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. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed 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. In addition, our framework 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 2.20, a competitive likelihood of 3.10 bits/dim, and demonstrate high fidelity generation of 1024×1024 images for the first time from a score-based generative model.

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 scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales 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 computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017), have proven effective at generation of images (Song & Ermon, 2019; 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 these two model classes is often quite different for reasons that are not fully 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 distributions that evolve over time according to a diffusion process. This process progressively diffuses a data point into random noise, and is given by a

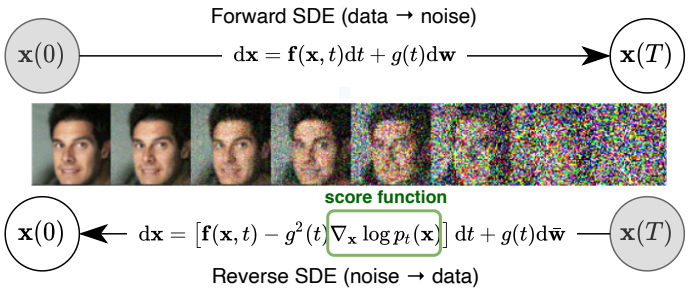


Figure 1: **Solving a reverse-time SDE yields a score-based generative model.** Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log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

prescribed SDE that does not depend on the data and has no trainable parameters. 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, 1982), which can be derived from the forward SDE given the score 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 has several theoretical and practical contributions:

Flexible sampling: We can employ any general-purpose SDE solver to integrate the reverse-time SDE for sampling. In addition, we propose two special methods not viable for 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 the probability flow ordinary differential equation (ODE). 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, *flexible data manipulation* via latent codes, and a *unique encoding*.

Controllable generation: We can modulate the generation process by conditioning on information not available during training, because the conditional reverse-time SDE can be efficiently computed from *unconditional* scores. This enables applications such as class-conditional generation, image inpainting, and colorization using a single unconditional score-based model without re-training.

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 for score-based generative modeling.

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 (Vincent, 2011) 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 $p_{\sigma_{\text{min}}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ under some regularity conditions.

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 Markov chain $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_N\}$ is constructed such that $p(\mathbf{x}_i \mid \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \sqrt{1-\beta_i}\mathbf{x}_{i-1}, \beta_i\mathbf{I})$, and therefore $p_{\alpha_i}(\mathbf{x}_i \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i; \sqrt{\alpha_i}\mathbf{x}_0, (1-\alpha_i)\mathbf{I})$, where $\alpha_i \triangleq \prod_{j=1}^{i}(1-\beta_j)$. Similar to SMLD, we can denote the perturbed data distribution as $p_{\alpha_i}(\tilde{\mathbf{x}}) \triangleq \int p_{\text{data}}(\mathbf{x}) p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$. The noise scales are prescribed such that $\mathbf{x}_N$ is an approximate sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. A variational Markov chain 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 re-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 Markov chain 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 performing 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 weighted sum of denoising score matching objectives, which implies that the optimal model, $\mathbf{s}_{\boldsymbol{\theta}^*}(\tilde{\mathbf{x}}, i)$, matches the score of the perturbed data distribution, $\nabla_{\mathbf{x}} \log p_{\alpha_i}(\mathbf{x})$. Moreover, we observe that the weights of the i-th summand in Eq. (1) and Eq. (3), namely σ_i^2 and $(1-\alpha_i)$, are related to corresponding perturbation kernels in the same functional form: $\sigma_i^2 \propto 1/\mathbb{E}[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$ and $(1-\alpha_i) \propto 1/\mathbb{E}[\|\nabla_{\mathbf{x}} \log p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2]$.

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 SDE as the noise intensifies. An overview of our framework is given in Fig. 2.

3.1 Perturbing data with SDEs

Our goal is to construct a diffusion process $\{\mathbf{x}(t)\}_{t=0}^{T}$ indexed by a continuous time variable $t \in [0, T]$, such that $\mathbf{x}(0) \sim p_0$, for which we have a dataset of i.i.d. samples, and $\mathbf{x}(T) \sim p_T$, for which we have a tractable form to generate samples efficiently. In other words, p_0 is the data distribution and p_T is the prior distribution. This diffusion process can be modeled as the solution to an Itô SDE:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \tag{5}$$

where $\mathbf{w}$ is the standard Wiener process (a.k.a., Brownian motion), $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ 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 *diffusion* coefficient of $\mathbf{x}(t)$. For ease of presentation we assume the diffusion coefficient is a scalar (instead 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). The SDE has a unique strong solution when the coefficients are globally Lipschitz in both state and time (Øksendal, 2003). We hereafter denote by $p_t(\mathbf{x})$ the

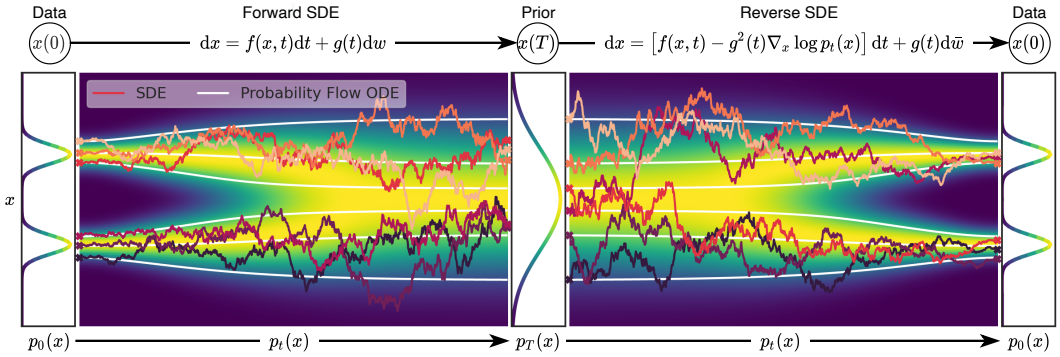


Figure 2: **Overview of score-based generative modeling through SDEs**. We can map data to a noise distribution (the prior) with an SDE (Section 3.1), and reverse this SDE for generative modeling (Section 3.2). We can also reverse the associated probability flow ODE (Section 4.3), which yields a deterministic process that samples from the same distribution as the SDE. Both the reverse-time SDE and probability flow ODE can be obtained by estimating the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ (Section 3.3).

probability density of $\mathbf{x}(t)$, and use $p_{st}(\mathbf{x}(t) \mid \mathbf{x}(s))$ to denote the transition kernel from $\mathbf{x}(s)$ to $\mathbf{x}(t)$, where $0 \leqslant s < t \leqslant T$.

Typically, p_T is an unstructured prior distribution that contains no information of p_0, such as a Gaussian distribution with simple mean and variance. There are various ways of designing the SDE in Eq. (5) such that it diffuses the data distribution into a fixed prior distribution. We provide two examples later in Section 3.4.

3.2 Generating samples by reversing the SDE

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 a diffusion process is also a diffusion process, running backwards in time and given by the reverse-time SDE:

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x},t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{6}$$

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 diffusion process from Eq. (6) and simulate it to sample from p_0.

3.3 Estimating scores for the SDE

The score of a distribution can be estimated by training a score-based model on samples with score matching (Hyvärinen, 2005; Song et al., 2019a). 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 continuous generalization to Eqs. (1) and (3):

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)} \big[\big\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t),t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \big\|_2^2 \big] \Big\}. \tag{7}$$

Here $\lambda : [0,T] \to \mathbb{R}^+$ is a positive weighting function, t is sampled from a uniform distribution over $[0,T]$, $\mathbf{x}(0) \sim p_0(\mathbf{x})$ and $\mathbf{x}(t) \sim p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. With sufficient data and model capacity, score matching ensures that the optimal solution to Eq. (7), 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. As in SMLD and DDPM, we can typically choose $\lambda(t) \propto 1/\mathbb{E}\big[\big\| \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \big\|_2^2 \big]$. Note that Eq. (7) 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.

We need to know the transition kernel $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ to efficiently solve Eq. (7). When $\mathbf{f}(\cdot,t)$ is affine, the transition kernel is always a Gaussian distribution, where the mean and variance are often known in closed-forms and can be obtained with standard techniques (see Section 5.5 in Särkkä & Solin (2019)). For more general SDEs, we may solve Kolmogorov's forward equation (Øksendal,

2003) to obtain $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. Alternatively, we can simulate the SDE to sample from $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ and replace denoising score matching in Eq. (7) with sliced score matching for model training, which does not require computing $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ (see Appendix A).

3.4 Special cases: VE and VP SDEs

The noise perturbations used in SMLD and DDPM can be regarded as discretizations of two different SDEs, which we name as Variance Exploding (VE) and Variance Preserving (VP) SDEs respectively. Below we provide a brief discussion and relegate more details to Appendix B.

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 distribution of $\mathbf{x}_i$ in the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N, \tag{8}$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and we have introduced $\sigma_0 = 0$ to simplify the notation. In the limit of $N \to \infty$, $\{\sigma_i\}_{i=1}^N$ becomes a function $\sigma(t)$, $\mathbf{z}_i$ becomes $\mathbf{z}(t)$, and the Markov chain $\{\mathbf{x}_i\}_{i=1}^N$ becomes a continuous stochastic process $\{\mathbf{x}(t)\}_{t=0}^1$, where we have used a continuous time variable $t \in [0, 1]$ for indexing, rather than an integer i. The process $\{\mathbf{x}(t)\}_{t=0}^1$ is given by the following SDE

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}. \tag{9}$$

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

$$\mathbf{x}_i = \sqrt{1 - \beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N. \tag{10}$$

As $N \to \infty$, Eq. (10) converges to the following SDE,

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}. \tag{11}$$

Therefore, the noise perturbations used in SMLD and DDPM correspond to discretizations of SDEs Eq. (9) and Eq. (11) respectively. Interestingly, the SDE of Eq. (9) always gives a process with exploding variance when $t \to \infty$, whilst the SDE of Eq. (11) yields a process with bounded variance (proof in Appendix B). Due to this difference, we hereafter refer to Eq. (9) as the Variance Exploding (VE) SDE, and Eq. (11) the Variance Preserving (VP) SDE. Since both VE and VP SDEs have affine drift coefficients, their perturbation kernels $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ are both Gaussian and can be computed in closed-form, as discussed in Section 3.3. This makes training with Eq. (7) particularly efficient.

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.

4.1 General-purpose numerical SDE solvers

Numerical solvers provide approximate trajectories from SDEs. Many general-purpose numerical methods exist for solving SDEs, 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. (11)) (see Appendix E). 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 (details in Appendix E). The resulting samplers for SMLD and DDPM are given as the "predictor" part in Algs. 1 and 2, and we call this family of methods the *reverse diffusion sampler*. As shown in Tab. 1, these sampling methods perform slightly better than ancestral sampling for both SMLD and DDPM models on CIFAR-10 (DDPM-type ancestral sampling can be applied to SMLD models, see Appendix F).

Algorithm 1 PC sampling (VE SDE)

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

Algorithm 2 PC sampling (VP SDE)

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

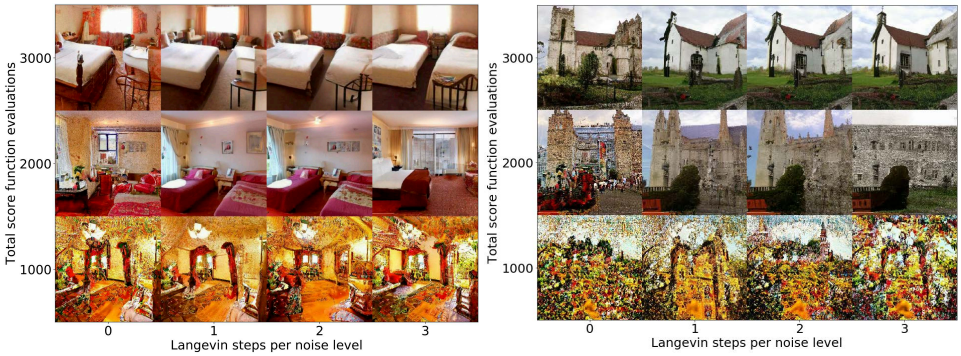


Figure 3: PC sampling for LSUN bedroom and church. The vertical axis corresponds to the total computation, and the horizontal axis represents the amount of computation allocated to the corrector. Samples are the best when computation is split between the predictor and corrector.

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". The idea is analogous to Predictor-Corrector methods for solving systems of equations (Allgower & Georg, 2012), and we similarly name our hybrid sampling algorithms *Predictor-Corrector* (PC) samplers.

When using the reverse diffusion SDE solver (Appendix E) as the predictor, and annealed Langevin dynamics (Song & Ermon, 2019) as the corrector, we have Algs. 1 and 2 for VE and VP SDEs respectively, where $\{\epsilon_i\}_{i=0}^{N-1}$ are step sizes for Langevin dynamics (specified in Appendix G). PC samplers 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 ancestral sampling method for DDPM when using it as the predictor and removing the corrector part.

We test PC samplers on SMLD/DDPM models trained with Eqs. (1) and (3), instead of our new continuous objective Eq. (7). This exhibits the compatibility of PC samplers to score-based models trained with a fixed number of noise scales. We compare different samplers in Tab. 1, where probability flow is a predictor to be discussed in Section 4.3. Experimental details are given in Appendix G. Due to additional corrector steps, PC samplers require twice as much computation as predictor-only samplers. To match the computation, one has to use more noise scales for predictor-only samplers, which is difficult for models trained with a fixed number of noise scales as in SMLD/DDPM. As a workaround, we use an ad-hoc method to interpolate between noise scales (Appendix G). Observations from Tab. 1 include: (i) our reverse diffusion method uniformly outperforms ancestral sampling; (ii) PC samplers uniformly outperform predictor-only methods with double computation; (iii) PC samplers can outperform predictor-only and corrector-only methods in most cases using

Table 1: Comparing different reverse-time SDE solvers on CIFAR-10. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs. "P1000" or "P2000": predictor-only samplers using 1000 or 2000 steps. "C2000": corrector-only samplers using 2000 steps. "PC1000": Predictor-Corrector (PC) samplers using 1000 predictor and 1000 corrector steps.

	Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
FID↓ Sampler / Predictor	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	4.98 ± .06	4.92 ± .02		**3.62 ± .03**	3.24 ± .02	**3.11 ± .03**		3.21 ± .02
reverse diffusion	4.79 ± .07	4.72 ± .07	20.43 ± .07	**3.60 ± .02**	3.21 ± .02	**3.10 ± .03**	19.06 ± .06	3.18 ± .01
probability flow	15.41 ± .15	12.87 ± .09		**3.51 ± .04**	3.59 ± .04	3.25 ± .04		**3.06 ± .03**

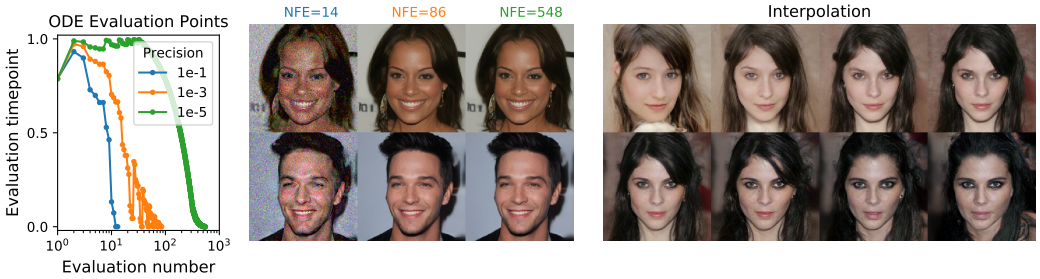


Figure 4: **Probability flow ODE enables efficient sampling** with adaptive step-sizes as the numerical precision is varied (*left*), and reduces the number of function evaluations (NFE) without harming quality (*middle*). The determinstic mapping from latents to images allows for interpolations (*right*).

matched computation. The performance gap may be even larger with different noise interpolation strategies (Tab. 4 in Appendix G). In Fig. 3, we additionally provide qualitative comparison for models trained with the continuous objective Eq. (7), where PC samplers clearly surpass predictor-only samplers under comparable computation, when using a proper number of corrector steps.

4.3 Probability flow and equivalence to neural ODEs

Score-based models enable another numerical method for solving the reverse-time SDE. For all diffusion processes, there exists a corresponding *deterministic process* whose trajectories share the same marginal probability densities $\{p_t(\mathbf{x})\}_{t=0}^{T}$. This deterministic process satisfies an ODE (more details in Appendix D.1):

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

which can be determined from the SDE once scores are known. We name the ODE in Eq. (12) the *probability flow ODE*. As the score functions are typically parameterized by a neural network, this is an example of a neural ODE (Chen et al., 2018).

Efficient sampling As with neural ODEs, we can sample $\mathbf{x}(0) \sim p_0$ by solving Eq. (12) from different final conditions $\mathbf{x}(T) \sim p_T$. Using a fixed discretization strategy we can generate competitive samples, especially when used in conjuction with correctors (Tab. 1, "probability flow sampler", details in Appendix D.3). Using a black-box ODE solver (Dormand & Prince, 1980) 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 affecting the visual quality of samples (Fig. 4).

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 (Chen et al., 2018). This allows us to compute the *exact likelihood on any input data* (details in Appendix D.2). As an example, we report negative log-likelihoods (NLLs) measured in bits/dim on the CIFAR-10 dataset in Tab. 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

Table 2: NLLs on CIFAR-10.

Model	NLL Test ↓	FID ↓
RealNVP (Dinh et al., 2016)	3.49	-
iResNet (Behrmann et al., 2019)	3.45	-
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., 2018)	3.40	-
Flow++ (Ho et al., 2019)	3.29	-
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)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	8.87 $\pm$ .12
NCSNv2 (Song & Ermon, 2020)	10.87	8.40 $\pm$ .07
DDPM (Ho et al., 2020)	3.17	9.46 $\pm$ .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**

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 denoising score matching objective in Eq. (7) on uniformly dequantized data. This continuously-trained model (DDPM cont. in Tab. 2) has much lower bits/dim compared to the original one, setting a new record bits/dim on uniformly dequantized CIFAR-10 even *without maximum likelihood training*.

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. 4 and Appendix D.4).

Uniquely identifiable encoding Unlike all current invertible models, our encoding 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. (5), has no trainable parameters, and its associated probability flow ODE, Eq. (12), provides the same trajectories given perfectly estimated scores. We provide additional empirical verification on this property in Appendix D.5.

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 10.23 (Song & Ermon, 2020), 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 Tab. 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 H). Our optimal architecture for the VE SDE, dubbed NCSN++, achieves an FID of 2.45 on CIFAR-10 with PC samplers, 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 our continuous training objective in Eq. (7) (FID 2.38), and increasing the number of residual blocks (FID 2.2, see Tab. 3 and Appendix H.2). This model sets new records for both inception score and FID on unconditional generation for CIFAR-10. Surprisingly, we can achieve better FID than the previous best conditional generative model without requiring 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 H.3).

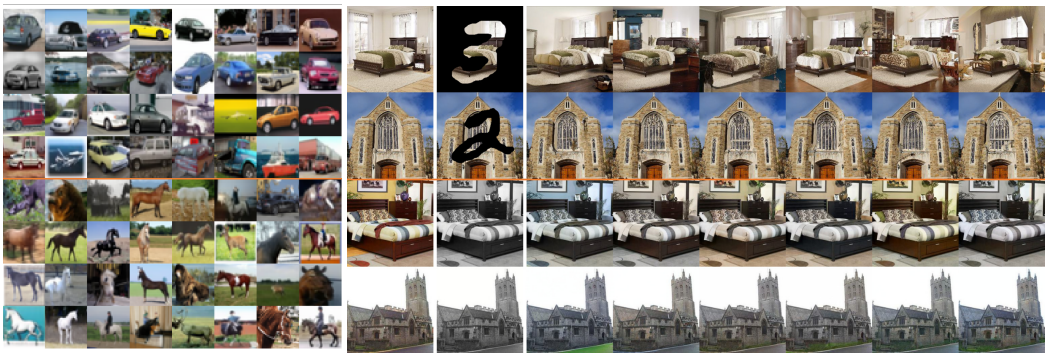


Figure 5: *Left*: Class-conditional samples on 32×32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right:* Inpainting (top two rows) and colorization (bottom two rows) results on 256×256 LSUN. First column is the original image, second column is the masked/gray-scale image, remaining columns are sampled image completions or colorizations.

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. (5), 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. (7), to train the time-dependent classifier $p_t(\mathbf{y} \mid \mathbf{x}(t))$. Class-conditional CIFAR-10 samples can be found in Fig. 5 (left). See more details and results are in Appendix I.

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 I.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 I.3). Fig. 5 (right) shows results for inpainting and colorization achieved with unconditional time-dependent score-based models.

6 Discussion

We presented a framework for score-based generative modeling based on SDEs. Our work enables a better understanding of existing approaches, new sampling algorithms, exact likelihood computation, uniquely identifiable encoding, latent code manipulation, 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 (Goodfellow et al., 2014) 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. Additionally, the breadth of samplers one can use when given access to score functions introduces a number of hyper-parameters. Future work would benefit from improved methods to automatically select and tune these hyper-parameters, as well as more extensive investigation on the merits and limitations of various samplers.

References

Eugene L Allgower and Kurt Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 2012.

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

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582, 2019.

Florian Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. *arXiv preprint arXiv:1703.06975*, 2017.

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.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Anirudh Goyal Alias Parth Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. Variational walkback: Learning a transition operator as a stochastic recurrent net. In *Advances in Neural Information Processing Systems*, pp. 4392–4402, 2017.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

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. *Advances in Neural Information Processing Systems*, 33, 2020.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

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. In *International Conference on Learning Representations*, 2018.

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. *Advances in Neural Information Processing Systems*, 33, 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.

John Skilling. The eigenvalues of mega-dimensional matrices. In *Maximum Entropy and Bayesian Methods*, pp. 455–466. Springer, 1989.

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. *Advances in Neural Information Processing Systems*, 33, 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.

APPENDIX

We include several appendices with additional details, derivations, and results. Our framework allows general SDEs with matrix-valued diffusion coefficients that depend on the state, for which we provide a detailed discussion in Appendix A. We give a full derivation of VE and VP SDEs in Appendix B, and discuss how to use them from a practitioner's perspective in Appendix C. We elaborate on the probability flow formulation of our framework in Appendix D, including a derivation of the probability flow ODE (Appendix D.1), exact likelihood computation (Appendix D.2), probability flow sampling (Appendices D.3 and D.4) and experimental verification on uniquely identifiable encoding (Appendix D.5). We give a full description of the reverse diffusion sampler in Appendix E, and the DDPM-type ancestral sampler for SMLD models in Appendix F. We provide full experimental details on Predictor-Corrector samplers in Appendix G. We explain our model architectures and detailed experimental settings in Appendix H, with 1024×1024 CelebA-HQ samples therein. Finally, we detail on the algorithms for controllable generation in Appendix I, and include extended results for class-conditional generation (Appendix I.1), image inpainting (Appendix I.2) and colorization (Appendix I.3).

A THE FRAMEWORK FOR MORE GENERAL SDES

In the main text, we introduced our framework based on a simplified SDE Eq. (5) where the diffusion coefficient is independent of $\mathbf{x}(t)$. It turns out that our framework can be extended to hold for more general diffusion coefficients. We can consider SDEs in the following 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 $\mathbf{G}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$. We follow the Itô interpretation of SDEs throughout this paper.

According to (Anderson, 1982), the reverse-time SDE is given by (*cf*., Eq. (6))

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

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 corresponding to Eq. (14) has the following form (*cf*., Eq. (12), see a detailed derivation in Appendix D.1):

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

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 I):

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

When the drift and diffusion coefficient of an SDE are not affine, it can be difficult to compute the transition kernel $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ in closed form. This hinders the training of score-based models, because Eq. (7) requires knowing $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. To overcome this difficulty, we can replace denoising score matching in Eq. (7) with other efficient variants of score matching that do not require computing $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$. For example, when using sliced score matching (Song et al., 2019a), our training objective Eq. (7) becomes

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t\left\{\lambda(t)\mathbb{E}_{\mathbf{x}(0)}\mathbb{E}_{\mathbf{x}(t)}\mathbb{E}_{\mathbf{v}\sim p_{\mathbf{v}}}\left[\frac{1}{2}\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)\right\|_2^2 + \mathbf{v}^\mathsf{T}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)\mathbf{v}\right]\right\}, \tag{18}$$

where $\lambda : [0, T] \to \mathbb{R}^+$ is a positive weighting function, $t \sim \mathcal{U}(0, T)$, $\mathbb{E}[\mathbf{v}] = \mathbf{0}$, and $\mathrm{Cov}[\mathbf{v}] = \mathbf{I}$. We can always simulate the SDE to sample from $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$, and solve Eq. (18) to train the time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$.

B DERIVATIONS OF VE AND VP SDES

Below we provide detailed derivations to show that the noise perturbations of SMLD and DDPM are discretizations of the Variance Exploding (VE) and Variance Preserving (VP) SDEs respectively.

First, when using a total of N noise scales, each perturbation kernel $p_{\sigma_i}(\mathbf{x} \mid \mathbf{x}_0)$ of SMLD can be derived from the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N, \tag{19}$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_0 \sim p_{\text{data}}$, and we have introduced $\sigma_0 = 0$ to simplify the notation. In the limit of $N \to \infty$, the Markov chain $\{\mathbf{x}_i\}_{i=1}^N$ becomes a continuous stochastic process $\{\mathbf{x}(t)\}_{t=0}^1$, $\{\sigma_i\}_{i=1}^N$ becomes a function $\sigma(t)$, and $\mathbf{z}_i$ becomes $\mathbf{z}(t)$, where we have used a continuous time variable $t \in [0, 1]$ for indexing, rather than an integer $i \in \{1, 2, \cdots, N\}$. 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. (19) 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 the approximate equality holds when $\Delta t \ll 1$. In the limit of $\Delta t \to 0$, this converges to

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}, \tag{20}$$

which is the VE SDE. Note that $\mathbf{x}(0) = \mathbf{x}_1 = \mathbf{x}_0 + \sigma_1\mathbf{z}_0 = \mathbf{x}_0 + \sigma^2(0)\mathbf{z}_0$ and $\mathbf{x}_0 \sim p_{\text{data}}$, so we have $p_0(\mathbf{x}(0)) = \int p_{\text{data}}(\mathbf{x})\mathcal{N}(\mathbf{x}(0); \mathbf{x}, \sigma_1^2\mathbf{I})\mathrm{d}\mathbf{x}$. In other words, $p_0(\mathbf{x}(0))$ for the VE SDE is a perturbed data distribution with a small Gaussian noise.

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

$$\mathbf{x}_i = \sqrt{1 - \beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N, \tag{21}$$

where $\mathbf{z}_{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To obtain the limit of this Markov chain when $N \to \infty$, we define an auxiliary set of noise scales $\{\bar{\beta}_i = (N-1)\beta_i\}_{i=1}^N$, and re-write Eq. (21) as below

$$\mathbf{x}_i = \sqrt{1 - \frac{\bar{\beta}_i}{N-1}}\mathbf{x}_{i-1} + \sqrt{\frac{\bar{\beta}_i}{N-1}}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N. \tag{22}$$

In the limit of $N \to \infty$, $\{\bar{\beta}_i\}_{i=1}^N$ becomes a function $\beta(t)$ indexed by $t \in [0, 1]$. Let $\beta\left(\frac{i-1}{N-1}\right) = \bar{\beta}_i$, $\mathbf{x}(\frac{i-1}{N-1}) = \mathbf{x}_i$, $\mathbf{z}(\frac{i-1}{N-1}) = \mathbf{z}_i$ and $\Delta t = \frac{1}{N-1}$. The Markov chain Eq. (22) becomes the following at $t \in \{0, 1, \cdots, \frac{N-2}{N-1}\}$

$$\mathbf{x}(t + \Delta t) = \sqrt{1 - \beta(t)\Delta t}\, \mathbf{x}(t) + \sqrt{\beta(t)\Delta t}\, \mathbf{z}(t) \approx -\frac{1}{2}\beta(t)\Delta t\, \mathbf{x}(t) + \sqrt{\beta(t)\Delta t}\, \mathbf{z}(t), \tag{23}$$

where the approximate equality holds when $\Delta t \ll 1$. Therefore, in the limit of $\Delta t \to 0$, Eq. (23) converges to the following VP SDE:

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\, \mathrm{d}t + \sqrt{\beta(t)}\, \mathrm{d}\mathbf{w}. \tag{24}$$

Because $\mathbf{x}(0) = \lim_{N\to\infty} \sqrt{1 - \beta_1}\mathbf{x}_0 + \sqrt{\beta_1}\mathbf{z}_0 = \lim_{N\to\infty} \sqrt{1 - \frac{\beta(0)}{N-1}}\mathbf{x}_0 + \sqrt{\frac{\beta(0)}{N-1}}\mathbf{z}_0 = \mathbf{x}_0$ and $\mathbf{x}_0 \sim p_{\text{data}}$, we have $p_0(\mathbf{x}(0)) = p_{\text{data}}(\mathbf{x})$.

So far, we have demonstrated that the noise perturbations used in SMLD and DDPM correspond to discretizations of VE and VP SDEs respectively. The VE SDE always yields a process with exploding variance when $t \to \infty$. In contrast, the VP SDE 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. Since the VP SDE has affine drift and diffusion coefficients, we can use Eq. (5.51) 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 $\mathbf{\Sigma}(0) = \mathbf{I}$. Due to this difference, we name Eq. (9) as the *Variance Exploding (VE) SDE*, and Eq. (11) the *Variance Preserving (VP) SDE*.

Since both VE and VP SDEs have affine drift coefficients, their perturbation kernels $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ are both Gaussian and can be computed with Eqs. (5.50) and (5.51) in Särkkä & Solin (2019):

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \begin{cases} \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}\big), & \text{(VE SDE)} \\ \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2}\int_0^t \beta(s)\mathrm{d}s}, \mathbf{I} - \mathbf{I}e^{-\int_0^t \beta(s)\mathrm{d}s}\big) & \text{(VP SDE)} \end{cases}. \tag{25}$$

Therefore, both SDEs allow efficient training with the objective in Eq. (7).

C SDEs in the wild

Below we discuss SDEs that directly correspond to implementations for practical SMLD and DDPM models. In SMLD, the noise scales $\{\sigma_i\}_{i=1}^N$ is typically a geometric sequence where $\sigma_{\min}$ is fixed to 0.01 and $\sigma_{\max}$ is chosen according to Technique 1 in Song & Ermon (2020). Usually, SMLD models normalize image inputs to the range $[0, 1]$. Since $\{\sigma_i\}_{i=1}^N$ is a geometric sequence, we have $\sigma_i = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{\frac{i-1}{N-1}}$ and therefore $\sigma(t) = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^t$. The VE SDE in this case is

$$\mathrm{d}\mathbf{x} = \sigma_{\min}\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^t \sqrt{2\log\frac{\sigma_{\max}}{\sigma_{\min}}}\mathrm{d}\mathbf{w}, \tag{26}$$

where $\mathbf{x}(0) \sim \int p_{\text{data}}(\mathbf{x})\mathcal{N}(\mathbf{x}(0); \mathbf{x}, \sigma_{\min}^2\mathbf{I})\mathrm{d}\mathbf{x}$. The perturbation kernel can be computed via Eq. (25):

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \mathcal{N}\left(\mathbf{x}(t); \mathbf{x}(0), \left[\sigma_{\min}^2\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{2t} - \sigma_{\min}^2\right]\mathbf{I}\right). \tag{27}$$

For DDPM models, $\{\beta_i\}_{i=1}^N$ is typically an arithmetic sequence where $\beta_i = \frac{\bar{\beta}_{\min}}{N-1} + \frac{i-1}{(N-1)^2}(\bar{\beta}_{\max} - \bar{\beta}_{\min})$. Therefore, $\beta(t) = \bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min})$. This corresponds to the following instantiation of the VP SDE:

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}(\bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min}))\mathbf{x}\mathrm{d}t + \sqrt{\bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min})}\mathrm{d}\mathbf{w}, \tag{28}$$

where $\mathbf{x}(0) \sim p_{\text{data}}(\mathbf{x})$. In our experiments, we let $\bar{\beta}_{\min} = 0.1$ and $\bar{\beta}_{\max} = 20$, which correspond to the settings in Ho et al. (2020). The perturbation kernel is given by

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \mathcal{N}\left(\mathbf{x}(t); e^{-\frac{1}{4}t^2(\bar{\beta}_{\max} - \bar{\beta}_{\min}) - \frac{1}{2}t\bar{\beta}_{\min}}\mathbf{x}(0), \mathbf{I} - \mathbf{I}e^{-\frac{1}{2}t^2(\bar{\beta}_{\max} - \bar{\beta}_{\min}) - t\bar{\beta}_{\min}}\right). \tag{29}$$

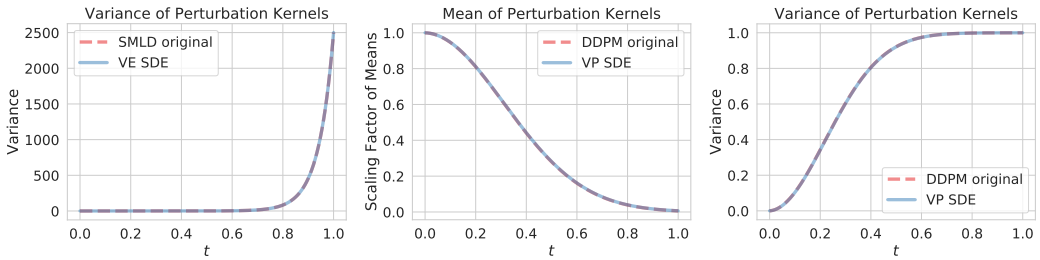
As a sanity check for our SDE generalizations to SMLD and DDPM, we compare the perturbation kernels of SDEs and original discrete Markov chains in Fig. 6. The SMLD and DDPM models both use $N = 1000$ noise scales. For SMLD, we only need to compare the variances of perturbation kernels. For DDPM, we compare the scaling factors of means and the variances. As demonstrated in Fig. 6, the discrete perturbation kernels of original SMLD and DDPM models align well with perturbation kernels derived from VE and VP SDEs.

D Probability flow ODE

D.1 Derivation

The idea of probability flow ODE is inspired by Maoutsa et al. (2020), and one can find the derivation of a simplified case therein. Below we provide a derivation for the fully general ODE in Eq. (16). We consider the SDE in Eq. (14), which possesses the following form:

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



(a) SMLD (b) DDPM (mean) (c) DDPM (variance)

Figure 6: Discrete-time perturbation kernels and our continuous generalizations match each other almost exactly. (a) compares the variance of perturbation kernels for SMLD and VE SDE; (b) compares the scaling factors of means of perturbation kernels for DDPM and VP SDE; and (c) compares the variance of perturbation kernels for DDPM and VP SDE.

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 marginal probability density $p_t(\mathbf{x}(t))$ evolves according to the forward Kolmogorov equation (Fokker-Planck equation) (Øksendal, 2003)

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\sum_{i=1}^d \frac{\partial}{\partial x_i}[f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}\sum_{i=1}^d\sum_{j=1}^d \frac{\partial^2}{\partial x_i \partial x_j}\Big[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x})\Big]. \tag{30}$$

We can easily rewrite Eq. (30) to obtain

$$\begin{aligned}\frac{\partial p_t(\mathbf{x})}{\partial t} &= -\sum_{i=1}^d \frac{\partial}{\partial x_i}[f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}\sum_{i=1}^d\sum_{j=1}^d \frac{\partial^2}{\partial x_i \partial x_j}\Big[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x})\Big] \\ &= -\sum_{i=1}^d \frac{\partial}{\partial x_i}[f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}\sum_{i=1}^d \frac{\partial}{\partial x_i}\Big[\sum_{j=1}^d \frac{\partial}{\partial x_j}\Big[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x})\Big]\Big]. \end{aligned} \tag{31}$$

Note that

$$\begin{aligned}&\sum_{j=1}^d \frac{\partial}{\partial x_j}\Big[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x})\Big] \\ =&\sum_{j=1}^d \frac{\partial}{\partial x_j}\Big[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)\Big]p_t(\mathbf{x}) + \sum_{j=1}^d\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x})\frac{\partial}{\partial x_j}\log p_t(\mathbf{x}) \\ =&p_t(\mathbf{x})\nabla\cdot[\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] + p_t(\mathbf{x})\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}}\log p_t(\mathbf{x}),\end{aligned}$$

based on which we can continue the rewriting of Eq. (31) to obtain

$$\begin{aligned}\frac{\partial p_t(\mathbf{x})}{\partial t} &= -\sum_{i=1}^d \frac{\partial}{\partial x_i}[f_i(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}\sum_{i=1}^d \frac{\partial}{\partial x_i}\Big[\sum_{j=1}^d \frac{\partial}{\partial x_j}\Big[\sum_{k=1}^d G_{ik}(\mathbf{x}, t)G_{jk}(\mathbf{x}, t)p_t(\mathbf{x})\Big]\Big] \\ &= -\sum_{i=1}^d \frac{\partial}{\partial x_i}[f_i(\mathbf{x}, t)p_t(\mathbf{x})] \\ &\quad + \frac{1}{2}\sum_{i=1}^d \frac{\partial}{\partial x_i}\Big[p_t(\mathbf{x})\nabla\cdot[\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] + p_t(\mathbf{x})\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\Big] \\ &= -\sum_{i=1}^d \frac{\partial}{\partial x_i}\Big\{f_i(\mathbf{x}, t)p_t(\mathbf{x}) \\ &\quad - \frac{1}{2}\Big[\nabla\cdot[\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] + \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\Big]p_t(\mathbf{x})\Big\} \\ &= -\sum_{i=1}^d \frac{\partial}{\partial x_i}[\tilde{f}_i(\mathbf{x}, t)p_t(\mathbf{x})],\end{aligned} \tag{32}$$

where we define

$$\tilde{\mathbf{f}}(\mathbf{x}, t) \triangleq \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}\nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^{\mathsf{T}}] - \frac{1}{2}\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^{\mathsf{T}}\nabla_{\mathbf{x}} \log p_t(\mathbf{x}).$$

Inspecting Eq. (32), we observe that it equates the forward Kolmogorov equation of the following SDE with $\tilde{\mathbf{G}}(\mathbf{x}, t) \triangleq \mathbf{0}$ (the forward Kolmogorov equation holds in this case, as can be justified by the instantaneous change of variables formula in Chen et al. (2018))

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

which is essentially an ODE:

$$\mathrm{d}\mathbf{x} = \tilde{\mathbf{f}}(\mathbf{x}, t)\mathrm{d}t,$$

same as the probability flow ODE given by Eq. (16). Therefore, we have shown that the probability flow ODE Eq. (16) induces the same marginal probability density $p_t(\mathbf{x})$ as the SDE in Eq. (14).

D.2 Likelihood computation

The probability flow ODE in Eq. (16) has the following form when we replace the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ with the time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$:

$$\mathrm{d}\mathbf{x} = \underbrace{\left\{\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}\nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^{\mathsf{T}}] - \frac{1}{2}\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^{\mathsf{T}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)\right\}}_{\triangleq \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)} \mathrm{d}t. \tag{33}$$

With the instantaneous change of variables formula (Chen et al., 2018), we can compute the log-likelihood of $p_0(\mathbf{x})$ using

$$\log p_0(\mathbf{x}(0)) = \log p_T(\mathbf{x}(T)) - \int_0^T \nabla \cdot \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}(t), t)\mathrm{d}t, \tag{34}$$

where the random variable $\mathbf{x}(t)$ as a function of t can be obtained by solving the probability flow ODE in Eq. (33). In many cases computing $\nabla \cdot \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ is expensive, so we follow Grathwohl et al. (2018) to estimate it with the Skilling-Hutchinson trace estimator (Skilling, 1989; Hutchinson, 1990). In particular, we have

$$\nabla \cdot \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t) = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}^{\mathsf{T}}\nabla\tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)\boldsymbol{\epsilon}], \tag{35}$$

where $\nabla\tilde{\mathbf{f}}_{\boldsymbol{\theta}}$ denotes the Jacobian of $\tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\cdot, t)$, and the random variable $\boldsymbol{\epsilon}$ satisfies $\mathbb{E}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}] = \mathbf{0}$ and $\mathrm{Cov}_{p(\boldsymbol{\epsilon})}[\boldsymbol{\epsilon}] = \mathbf{I}$. The vector-Jacobian product $\boldsymbol{\epsilon}^{\mathsf{T}}\nabla\tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ can be efficiently computed using reverse-mode automatic differentiation, at approximately the same cost as evaluating $\tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$. As a result, we can sample $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ and then compute an efficient unbiased estimate to $\nabla \cdot \tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ using $\boldsymbol{\epsilon}^{\mathsf{T}}\nabla\tilde{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, t)\boldsymbol{\epsilon}$. Since this estimator is unbiased, we can attain an arbitrarily small error by averaging over a sufficient number of runs. Therefore, by applying the Skilling-Hutchinson estimator Eq. (35) to Eq. (34), we can compute the log-likelihood to any accuracy.

D.3 Probability flow sampling

Suppose we have a forward SDE

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

and one of its discretization

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i\mathbf{z}_i, \quad i = 0, 1, \cdots, N-1, \tag{36}$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We assume the discretization schedule of time is fixed beforehand, and thus we absorb the dependency on Δt into the notations of $\mathbf{f}_i$ and $\mathbf{G}_i$. Using Eq. (16), we can obtain the following probability flow ODE:

$$\mathrm{d}\mathbf{x} = \left\{\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}\mathbf{G}(t)\mathbf{G}(t)^{\mathsf{T}}\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right\} \mathrm{d}t. \tag{37}$$

We may employ any numerical method to integrate the probability flow ODE backwards in time for sample generation. In particular, we propose a discretization in a similar functional form to Eq. (36):

$$\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{f}_{i+1}(\mathbf{x}_{i+1}) + \frac{1}{2}\mathbf{G}_{i+1}\mathbf{G}_{i+1}^{\mathsf{T}}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1), \quad i = 0, 1, \cdots, N-1,$$

where the score-based model $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i)$ is conditioned on the iteration number i. This is a deterministic iteration rule. Unlike reverse diffusion samplers or ancestral sampling, there is no additional randomness once the initial sample $\mathbf{x}_N$ is obtained from the prior distribution. When applied to SMLD models, we can get the following iteration rule for probability flow sampling:

$$\mathbf{x}_i = \mathbf{x}_{i+1} + \frac{1}{2}(\sigma_{i+1}^2 - \sigma_i^2)\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, \sigma_{i+1}), \quad i = 0, 1, \cdots, N-1. \tag{38}$$

Similarly, for DDPM models, we have

$$\mathbf{x}_i = (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \frac{1}{2}\beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1), \quad i = 0, 1, \cdots, N-1. \tag{39}$$

D.4 Probability flow sampling with black-box ODE solvers

For producing figures in Fig. 4, 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 (Dormand & Prince, 1980) provided by `scipy.integrate.solve_ivp` in all cases. The bits/dim values in Tab. 2 are computed with `atol=1e-5` and `rtol=1e-5`, same as Grathwohl et al. (2018). To give the results for DDPM (probability flow) and DDPM cont. (probability flow) in Tab. 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. 4, we demonstrate more examples of latent space manipulation in Fig. 7, including interpolation and temperature scaling. The model tested here is a DDPM model trained with the same settings in Ho et al. (2020).

D.5 Uniquely identifiable encoding

As a sanity check, we train two models (denoted as "Model A" and "Model B") with different architectures using the VE SDE on CIFAR-10. Here Model A is an NCSN++ model with 4 layers per resolution trained using the continuous objective in Eq. (7), and Model B is all the same except that it uses 8 layers per resolution. Model definitions are in Appendix H.

We report the latent codes obtained by Model A and Model B for a random CIFAR-10 image in Fig. 8. In Fig. 9, we show the dimension-wise differences and correlation coefficients between latent encodings on a total of 16 CIFAR-10 images. Our results demonstrate that for the same inputs, Model A and Model B provide encodings that are close in every dimension, despite having different model architectures and training runs.

E 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 discretization of it:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i\mathbf{z}_i, \quad i = 0, 1, \cdots, N-1 \tag{40}$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here we assume the discretization schedule of time is fixed beforehand, and thus we can absorb it into the notations of $\mathbf{f}_i$ and $\mathbf{G}_i$.

Based on Eq. (40), we propose to discretize the reverse-time SDE

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

with a similar functional form, which gives the following iteration rule for $i \in \{0, 1, \cdots, N-1\}$:

$$\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{f}_{i+1}(\mathbf{x}_{i+1}) + \mathbf{G}_{i+1}\mathbf{G}_{i+1}^{\mathsf{T}}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \mathbf{G}_{i+1}\mathbf{z}_{i+1}, \tag{41}$$

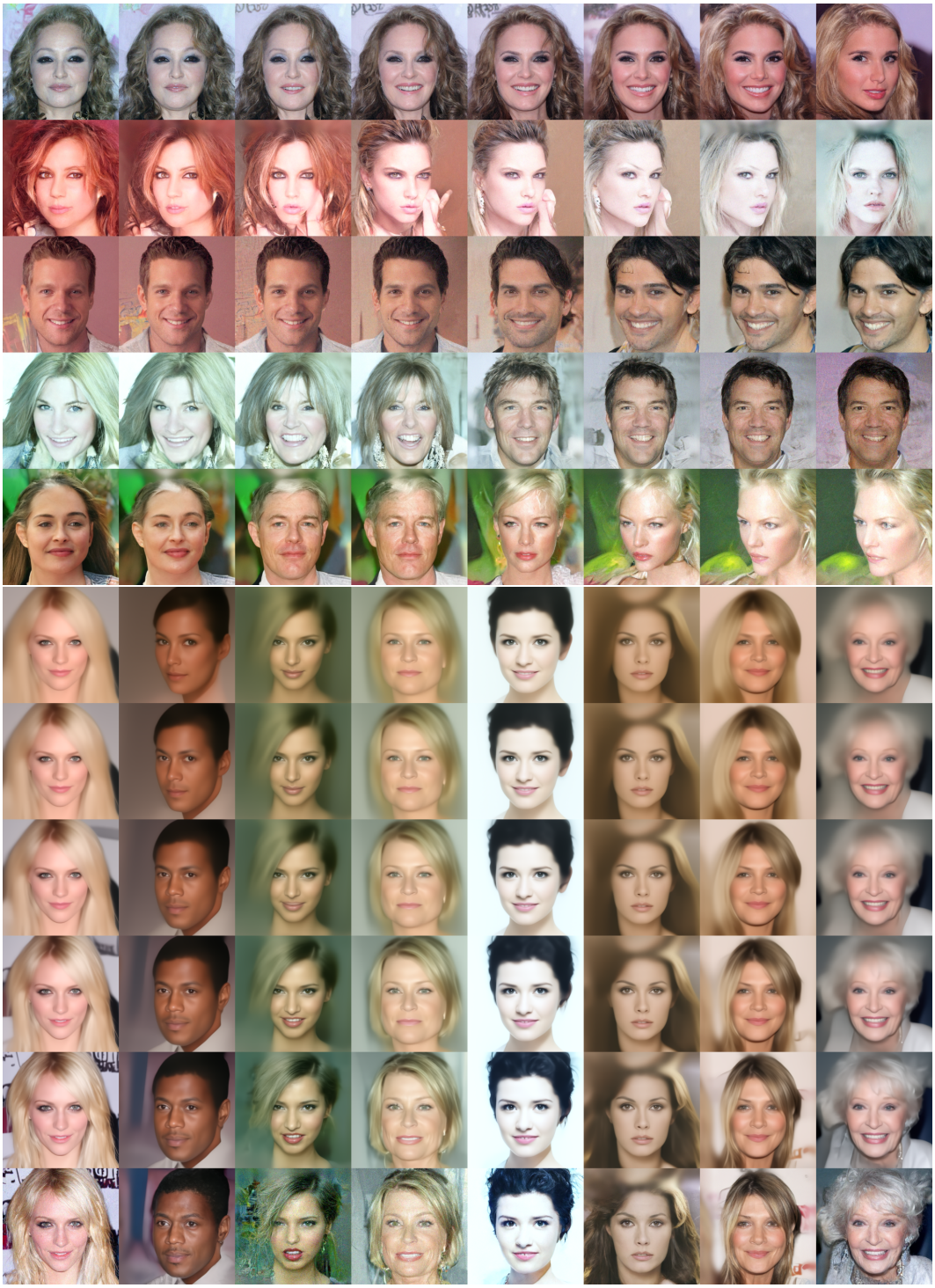


Figure 7: Samples from the probability flow ODE on 256×256 CelebA-HQ with VP perturbations. Top: spherical interpolations between random samples. Bottom: temperature rescaling (reducing norm of embedding).

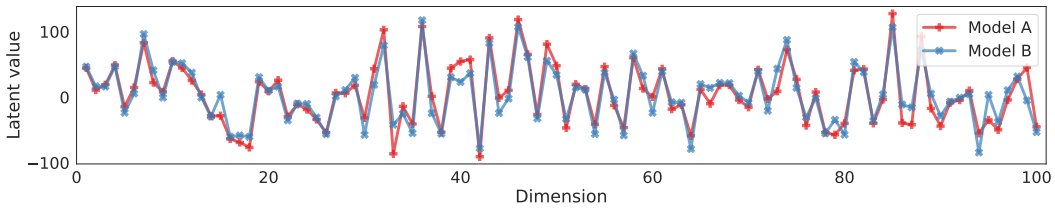


Figure 8: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. "Model A" and "Model B" are separately trained with different architectures.

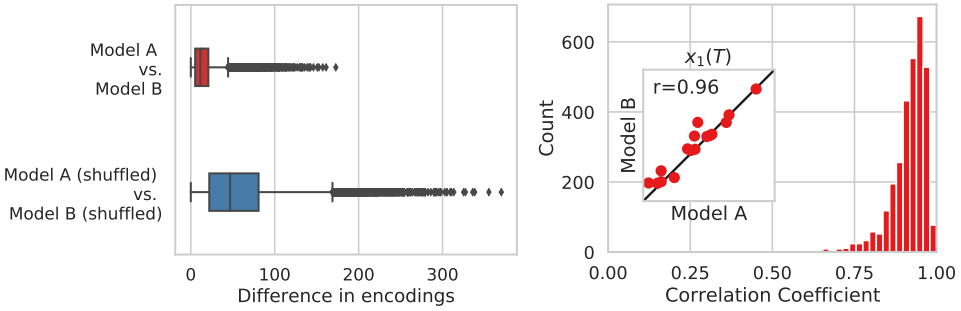


Figure 9: *Left*: The dimension-wise difference between encodings obtained by Model A and B. As a baseline, we also report the difference between shuffled representations of these two models. *Right*: The dimension-wise correlation coefficients of encodings obtained by Model A and Model B.

where our trained score-based model $\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i)$ is conditioned on iteration number i.

When applying Eq. (41) to Eqs. (10) and (19), 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 Algs. 1 and 2. We name these sampling methods (that are based on the discretization strategy in Eq. (41)) *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 happens when $\Delta t \to 0$ since $\beta_i = \bar{\beta}_i \Delta t$), because

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

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.

F ANCESTRAL SAMPLING FOR SMLD MODELS

The ancestral sampling method for DDPM models can also be adapted to SMLD models. Consider a sequence of noise scales $\sigma_1 < \sigma_2 < \cdots < \sigma_N$ as in SMLD. By perturbing a data point $\mathbf{x}_0$ with these noise scales sequentially, we obtain a Markov chain $\mathbf{x}_0 \to \mathbf{x}_1 \to \cdots \to \mathbf{x}_N$, where

$$p(\mathbf{x}_i \mid \mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \mathbf{x}_{i-1}, (\sigma_i^2 - \sigma_{i-1}^2)\mathbf{I}), \quad i = 1, 2, \cdots, N.$$

Here we assume $\sigma_0 = 0$ to simplify notations. Following Ho et al. (2020), we can compute

$$q(\mathbf{x}_{i-1} \mid \mathbf{x}_i, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{i-1}; \frac{\sigma_{i-1}^2}{\sigma_i^2}\mathbf{x}_i + \left(1 - \frac{\sigma_{i-1}^2}{\sigma_i^2}\right)\mathbf{x}_0, \frac{\sigma_{i-1}^2(\sigma_i^2 - \sigma_{i-1}^2)}{\sigma_i^2}\mathbf{I}\right).$$

If we parameterize the reverse transition kernel as $p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_i, i), \tau_i^2\mathbf{I})$, then

$$\begin{aligned}
L_{t-1} &= \mathbb{E}_q[D_{\mathrm{KL}}(q(\mathbf{x}_{i-1} \mid \mathbf{x}_i, \mathbf{x}_0)) \parallel p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)] \\
&= \mathbb{E}_q\left[\frac{1}{2\tau_i^2}\left\|\frac{\sigma_{i-1}^2}{\sigma_i^2}\mathbf{x}_i + \left(1 - \frac{\sigma_{i-1}^2}{\sigma_i^2}\right)\mathbf{x}_0 - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)\right\|_2^2\right] + C \\
&= \mathbb{E}_{\mathbf{x}_0, \mathbf{z}}\left[\frac{1}{2\tau_i^2}\left\|\mathbf{x}_i(\mathbf{x}_0, \mathbf{z}) - \frac{\sigma_i^2 - \sigma_{i-1}^2}{\sigma_i}\mathbf{z} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_i(\mathbf{x}_0, \mathbf{z}), i)\right\|_2^2\right] + C,
\end{aligned}$$

where L_{t-1} is one representative term in the ELBO objective (see Eq. (8) in Ho et al. (2020)), C is a constant that does not depend on $\boldsymbol{\theta}$, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{x}_i(\mathbf{x}_0, \mathbf{z}) = \mathbf{x}_0 + \sigma_i \mathbf{z}$. We can therefore parameterize $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)$ via

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_i, i) = \mathbf{x}_i + (\sigma_i^2 - \sigma_{i-1}^2)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i),$$

where $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_i, i)$ is to estimate $\mathbf{z}/\sigma_i$. As in Ho et al. (2020), we let $\tau_i = \sqrt{\frac{\sigma_{i-1}^2(\sigma_i^2 - \sigma_{i-1}^2)}{\sigma_i^2}}$. Through ancestral sampling on $\prod_{i=1}^N p_{\boldsymbol{\theta}}(\mathbf{x}_{i-1} \mid \mathbf{x}_i)$, we obtain the following iteration rule

$$\mathbf{x}_{i-1} = \mathbf{x}_i + (\sigma_i^2 - \sigma_{i-1}^2)\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_i, i) + \sqrt{\frac{\sigma_{i-1}^2(\sigma_i^2 - \sigma_{i-1}^2)}{\sigma_i^2}}\mathbf{z}_i, i = 1, 2, \cdots, N, \tag{42}$$

where $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_N^2\mathbf{I})$, $\boldsymbol{\theta}^*$ denotes the optimal parameter of $\mathbf{s}_{\boldsymbol{\theta}}$, and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We call Eq. (42) the ancestral sampling method for SMLD models.

G ADDITIONAL DETAILS ON PREDICTOR-CORRECTOR SAMPLERS

Training We use the same architecture in Ho et al. (2020) for our score-based models. For the VE SDE, we train a model with the original SMLD objective in Eq. (1); similarly for the VP SDE, we use the original DDPM objective in Eq. (3). We apply a total number of 1000 noise scales for training both models. For results in Fig. 3, we train an NCSN++ model (definition in Appendix H) on 256×256 LSUN bedroom and church_outdoor (Yu et al., 2015) datasets with the VE SDE and our continuous objective Eq. (7).

The corrector algorithms We take the schedule of annealed Langevin dynamics in Song & Ermon (2019), but re-frame it with slight modifications in order to get better interpretability and empirical performance. We provide the corrector algorithms in Algs. 3 and 4 respectively, where we call r the "signal-to-noise" ratio. We determine the step size ϵ using the norm of the Gaussian noise $\|\mathbf{z}\|_2$, norm of the score-based model $\|\mathbf{s}_{\boldsymbol{\theta}^*}\|_2$ and the signal-to-noise ratio r. When sampling a large batch of samples together, we replace the norm $\|\cdot\|_2$ with the average norm across the mini-batch. When the batch size is small, we suggest replacing $\|\mathbf{z}\|_2$ with $\sqrt{d}$, where d represents the dimensionality of $\mathbf{z}$.

Algorithm 3 Corrector algorithm (VE SDE).

Require: $\{\sigma_i\}_{i=1}^N, r, N, M.$
$\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{max}}^2 \mathbf{I})$
for $i \leftarrow N$ to 1 **do**
  for $j \leftarrow 1$ to M **do**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{g} \leftarrow \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i^{j-1}, \sigma_i)$
    $\epsilon \leftarrow 2(r \left\|\mathbf{z}\right\|_2 / \left\|\mathbf{g}\right\|_2)^2$
    $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^{j-1} + \epsilon\, \mathbf{g} + \sqrt{2\epsilon}\, \mathbf{z}$
  $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$
return $\mathbf{x}_0^0$

Algorithm 4 Corrector algorithm (VP SDE).

Require: $\{\beta_i\}_{i=1}^N, \{\alpha_i\}_{i=1}^N, r, N, M.$
$\mathbf{x}_N^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i \leftarrow N$ to 1 **do**
  for $j \leftarrow 1$ to M **do**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{g} \leftarrow \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i^{j-1}, i)$
    $\epsilon \leftarrow 2\alpha_i(r \left\|\mathbf{z}\right\|_2 / \left\|\mathbf{g}\right\|_2)^2$
    $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^{j-1} + \epsilon\, \mathbf{g} + \sqrt{2\epsilon}\, \mathbf{z}$
  $\mathbf{x}_{i-1}^0 \leftarrow \mathbf{x}_i^M$
return $\mathbf{x}_0^0$

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 single denoising step at the end of sampling, using Tweedie's formula (Efron, 2011).

Ad-hoc interpolation methods for noise scales Models in this experiment are all trained with 1000 noise scales. To get results for P2000 (predictor-only sampler using 2000 steps) which requires 2000 noise scales, we need to interpolate between 1000 noise scales at test time. The specific architecture of the noise-conditional score-based model in Ho et al. (2020) uses sinusoidal positional embeddings for conditioning on integer time steps. This allows us to interpolate between noise scales at test time in an ad-hoc way (while it is hard to do so for other architectures like the one in Song & Ermon (2019)). Specifically, for SMLD models, we keep σ_{min} and σ_{max} fixed and double the number of time steps. For DDPM models, we halve $\bar{\beta}_{\text{min}}$ and $\bar{\beta}_{\text{max}}$ before doubling the number of time steps. Suppose $\{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, i)\}_{i=0}^{N-1}$ is a score-based model trained on N time steps, and let $\{\mathbf{s}'_{\boldsymbol{\theta}}(\mathbf{x}, i)\}_{i=0}^{2N-1}$ denote the corresponding interpolated score-based model at $2N$ time steps. We test two different interpolation strategies for time steps: linear interpolation where $\mathbf{s}'_{\boldsymbol{\theta}}(\mathbf{x}, i) = \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, i/2)$ and rounding interpolation where $\mathbf{s}'_{\boldsymbol{\theta}}(\mathbf{x}, i) = \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \lfloor i/2 \rfloor)$. We provide results with rounding interpolation in Tab. 1, and give results of linear interpolation in Tab. 4. We observe that different interpolation methods result in performance differences but maintain the general trend of predictor-corrector methods performing on par or better than predictor-only or corrector-only samplers.

Hyper-parameters of the samplers For Predictor-Corrector and corrector-only samplers on CIFAR-10, we search for the best signal-to-noise ratio (r) over a grid that increments at 0.01. We report the best r in Tab. 5. For LSUN bedroom/church_outdoor, we fix r to 0.075. Unless otherwise noted, we use one corrector step per noise scale for all PC samplers. We use two corrector steps per noise scale for corrector-only samplers on CIFAR-10. We always use a batch size of 128 for training. For sample generation, the batch size is 1024 on CIFAR-10 and 8 on LSUN bedroom/church_outdoor.

H ARCHITECTURAL IMPROVEMENTS 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. Code and checkpoints will be open sourced.

Table 4: Comparing different samplers on CIFAR-10, where "P2000" uses linear interpolation between noise scales. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs.

FID↓ Sampler / Predictor	Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	4.98 ± .06	4.88 ± .06		**3.62 ± .03**	3.24 ± .02	3.24 ± .02		**3.21 ± .02**
reverse diffusion	4.79 ± .07	4.74 ± .08	20.43 ± .07	**3.60 ± .02**	3.21 ± .02	3.19 ± .02	19.06 ± .06	**3.18 ± .01**
probability flow	15.41 ± .15	10.54 ± .08		**3.51 ± .04**	3.59 ± .04	3.23 ± .03		**3.06 ± .03**

Table 5: Optimal signal-to-noise ratios of different samplers. "P1000" or "P2000": predictor-only samplers using 1000 or 2000 steps. "C2000": corrector-only samplers using 2000 steps. "PC1000": PC samplers using 1000 predictor and 1000 corrector steps.

r Sampler / Predictor	VE SDE (SMLD)				VP SDE (DDPM)			
	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
ancestral sampling	-	-		0.17	-	-		0.01
reverse diffusion	-	-	0.22	0.16	-	-	0.27	0.01
probability flow	-	-		0.17	-	-		0.04

H.1 Settings

We consider two datasets: 32×32 CIFAR-10 (Krizhevsky et al., 2009) and 64×64 CelebA (Liu et al., 2015), which is pre-processed following Song & Ermon (2020). All models are trained with 1.3M iterations, and we save one checkpoint per 50k iterations. We compare different configurations based on their FID scores averaged over 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) and use a batch size of 128.

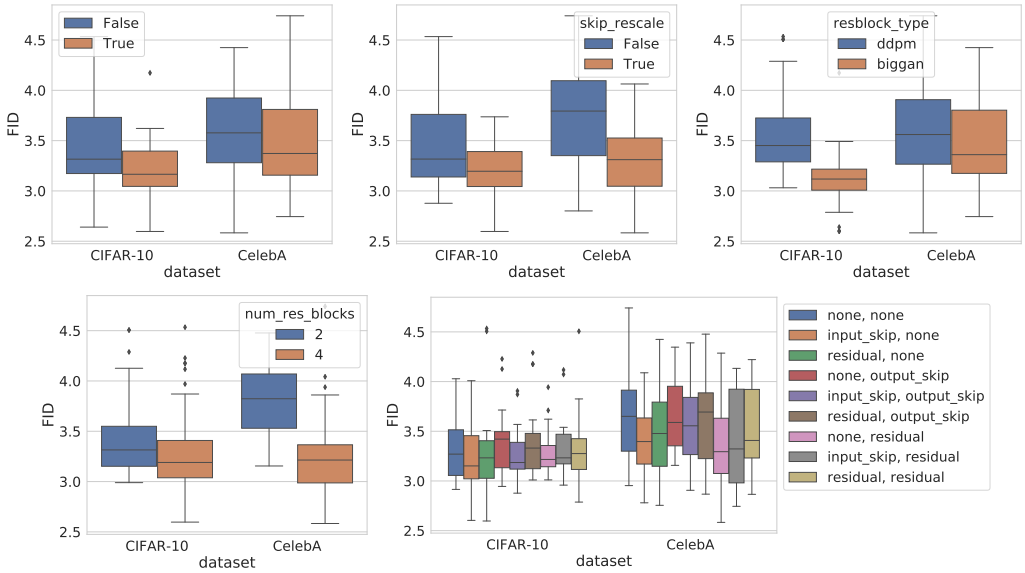


Figure 10: The effects of different architecture components for score-based models trained with VE perturbations.

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 hyper-parameters 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., 2018), 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". These progressive architectures are defined and implemented according to StyleGAN-2.

We also tested equalized learning rates, a trick used in very successful models like ProgressiveGAN (Karras et al., 2018) 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.

For sampling from models with VE perturbations, we use the PC sampler discretized at 1000 noise scales with one Langevin step per scale and a signal-of-noise ratio of 0.16. For VP perturbations, we use the reverse diffusion sampler discretized at 1000 noise scales without correctors.

H.2 Results on CIFAR-10

We provide the architecture search results in Fig. 10. The box plots demonstrate the importance of each component when other components can vary 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 (Song & Ermon, 2019; 2020).

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. (2018); 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 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 on 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 upon conditioning on continuous time variables, we change positional embeddings, the layers in Ho et al. (2020) for conditioning on discrete time steps, to random Fourier feature embeddings, as advocated in Tancik et al. (2020). The scale parameter of these random Fourier feature embeddings is set to 16. We additionally train the model with our continuous objective Eq. (7). 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 Tab. 3.

H.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., 2018), a task that was previously only achievable by some GAN models 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. (7)) for around 700k iterations. We use the PC sampler discretized at 2000 steps with one Langevin step per noise scale and a signal-to-noise ratio of 0.15. The scale parameter for the random Fourier feature embeddings is fixed to 4. We use the "residual" progressive architecture for the input, and "output_skip" progressive architecture for the output. We provide samples in Fig. 11. Although these samples are not perfect (*e.g.*, there are visible flaws on facial symmetry), we believe these results are encouraging and can demonstrate the scalability of our approach. Future work on more effective architectures are likely to significantly advance the performance of score-based generative models on this task.

I 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

$$\mathrm{d}\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - \nabla \cdot [\mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}] - \mathbf{G}(\mathbf{x}, t)\mathbf{G}(\mathbf{x}, t)^\mathsf{T}\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y})\}\mathrm{d}t + \mathbf{G}(\mathbf{x}, t)\mathrm{d}\bar{\mathbf{w}}. \quad (43)$$

Since $p_t(\mathbf{x}(t) \mid \mathbf{y}) \propto p_t(\mathbf{x}(t))p(\mathbf{y} \mid \mathbf{x}(t))$, the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y})$ can be computed easily by

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t)) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}(t)). \quad (44)$$

This subsumes the conditional reverse-time SDE in Eq. (13) as a special case. All sampling methods we have discussed so far can be applied to the conditional reverse-time SDE for sample generation.

I.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. (7), 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 sum of cross-entropy losses sampled at different scales. We provide a plot to show the accuracy of this classifier over noise scales in Fig. 12. The score-based model is an unconditional NCSN++ (4 blocks/resolution) in Tab. 3, and we generate samples using the PC algorithm with 2000 discretization steps. The class-conditional samples are provided in Fig. 5, and an extended set of conditional samples is given in Fig. 12.

I.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 $\mathbf{f}_{\bar{\Omega}}(\cdot, t)$ and $\mathbf{G}_{\bar{\Omega}}(\cdot, t)$ denote $\mathbf{f}(\cdot, t)$ and $\mathbf{G}(\cdot, t)$ restricted to the unknown dimensions. For VE/VP SDEs, the drift coefficient $\mathbf{f}(\cdot, t)$ is element-wise, and the diffusion coefficient $\mathbf{G}(\cdot, t)$ is diagonal. When $\mathbf{f}(\cdot, t)$ is element-wise, $\mathbf{f}_{\bar{\Omega}}(\cdot, t)$ denotes the same element-wise function applied only to the unknown dimensions. When $\mathbf{G}(\cdot, t)$ is diagonal, $\mathbf{G}_{\bar{\Omega}}(\cdot, t)$ denotes the sub-matrix restricted to unknown dimensions.

Figure 11: Samples on 1024×1024 CelebA-HQ from continuously trained NCSN++.

Figure 12: 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.

For imputation, our goal is to sample from $p(\bar{\Omega}(\mathbf{x}(0)) \mid \Omega(\mathbf{x}(0)) = \mathbf{y})$. Define a new diffusion process $\mathbf{z}(t) = \bar{\Omega}(\mathbf{x}(t))$, and note that the SDE for $\mathbf{z}(t)$ can be written as

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

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

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

Although $p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y})$ is in general intractable, it can be approximated. Let $A \triangleq \mathbb{1}\{\Omega(\mathbf{x}(0)) = \mathbf{y}\}$, where $\mathbb{1}\{\cdot\}$ denotes the indicator function. We have

$$\begin{aligned}p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y}) &= \int p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)), A) p_t(\Omega(\mathbf{x}(t)) \mid A)\mathrm{d}\Omega(\mathbf{x}(t)) \\ &= \mathbb{E}_{p_t(\Omega(\mathbf{x}(t))|A)}[p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)), A)] \\ &\approx \mathbb{E}_{p_t(\Omega(\mathbf{x}(t))|A)}[p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(t)))] \\ &\approx p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t))),\end{aligned}$$

where $\hat{\Omega}(\mathbf{x}(t))$ is a random sample from $p_t(\Omega(\mathbf{x}(t)) \mid A)$, which is typically a tractable distribution. Therefore,

$$\begin{aligned}\nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \Omega(\mathbf{x}(0)) = \mathbf{y}) &\approx \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t))) \\ &= \nabla_{\mathbf{z}} \log p_t([\mathbf{z}(t); \hat{\Omega}(\mathbf{x}(t))]),\end{aligned}$$

where $[\mathbf{z}(t); \hat{\Omega}(\mathbf{x}(t))]$ denotes a vector $\mathbf{u}(t)$ such that $\Omega(\mathbf{u}(t)) = \hat{\Omega}(\mathbf{x}(t))$ and $\bar{\Omega}(\mathbf{u}(t)) = \mathbf{z}(t)$, and the identity holds because $\nabla_{\mathbf{z}} \log p_t([\mathbf{z}(t); \hat{\Omega}(\mathbf{x}(t))]) = \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t))) + \nabla_{\mathbf{z}} \log p(\hat{\Omega}(\mathbf{x}(t))) = \nabla_{\mathbf{z}} \log p_t(\mathbf{z}(t) \mid \hat{\Omega}(\mathbf{x}(t)))$.

We provided an extended set of inpainting results in Figs. 13 and 14.

I.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 standard Wiener process in the transformed space, allowing us to build an SDE and use the same imputation method in Appendix I.2.

We provide an extended set of colorization results in Figs. 15 and 16.

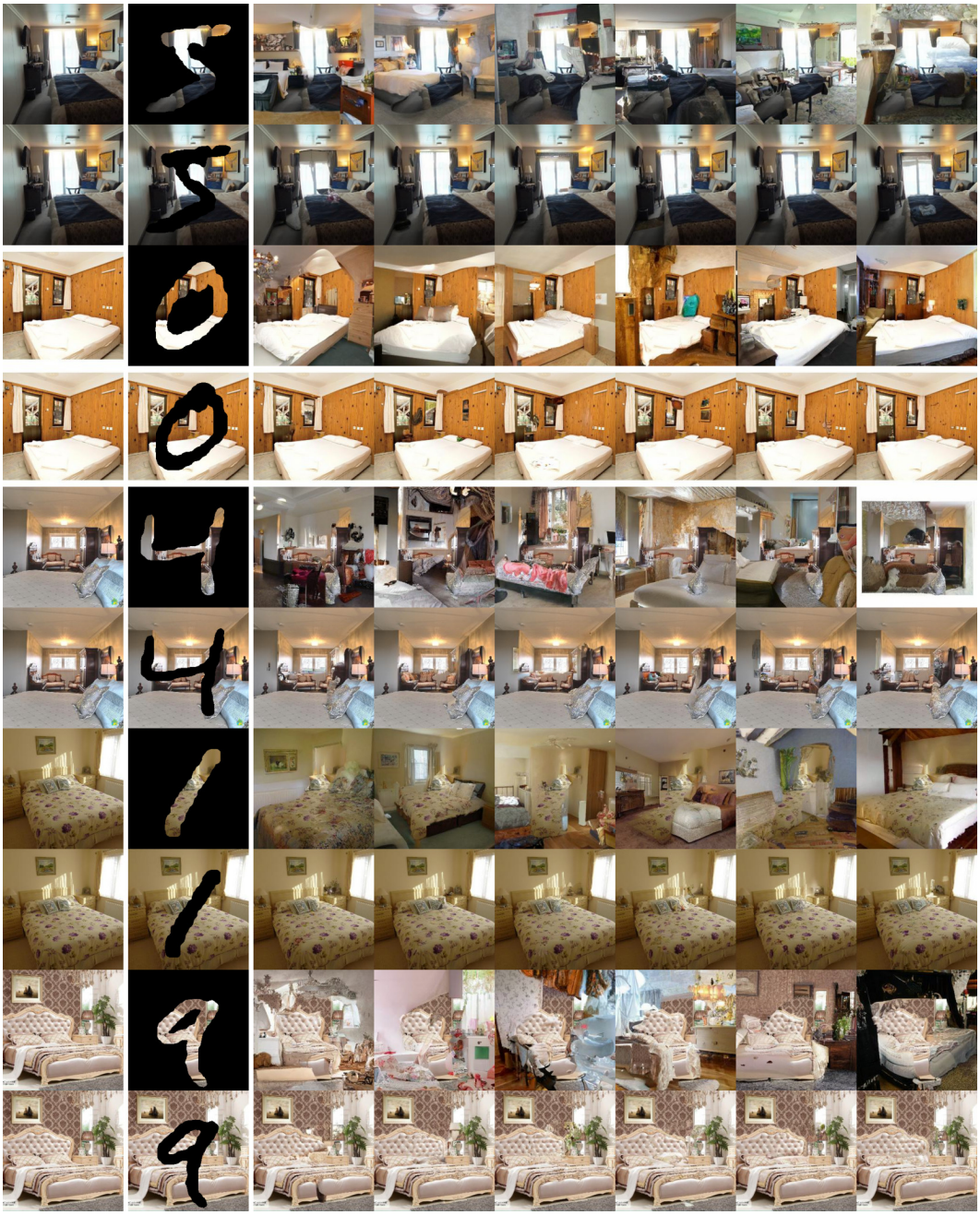


Figure 13: Extended inpainting results for 256×256 bedroom images.

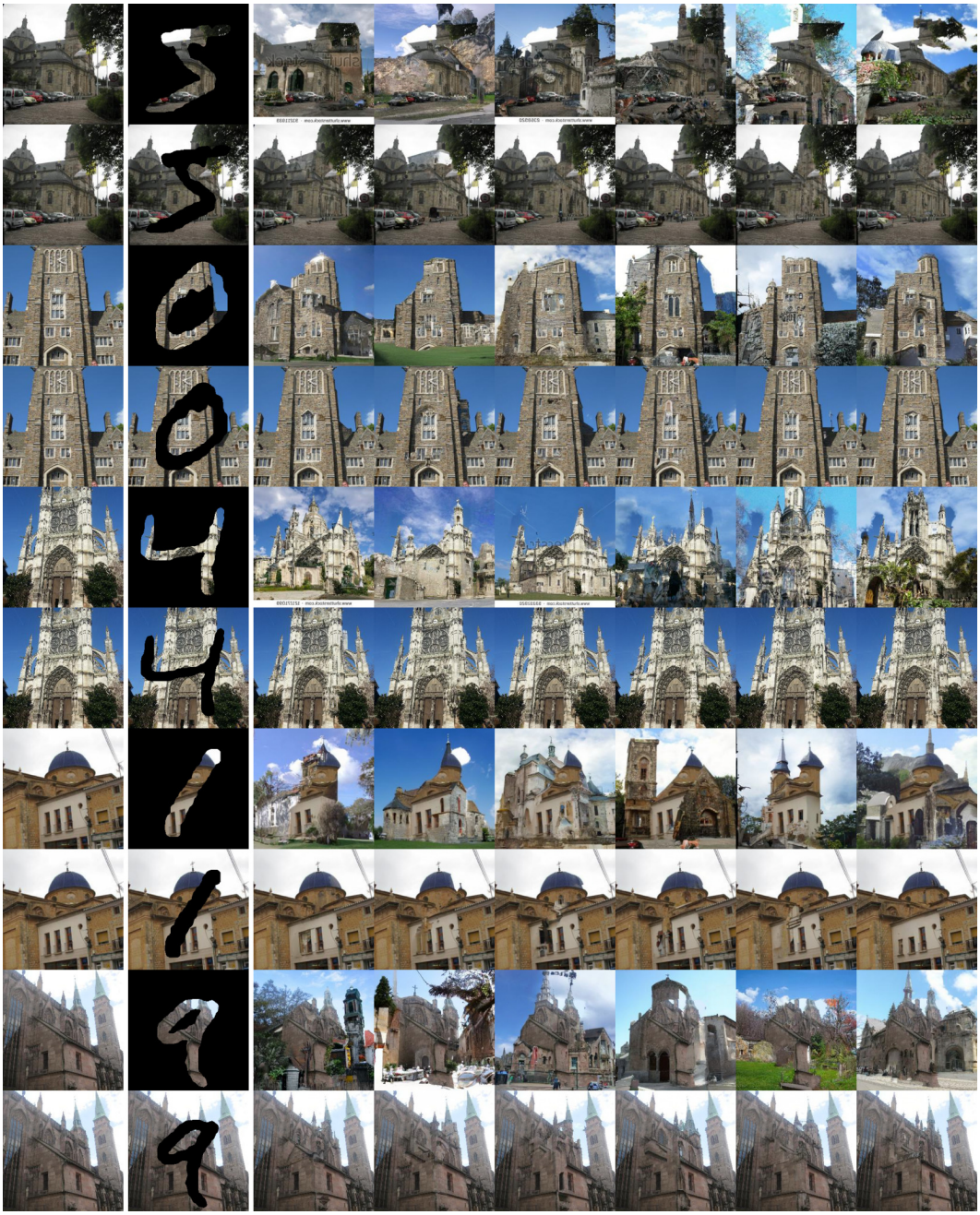


Figure 14: Extended inpainting results for 256×256 church images.

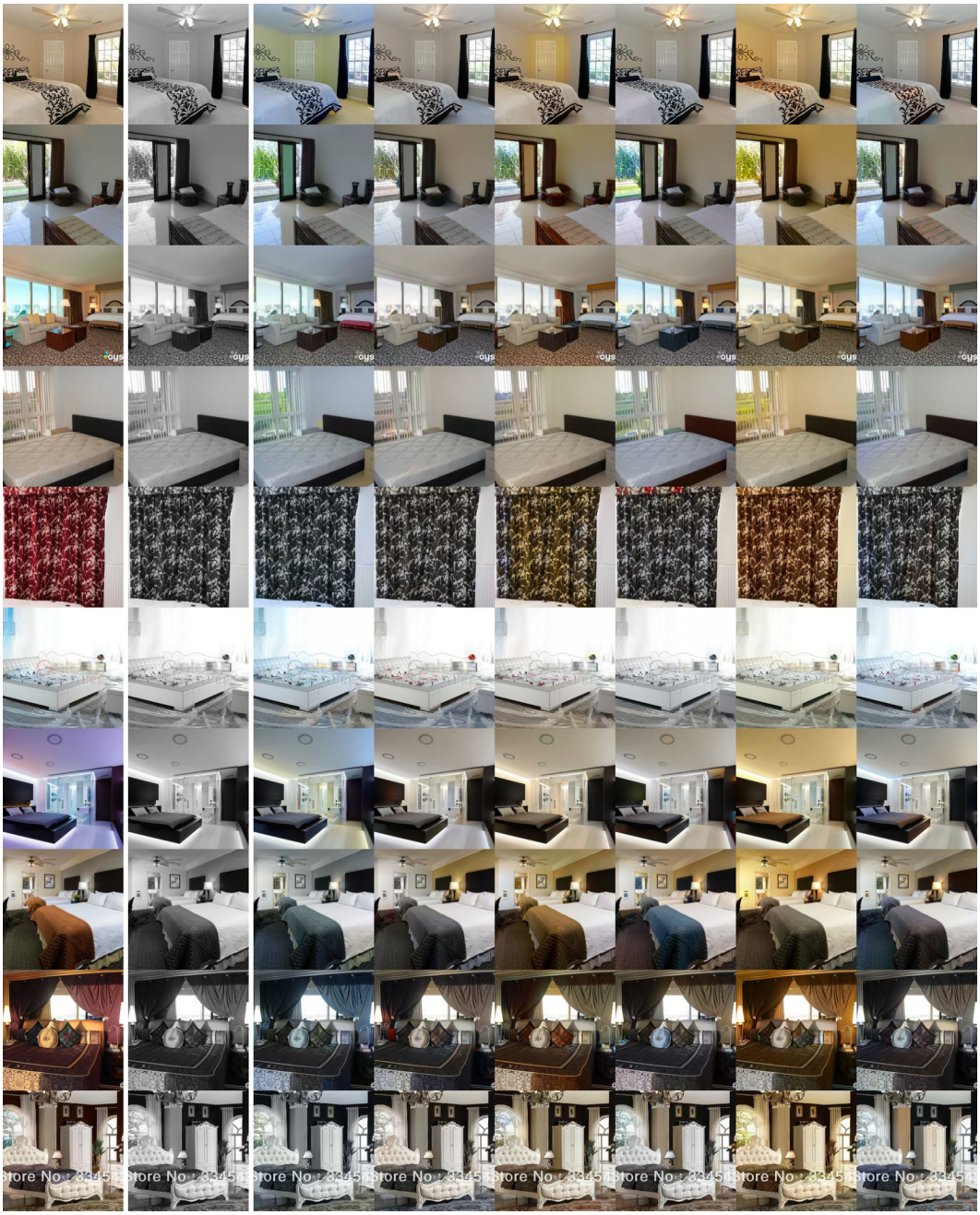


Figure 15: Extended colorization results for 256×256 bedroom images.

Figure 16: Extended colorization results for 256×256 church images.