-time diffusion equation models. *Stochastic Process. Appl.*, 12(3): 313–326, May 1982.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9916–9926, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

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

Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *arXiv preprint arXiv:2009.05475*, 2020.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020a.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020b.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker-planck equations through gradient-log-density estimation. *arXiv preprint arXiv:2006.00702*, 2020.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. volume 108 of *Proceedings of Machine Learning Research*, pp. 4474–4484, Online, 26–28 Aug 2020. PMLR. URL http://proceedings.mlr.press/v108/niu20a.html.

Bernt Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

Tianyu Pang, Kun Xu, Chongxuan Li, Yang Song, Stefano Ermon, and Jun Zhu. Efficient learning of generative models via finite-difference score matching. *arXiv preprint arXiv:2007.03317*, 2020.

Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pp. 14837–14847, 2019.

Geoffrey Roeder, Luke Metz, and Diederik P Kingma. On linear identifiability of learned representations. *arXiv preprint arXiv:2007.00810*, 2020.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, pp. 204, 2019a. URL http://auai.org/uai2019/proceedings/papers/204.pdf.

Yang Song, Chenlin Meng, and Stefano Ermon. Mintnet: Building invertible neural networks with masked convolutions. In *Advances in Neural Information Processing Systems*, pp. 11002–11012, 2019b.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

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

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.

A THE THEORY WITH MORE GENERAL SDES

In the main text, we introduced our framework based on a simplified SDE Eq. (9) where the dispersion is independent of $\mathbf{x}(t)$. It turns out that our framework can be extended to more general forms of the dispersion. Let us consider SDEs in the most universal form:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \mathbf{G}(\mathbf{x}, t)\mathrm{d}\mathbf{w}, \tag{14}$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $\mathbf{G}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$. The reverse-time SDE (Anderson, 1982) is given by (*cf*., Eq. (10))

$$\mathrm{d}\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\}\mathrm{d}t + \mathbf{G}(\mathbf{x}, t)\mathrm{d}\bar{\mathbf{w}}, \tag{15}$$

where we define $\nabla \cdot \mathbf{F}(\mathbf{x}) := (\nabla \cdot \mathbf{f}^1(\mathbf{x}), \nabla \cdot \mathbf{f}^2(\mathbf{x}), \cdots, \nabla \cdot \mathbf{f}^d(\mathbf{x}))^\mathsf{T}$ for a matrix-valued function $\mathbf{F}(\mathbf{x}) := (\mathbf{f}^1(\mathbf{x}), \mathbf{f}^2(\mathbf{x}), \cdots, \mathbf{f}^d(\mathbf{x}))^\mathsf{T}$ throughout the paper. The probability flow ODE (Maoutsa et al., 2020) corresponding to Eq. (14) has the following form (*cf*., Eq. (12)):

$$\mathrm{d}\mathbf{x} = \left\{\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}\nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] - \frac{1}{2}\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right\}\mathrm{d}t. \tag{16}$$

Finally for conditional generation with the general SDE Eq. (14), we can solve the conditional reverse-time SDE below (*cf*., Eq. (13), details in Appendix D):

$$\begin{aligned}\mathrm{d}\mathbf{x} = \{&\mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\ &- \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})\}\mathrm{d}t + \mathbf{G}(\mathbf{x}, t)\mathrm{d}\bar{\mathbf{w}}.\end{aligned} \tag{17}$$

B DERIVATION OF THE VP SDE

For the perturbation kernels $\{p_{\alpha_i}(\mathbf{x} \mid \mathbf{x}_0)\}_{i=1}^N$ used in DDPM, the discrete Markov chain is

$$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_i, \quad i = 1, \cdots, N. \tag{18}$$

In the limit of $N \to \infty$, $\{\beta_i\}_{i=1}^N$ becomes a function $\beta(t)$. Let $\beta\left(\frac{i-1}{N-1}\right) = (N-1)\beta_i$ and $\Delta t = \frac{1}{N-1}$. The above Markov chain becomes the following at $t \in \{0, 1, \cdots, \frac{N-2}{N-1}\}$

$$\mathbf{x}(t + \Delta t) = \sqrt{1 - \beta(t)\Delta t}\, \mathbf{x}(t) + \sqrt{\beta(t)\Delta t}\, \mathbf{z}(t). \tag{19}$$

Therefore, in the limit of $\Delta t \to 0$, Eq. (19) converges to the VP SDE:

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\, \mathrm{d}t + \sqrt{\beta(t)}\, \mathrm{d}\mathbf{w}. \tag{20}$$

Next, we compute the time-evolution of variance in the VP SDE. Since it is a linear SDE, we can use Eq. (4.18) in Särkkä & Solin (2019) to obtain an ODE that governs the evolution of variance

$$\frac{\mathrm{d}\mathbf{\Sigma}(t)}{\mathrm{d}t} = \beta(t)(\mathbf{I} - \mathbf{\Sigma}(t)),$$

where $\mathbf{\Sigma}(t) = \mathrm{Cov}[\mathbf{x}(t)]$. Solving this ODE, we obtain

$$\mathbf{\Sigma}(t) = \mathbf{I} + e^{\int_0^T -\beta(t)\mathrm{d}t}(\mathbf{\Sigma}(0) - \mathbf{I}),$$

from which it is clear that the variance $\mathbf{\Sigma}(t)$ is always bounded given $\mathbf{\Sigma}(0)$. Moreover, $\mathbf{\Sigma}(t) \equiv \mathbf{I}$ if $\mathbf{\Sigma}(0) = \mathbf{I}$.

C REVERSE DIFFUSION SAMPLING

Given a forward SDE

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \mathbf{G}(t)\mathrm{d}\mathbf{w},$$

and suppose the following iteration rule is a corresponding discretization:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i \mathbf{z}_i, \quad i \in \{0, 1, \cdots, N-1\} \tag{21}$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We can apply the same discretization to the reverse-time SDE

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - \mathbf{G}(t)\mathbf{G}(t)^\mathsf{T} \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + \mathbf{G}(t)\mathrm{d}\bar{\mathbf{w}},$$

which will give the following iteration rule for $i \in \{0, 1, \cdots, N-1\}$:

$$\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{f}_{i+1}(\mathbf{x}_{i+1}) + \mathbf{G}_{i+1}\mathbf{G}_{i+1}^\mathsf{T} \nabla_{\mathbf{x}} \log p(\mathbf{x}_{i+1}, t_{i+1}) + \mathbf{G}_{i+1}\mathbf{z}_{i+1}. \tag{22}$$

When applied to Eqs. (5) and (7), we obtain a new set of numerical solvers for the reverse-time VE and VP SDEs, resulting in sampling algorithms as shown in the "predictor" part of Algorithms 1 and 2. We name these sampling methods (that are based on the discretization strategy in Eq. (22)) *reverse diffusion samplers*.

Note that the ancestral sampling of DDPM (Ho et al., 2020) (Eq. (4)) matches its reverse diffusion counterpart when $\beta_i \to 0$ for all i (which amounts to $\Delta t \to 0$), because

$$\begin{aligned}
&\frac{1}{\sqrt{1-\beta_{i+1}}}(\mathbf{x}_i + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)) \approx \left(1 + \frac{1}{2}\beta_{i+1}\right)(\mathbf{x}_i + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)) \\
\approx &\left(1 + \frac{1}{2}\beta_{i+1}\right)\mathbf{x}_i + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i) = \left[2 - \left(1 - \frac{1}{2}\beta_{i+1}\right)\right]\mathbf{x}_i + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i) \\
\approx &(2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_i + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i).
\end{aligned}$$

Therefore, the original ancestral sampler of Eq. (4) is essentially a different discretization to the same reverse-time SDE. This unifies the sampling method in Ho et al. (2020) as a numerical solver to the reverse-time VP SDE in our continuous framework.

D CONTROLLABLE GENERATION

Consider a forward SDE with the following general form

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \mathbf{G}(\mathbf{x}, t)\mathrm{d}\mathbf{w},$$

and suppose the initial state distribution is $p_0(\mathbf{x}(0) \mid \mathbf{y})$. The density at time t is $p_t(\mathbf{x}(t) \mid \mathbf{y})$ when conditioned on $\mathbf{y}$. Therefore, using Anderson (1982), the reverse-time SDE is given by

$$\mathrm{d}\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T} \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y})\}\mathrm{d}t + \mathbf{G}(\mathbf{x}, t)\mathrm{d}\bar{\mathbf{w}}. \tag{23}$$

The score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y})$ can be computed easily since $p_t(\mathbf{x}(t) \mid \mathbf{y}) \propto p_t(\mathbf{x}(t)) p(\mathbf{y} \mid \mathbf{x}(t))$ and

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}(t)). \tag{24}$$

This subsumes the conditional reverse-time SDE in Eq. (13) as a special case.

D.1 CLASS-CONDITIONAL SAMPLING

When $\mathbf{y}$ represents class labels, we can train a time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}_t)$ for class-conditional sampling. Since the forward SDE is tractable, we can easily create a pair of training data $(\mathbf{x}_t, \mathbf{y})$ by first sampling $(\mathbf{x}_0, \mathbf{y})$ from a dataset and then obtaining $\mathbf{x}_t \sim p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$. Afterwards, we may employ a mixture of cross-entropy losses over different time steps, like Eq. (11), to train the time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}_t)$.

To test this idea, we trained a Wide ResNet (Zagoruyko & Komodakis, 2016) (`Wide-ResNet-28-10`) on CIFAR-10 with VE perturbations. The classifier is conditioned on $\log \sigma_i$ using random Fourier features (Tancik et al., 2020), and the training objective is a simple unweighted mixture of cross-entropy losses sampled at different scales. We provide a plot to show the accuracy of this classifier over noise scales in Fig. 5. The score-based model is an unconditional NCSN++ (4 blocks/resolution) in Table 3, and we generate samples using the PC algorithm with 2000 discretization steps. The class-conditional samples are provided in Fig. 4, and an extended set of conditional samples is given in Fig. 5.

Figure 5: Class-conditional image generation by solving the conditional reverse-time SDE with PC. The curve shows the accuracy of our noise-conditional classifier over different noise scales.

D.2 IMPUTATION

Imputation is a special case of conditional sampling. Denote by $\Omega(\mathbf{x})$ and $\bar{\Omega}(\mathbf{x})$ the known and unknown dimensions of $\mathbf{x}$ respectively, and let $\mathbf{f}_\Omega(\cdot, t)$ and $\mathbf{G}_\Omega(\cdot, t)$ denote $\mathbf{f}(\cdot, t)$ and $\mathbf{G}(\cdot, t)$ restricted to the known dimensions. For imputation, our goal is to sample from $p(\bar{\Omega}(\mathbf{x}(0)) \mid \Omega(\mathbf{x}(0)) = \mathbf{y})$. Define a new Itô process $\mathbf{z}(t) = \Omega(\mathbf{x}(t))$, and note that the SDE for $\mathbf{z}(t)$ can be written as

$$\mathrm{d}\mathbf{z} = \mathbf{f}_\Omega(\mathbf{z}, t)dt + \mathbf{G}_\Omega(\mathbf{z}, t)\mathrm{d}\mathbf{w}.$$

The reverse-time SDE, conditioned on $\Omega(\mathbf{x}(0)) = \mathbf{y}$, is

$$\begin{aligned}\mathrm{d}\mathbf{z} = \{&\mathbf{f}_\Omega(\mathbf{z}, t) - \nabla \cdot [\mathbf{G}_\Omega(\mathbf{z}, t)\mathbf{G}_\Omega(\mathbf{z}, t)^\mathsf{T}] \\ &- \mathbf{G}_\Omega(\mathbf{z}, t)\mathbf{G}_\Omega(\mathbf{z}, t)^\mathsf{T}\nabla_{\mathbf{z}} \log p_t(\mathbf{z} \mid \Omega(\mathbf{z}(0)) = \mathbf{y})\}\mathrm{d}t + \mathbf{G}_\Omega(\mathbf{z}, t)\mathrm{d}\bar{\mathbf{w}}.\end{aligned}$$

Although $p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}_0) = \mathbf{y})$ is in general intractable, it can be approximated by

$$\begin{aligned}p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}_0) = \mathbf{y}) &= \int p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)), \mathbf{y}) p_t(\Omega(\mathbf{x}(t)) \mid \mathbf{y}) \mathrm{d}\Omega(\mathbf{x}(t)) \\ &= \mathbb{E}_{p_t(\Omega(\mathbf{x}(t))|\mathbf{y})}[p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)), \mathbf{y})] \\ &\approx \mathbb{E}_{p_t(\Omega(\mathbf{x}(t))|\mathbf{y})}[p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)))] \\ &\approx p_t(\mathbf{z}(t) \mid \Omega'(\mathbf{x}(t))),\end{aligned}$$

where $\Omega'(\mathbf{x}(t))$ is a random sample from $p_t(\Omega(\mathbf{x}(t)) \mid \mathbf{y})$, which is typically a tractable distribution. Therefore,

$$\begin{aligned}\nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y}) &\approx \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \Omega'(\mathbf{x}(t))) \\ &= \nabla_{\mathbf{z}} \log p_t([\mathbf{z}(t); \Omega'(\mathbf{x}(t))]),\end{aligned}$$

where $[\mathbf{z}(t); \Omega'(\mathbf{x}(t))]$ denotes a vector $\mathbf{u}(t)$ such that $\Omega(\mathbf{u}(t)) = \Omega'(\mathbf{x}(t))$ and $\bar{\Omega}(\mathbf{u}(t)) = \mathbf{z}(t)$, and the identity holds because $\nabla_{\mathbf{z}} \log p_t(\Omega'(\mathbf{x}(t))) = 0$.

D.3 COLORIZATION

Colorization is a special case of imputation, except that the known data dimensions are coupled. We can decouple these data dimensions by using an orthogonal linear transformation to map the gray-scale image to a separate channel in a different space, and then perform imputation to complete the other channels before transforming everything back to the original image space. The orthogonal matrix we used to decouple color channels is

$$\begin{pmatrix} 0.577 & -0.816 & 0 \\ 0.577 & 0.408 & 0.707 \\ 0.577 & 0.408 & -0.707 \end{pmatrix}.$$

Because the transformations are all orthogonal matrices, the standard Wiener process $\mathbf{w}(t)$ will still be a valid Wiener process in the transformed space, allowing us to build an SDE and use the same imputation algorithm in Appendix D.2.

E PROBABILITY FLOW SAMPLING

For producing figures in Fig. 3, we use a DDPM model trained on 256×256 CelebA-HQ with the same settings in Ho et al. (2020). We use the RK45 ODE solver in all cases. The bits/dim values in Table 2 are computed with `atol=1e-5` and `rtol=1e-5`, same as Grathwohl et al. (2019). To give the results for DDPM (probability flow) and DDPM cont. (probability flow) in Table 2, we average the bits/dim obtained on the test dataset over five different runs. The standard deviation is below 0.005 for both of them. FID scores are computed on samples from the probability flow sampler.

Aside from the interpolation results in Fig. 3, we demonstrate more examples of latent space manipulation in Fig. 6, including interpolation and temperature scaling. The model tested here is a DDPM model trained with the same settings as in Ho et al. (2020).

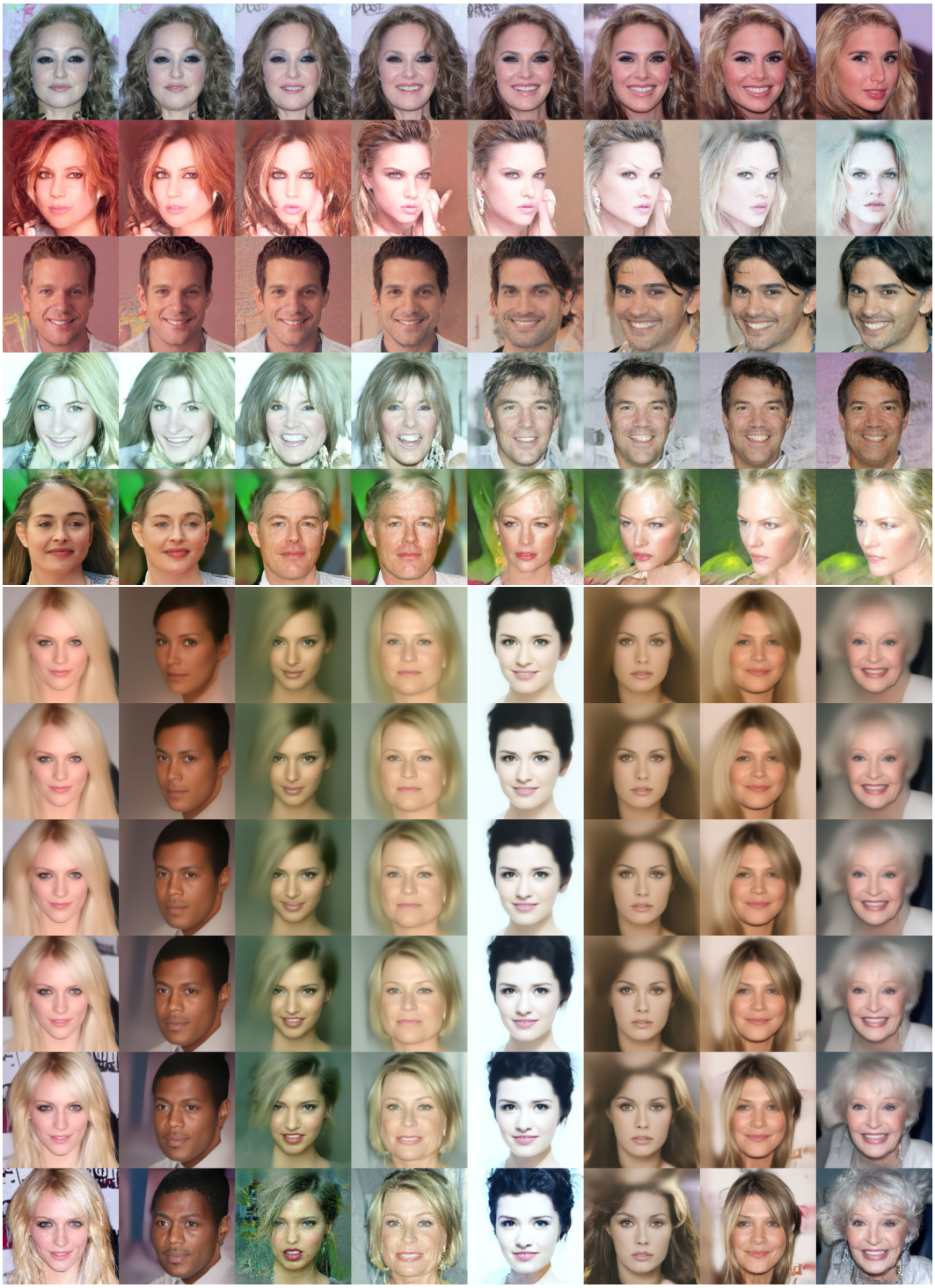

Figure 6: Samples from the probability flow ODE on 256×256 CelebA-HQ with VP perturbations. Top: spherical interpolations between random samples. Bottom: temperature rescaling (reducing norm of embedding).

F ADDITIONAL EXPERIMENTAL DETAILS

SDEs in the wild In SMLD, the noise scales $\{\sigma_i\}_{i=1}^N$ are often chosen to be a geometric sequence, while in DDPM, $\{\beta_i\}_{i=1}^N$ is typically an arithmetic sequence. This corresponds to the following two instantiations of the VE and VP SDEs respectively:

$$\mathrm{d}\mathbf{x} = \sigma_{\text{min}}\left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}\right)^t \sqrt{2\log\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}}\mathrm{d}\mathbf{w} \tag{25}$$

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}(\bar{\beta}_{\text{min}} + t(\bar{\beta}_{\text{max}} - \bar{\beta}_{\text{min}}))\mathbf{x}\mathrm{d}t + \sqrt{\bar{\beta}_{\text{min}} + t(\bar{\beta}_{\text{max}} - \bar{\beta}_{\text{min}})}\mathrm{d}\mathbf{w}, \tag{26}$$

where $\bar{\beta}_{\text{min}} = (N-1)\beta_{\text{min}}$ and $\bar{\beta}_{\text{max}} = (N-1)\beta_{\text{max}}$. In all experiments, we set $\sigma_{\text{min}} = 0.01$ and choose σ_{max} according to Technique 1 in Song & Ermon (2020). For the VP SDE, we let $\bar{\beta}_{\text{min}} = 0.1$ and $\bar{\beta}_{\text{max}} = 20$, which corresponds to the settings in Ho et al. (2020). When training score-based models with the continuous objective Eq. (11), we always use $\lambda(t) = \mathrm{Var}(p_t(\mathbf{x}(t) \mid \mathbf{x}(0)))$.

The corrector algorithm We take the schedule of annealed Langevin dynamics in Song & Ermon (2019), but modify it slightly in order to achieve better empirical performance. The corrector-only samplers for VE/VP SDEs used in our experiments are given in Algorithms 3 and 4, where we call r the "signal-to-noise" ratio. When sampling a batch of samples, we replace the norm $\|\cdot\|_2$ with the average norm across the batch.

Algorithm 3 Corrector algorithm for the VE SDE.

Require: $\{\sigma_i\}_{i=1}^N, r, N, M.$
1: $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{max}}^2\mathbf{I})$
2: **for** $i \leftarrow N$ to 1 **do**
3: **for** $j \leftarrow 1$ to M **do**
4: $\mathbf{z}_i^j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\mathbf{g} \leftarrow \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i^{j-1}, \sigma_i)$
6: $\epsilon \leftarrow 2(r\|\mathbf{z}_i^j\|_2 / \|\mathbf{g}\|_2)^2$
7: $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^{j-1} + \epsilon\,\mathbf{g} + \sqrt{2\epsilon}\,\mathbf{z}_i^j$
8: $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$
return $\mathbf{x}_{i-1}^0$

Algorithm 4 Corrector algorithm for the VP SDE.

Require: $\{\beta_i\}_{i=1}^N, \{\alpha_i\}_{i=1}^N, r, N, M.$
1: $\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: $\mathbf{x}_N^0 \leftarrow \mathbf{x}_N^0/\sqrt{\alpha_N}$
3: **for** $i \leftarrow N$ to 1 **do**
4: **for** $j \leftarrow 1$ to M **do**
5: $\mathbf{z}_i^j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: $\mathbf{g} \leftarrow \sqrt{\alpha_i}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i^{j-1}, i)$
7: $\epsilon \leftarrow 2(r\|\mathbf{z}_i^j\|_2 / \|\mathbf{g}\|_2)^2$
8: $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^{j-1} + \epsilon\,\mathbf{g} + \sqrt{2\epsilon}\,\mathbf{z}_i^j$
9: $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$
return $\mathbf{x}_{i-1}^0$

Denoising For both SMLD and DDPM models, the generated samples typically contain small noise that is hard to detect by humans. As noted by Jolicoeur-Martineau et al. (2020), FIDs can be significantly worse without removing this noise. This unfortunate sensitivity to noise is also part of the reason why NCSN models trained with SMLD has been performing worse than DDPM models in terms of FID, because the former does not use a denoising step at the end of sampling, while the

latter does. In all experiments of this paper we ensure there is a denoising step at the end of sampling, using Tweedie's formula (Efron, 2011).

Additional setups for PC samplers In Table 1, all samplers with the same type of SDEs use comparable amount of computation. For VE SDEs, we use a total of 1000 score-based model evaluations, while for VP SDE we use 2000. We search over the best signal-to-noise ratio for all samplers involving a corrector. For qualitative results in Fig. 2 we use an NCSN++ model trained with the continuous objective on 256×256 LSUN bedroom and church.

G ARCHITECTURE SEARCH FOR MODELS WITH VE PERTURBATIONS

To study whether the VE perturbations are inherently weaker than VP perturbations, we conducted a comprehensive architecture search to compare their best performance. Our results demonstrate that the VE perturbation can be advantageous in certain cases. Further, our endeavor gives rise to new state-of-the-art models on CIFAR-10, and enables the first high-fidelity image samples of resolution 1024×1024 from score-based generative models. Models will be open sourced.

G.1 SETTINGS

We consider two datasets: 32×32 CIFAR-10 (Krizhevsky et al., 2009) and 64×64 CelebA (Liu et al., 2015). All models are trained with 1.3M iterations, and we save one checkpoint per 50k iterations. We compare different configurations based on their average FID scores over all checkpoints after 0.5M iterations. The FIDs are computed on 50k samples with TF-GAN. For sampling, we use the PC sampler discretized at 1000 noise scales. We follow Ho et al. (2020) for optimization, including the learning rate, gradient clipping, and learning rate warm-up schedules. Unless otherwise noted, all models are trained with the original SMLD objective Eq. (1).

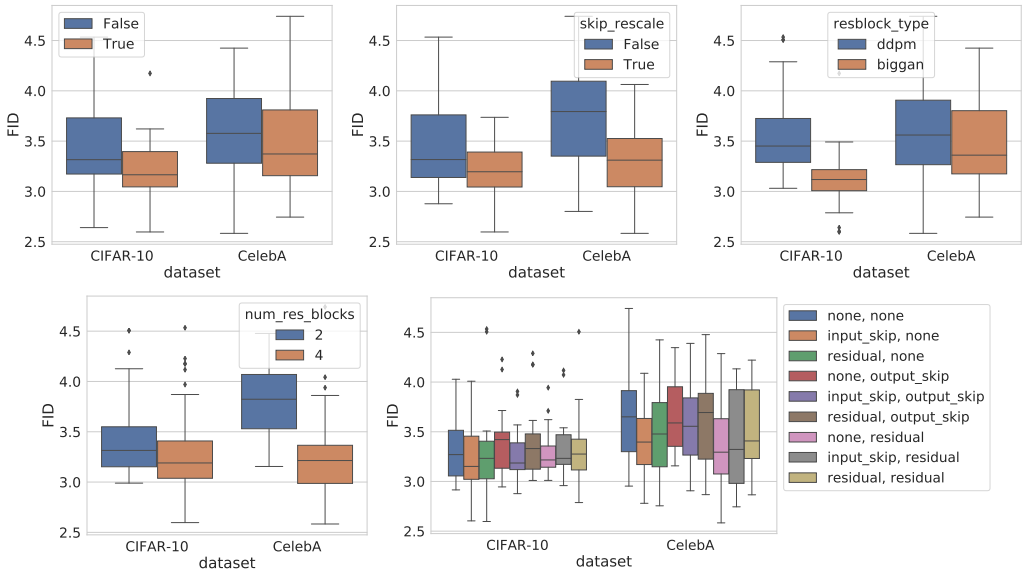

Figure 7: The effects of different architecture components for score-based models trained with VE perturbations.

Architectures Our architecture is mostly based on Ho et al. (2020). We additionally search over the following components to explore the potential of score-based models trained with VE perturbations.

- Upsampling and downsampling images with anti-aliasing based on Finite Impulse Response (FIR) (Zhang, 2019). We follow the same implementation and hyperparameters in StyleGAN-2 (Karras et al., 2020b).

- Rescaling all skip connections by $1/\sqrt{2}$. This has been demonstrated effective in several works, including ProgressiveGAN (Karras et al., 2017), StyleGAN (Karras et al., 2019) and StyleGAN-2 (Karras et al., 2020b).
- Replacing the original residual blocks in DDPM with residual blocks from BigGAN (Brock et al., 2018).
- Increasing the number of residual blocks per resolution from 2 to 4.
- Incorporating progressive growing architectures. We consider two progressive architectures for input: "input_skip" and "residual", and two progressive architectures for output: "output_skip" and "residual". All progressive architectures are implemented with reference to StyleGAN-2.

We also tested equalized learning rates, a trick used in very successful models like Progressive-GAN (Karras et al., 2017) and StyleGAN (Karras et al., 2019). However, we found it harmful at an early stage of our experiments, and therefore decided not to include it in the architecture search.

The exponential moving average (EMA) rate has a significant impact on performance. For models trained with VE perturbations, we notice that 0.999 works better than 0.9999, whereas for models trained with VP perturbations it is the opposite. We therefore use an EMA rate of 0.999 and 0.9999 for VE and VP models respectively.

G.2 Results on CIFAR-10

We provide the architecture search results in Fig. 7. The box plots demonstrate the importance of each component when other components change freely. On both CIFAR-10 and CelebA, the additional components that we searched over can always improve the performance on average. For progressive growing, it is not clear which combination of configurations consistently performs the best, but the results are typically better than when no progressive growing architecture is used. Our best score-based model uses FIR upsampling/downsampling, rescales skip connections, employs BigGAN-type residual blocks, uses 4 residual blocks per resolution instead of 2, and uses "residual" for input and no progressive growing architecture for output. We name this model "NCSN++", following the naming convention of previous SMLD models.

The basic NCSN++ model with 4 residual blocks achieves an FID of 2.45 on CIFAR-10. Here in order to match the convention used in Karras et al. (2017); Song & Ermon (2019); Ho et al. (2020), the FID value reported here is the lowest over the course of training, rather than the average over all checkpoints after 0.5M iterations (as done in our architecture search). We conducted the same architecture search for a score-based model trained with the DDPM objective Eq. (3). The optimal model obtains an FID of 2.88 CIFAR-10, significantly worse than the model trained with VE perturbations. Interestingly, the optimal score-based model for VP perturbations has significantly different configurations than those trained with VE perturbation. Some of the components like rescaling skip connections and progressive growing architectures seem to be harmful for them. This further confirms that the relative advantage of VE/VP SDEs is strongly correlated with model architecture, and should be jointly tuned as part of the design choices.

To further improve the NCSN++ model, we change positional embeddings, the layers in Ho et al. (2020) for conditioning on noise scales, to random Fourier feature embeddings, as advocated in Tancik et al. (2020). We additionally train the model with our continuous objective Eq. (11). These changes improve the FID on CIFAR-10 from 2.45 to 2.38, resulting in a model dubbed NCSN++ cont. (4 blocks/resolution). Finally, we can further improve the FID from 2.38 to 2.2 by doubling the number of residual blocks per resolution, resulting in the model denoted as NCSN++ cont. (8 blocks/resolution). All results are summarized in Table 3.

G.3 High resolution images

Encouraged by the success of NCSN++ on CIFAR-10, we proceed to test it on 1024×1024 CelebA-HQ (Karras et al., 2017), a task that was previously only achievable by GANs and VQ-VAE-2 (Razavi et al., 2019). We used a batch size of 8, increased the EMA rate to 0.9999, and trained the NCSN++ model with the continuous objective (Eq. (11)) for around 700k iterations. We use the PC sampler discretized at 2000 steps. A set of uncurated samples is given in Fig. 8.

Figure 8: Samples on 1024×1024 CelebA-HQ from continuously trained NCSN++.