SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

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 score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. 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, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated 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 2.99 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 with respect to data) 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; Du & Mordatch, 2019), 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

*Work partially done during an internship at Google Brain.

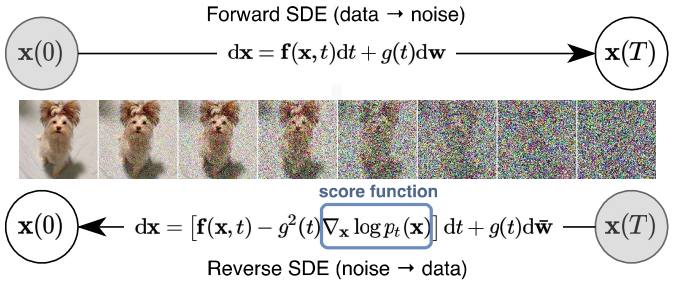

Figure 1: **Solving a reverse-time SDE yields a score-based generative model.** Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_{\mathbf{x}} \log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

et al., 2020). To enable new sampling methods and further extend the capabilities of score-based generative models, we propose a unified framework that generalizes previous approaches through the lens of stochastic differential equations (SDEs).

Specifically, 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 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 and likelihood computation: 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) and HMC (Neal et al., 2011); 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 *fast adaptive sampling* via black-box ODE solvers, *flexible data manipulation* via latent codes, a *uniquely identifiable encoding*, and notably, *exact likelihood computation*.

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 estimated from *unconditional* scores. This enables applications such as class-conditional generation, image inpainting, colorization and other inverse problems, all achievable using a single unconditional score-based model without re-training.

Unified framework: Our framework provides a unified way to explore and tune various SDEs for improving score-based generative models. The methods of SMLD and DDPM can be amalgamated into our framework as discretizations of two separate SDEs. Although DDPM (Ho et al., 2020) was recently reported to achieve higher sample quality than SMLD (Song & Ermon, 2019; 2020), we show that with better architectures and new sampling algorithms allowed by our framework, the latter can catch up—it achieves new state-of-the-art Inception score (9.89) and FID score (2.20) on CIFAR-10, as well as high-fidelity generation of 1024×1024 images for the first time from a score-based model. In addition, we propose a new SDE under our framework that achieves a likelihood value of 2.99 bits/dim on uniformly dequantized CIFAR-10 images, setting a new record on this task.

2 Background

2.1 Denoising score matching with Langevin dynamics (SMLD)

Let $p_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) := \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2\mathbf{I})$ be a perturbation kernel, and $p_\sigma(\tilde{\mathbf{x}}) := \int p_{\text{data}}(\mathbf{x}) p_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$, 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:

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

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, $\mathbf{x}_1^M$ 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 $\alpha_i := \prod_{j=1}^{i}(1-\beta_j)$. Similar to SMLD, we can denote the perturbed data distribution as $p_{\alpha_i}(\tilde{\mathbf{x}}) := \int p_{\text{data}}(\mathbf{x}) p_{\alpha_i}(\tilde{\mathbf{x}} \mid \mathbf{x}) \mathrm{d}\mathbf{x}$. The noise scales are prescribed such that $\mathbf{x}_N$ is approximately distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. 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 L_{simple} in Ho et al. (2020), written in a 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, $\nabla_{\mathbf{x}} \log p_{\alpha_i}(\mathbf{x})$. Notably, 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:

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

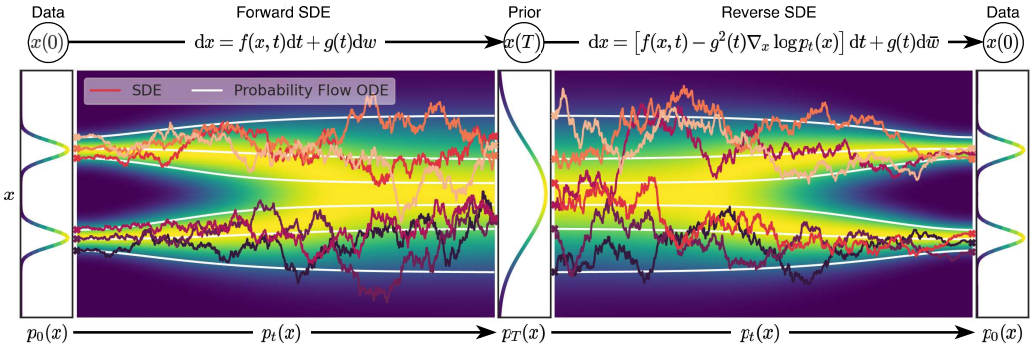

Figure 2: **Overview of score-based generative modeling through SDEs**. We can map data to a noise distribution (the prior) with an SDE (Section 3.1), and reverse this SDE for generative modeling (Section 3.2). We can also reverse the associated probability flow ODE (Section 4.3), which yields a deterministic 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).

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 as long as the coefficients are globally Lipschitz in both state and time (Øksendal, 2003). We hereafter denote by $p_t(\mathbf{x})$ the 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 fixed 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 several examples later in Section 3.4 that are derived from continuous generalizations of SMLD and DDPM.

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):

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

Here $\lambda : [0, T] \to \mathbb{R}_{>0}$ is a positive weighting function, t is uniformly sampled 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 $\lambda \propto 1/\mathbb{E}\big[\big\| \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \big\|_2^2 \big]$. 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 typically 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 bypasses the computation of $\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ (see Appendix A).

3.4 Examples: VE, VP SDEs and beyond

The noise perturbations used in SMLD and DDPM can be regarded as discretizations of two different SDEs. 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 Eqs. (9) and (11). 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 a fixed variance of one when the initial distribution has unit 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.

Inspired by the VP SDE, we propose a new type of SDEs which perform particularly well on likelihoods (see Section 4.3), given by

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)\mathrm{d}s})}\mathrm{d}\mathbf{w}. \tag{12}$$

When using the same $\beta(t)$ and starting from the same initial distribution, the variance of the stochastic process induced by Eq. (12) is always bounded by the VP SDE at every intermediate time step (proof in Appendix B). For this reason, we name Eq. (12) the sub-VP SDE.

Since VE, VP and sub-VP SDEs all have affine drift coefficients, their perturbation kernels $p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))$ are all Gaussian and can be computed in closed-forms, 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 $\mathbf{s}_{\boldsymbol{\theta}}$, we can use it to construct the reverse-time SDE and then simulate it with numerical approaches to generate samples from p_0.

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	4.88 ± .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**

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). Deriving the ancestral sampling rules for new SDEs, however, can be non-trivial. To remedy this, we propose *reverse diffusion samplers* (details in Appendix E), which discretize the reverse-time SDE in the same way as the forward one, and thus can be readily derived given the forward discretization. As shown in Table 1, reverse diffusion samplers perform slightly better than ancestral sampling for both SMLD and DDPM models on CIFAR-10 (DDPM-type ancestral sampling is also applicable to SMLD models, see Appendix F.)

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.

Specifically, at each time step, the numerical SDE solver first gives an estimate of the sample at the next time step, playing the role of a "predictor". Then, 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, a family of numerical continuation techniques for solving systems of equations (Allgower & Georg, 2012), and we similarly name our hybrid sampling algorithms *Predictor-Corrector* (PC) samplers. Please find pseudo-code and a complete description in Appendix G. PC samplers generalize the original sampling methods of SMLD and DDPM: the former uses an identity function as the predictor and annealed Langevin dynamics as the corrector, while the latter uses ancestral sampling as the predictor and identity as the corrector.

We test PC samplers on SMLD and DDPM models (see Algorithms 2 and 3 in Appendix G) trained with original discrete objectives given by Eqs. (1) and (3). This exhibits the compatibility of PC samplers to score-based models trained with a fixed number of noise scales. We summarize the performance of different samplers in Table 1, where probability flow is a predictor to be discussed in Section 4.3. Detailed experimental settings and additional results are given in Appendix G. We observe that our reverse diffusion sampler always outperform ancestral sampling, and corrector-only methods (C2000) perform worse than other competitors (P2000, PC1000) with the same computation (In fact, we need way more corrector steps per noise scale, and thus more computation, to match the performance of other samplers.) For all predictors, adding one corrector step for each predictor step (PC1000) doubles computation but always improves sample quality (against P1000). Moreover, it is typically better than doubling the number of predictor steps without adding a corrector (P2000), where we have to interpolate between noise scales in an ad hoc manner (detailed in Appendix G) for SMLD/DDPM models. In Fig. 9 (Appendix G), we additionally provide qualitative comparison for

Table 2: NLLs and FIDs (ODE) 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	3.28	3.37
DDPM cont. (VP)	3.21	3.69
DDPM cont. (sub-VP)	3.05	3.56
DDPM++ cont. (VP)	3.16	3.93
DDPM++ cont. (sub-VP)	3.02	3.16
DDPM++ cont. (deep, VP)	3.13	3.08
DDPM++ cont. (deep, sub-VP)	**2.99**	**2.92**

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 ± .12
NCSNv2 (Song & Ermon, 2020)	10.87	8.40 ± .07
DDPM (Ho et al., 2020)	3.17	9.46 ± .11
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

models trained with the continuous objective Eq. (7) on 256×256 LSUN images and the VE SDE, where PC samplers clearly surpass predictor-only samplers under comparable computation, when using a proper number of corrector steps.

4.3 Probability flow and connection 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 $\{p_t(\mathbf{x})\}_{t=0}^{T}$ as the SDE. This deterministic process satisfies an ODE (more details in Appendix D.1):

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t, \tag{13}$$

which can be determined from the SDE once scores are known. We name the ODE in Eq. (13) the *probability flow ODE*. When the score function is approximated by the time-dependent score-based model, which is typically a neural network, this is an example of a neural ODE (Chen et al., 2018).

Exact likelihood computation Leveraging the connection to neural ODEs, we can compute the density defined by Eq. (13) 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 Table 2. We compute log-likelihoods on uniformly dequantized data, and only compare to models evaluated in the same way (omitting models evaluated with variational dequantization (Ho et al., 2019) or discrete data), except for DDPM (L/L_{simple}) whose ELBO values (annotated with *) are reported on discrete data. Main results: (i) For the same DDPM model in Ho et al. (2020), we obtain better bits/dim than ELBO, since our likelihoods are exact; (ii) Using the same architecture, we trained another DDPM model with the continuous objective in Eq. (7) (*i.e.*, DDPM cont.), which further improves the likelihood; (iii) With sub-VP SDEs, we always get higher likelihoods compared to VP SDEs; (iv) With improved architecture (*i.e.*, DDPM++ cont., details in Section 4.4) and the sub-VP SDE, we can set a new record bits/dim of 2.99 on uniformly dequantized CIFAR-10 even *without maximum likelihood training*.

Manipulating latent representations By integrating Eq. (13), 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; Kingma & Dhariwal, 2018), we can manipulate this latent representation for image editing, such as interpolation, and temperature scaling (see Fig. 3 and Appendix D.4).

Uniquely identifiable encoding Unlike most 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

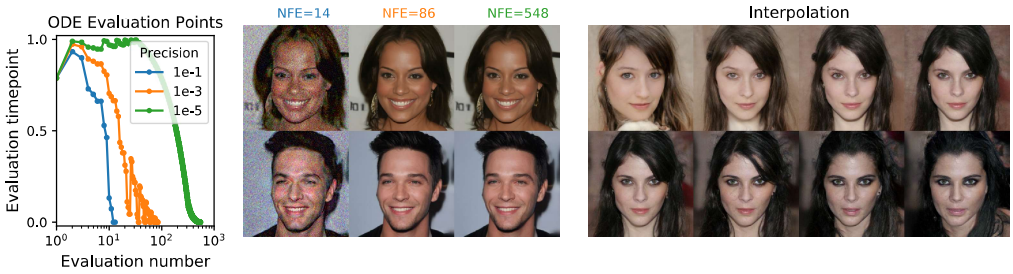

Figure 3: **Probability flow ODE enables fast sampling** with adaptive stepsizes as the numerical precision is varied (*left*), and reduces the number of score function evaluations (NFE) without harming quality (*middle*). The invertible mapping from latents to images allows for interpolations (*right*).

ODE, Eq. (13), provides the same trajectories given perfectly estimated scores. We provide additional empirical verification on this property in Appendix D.5.

Efficient sampling As with neural ODEs, we can sample $\mathbf{x}(0) \sim p_0$ by solving Eq. (13) 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 (Table 1, "probability flow sampler", details in Appendix D.3). Using a black-box ODE solver (Dormand & Prince, 1980) not only produces high quality samples (Table 2, details in Appendix D.4), but also 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. 3).

4.4 Architecture improvements

We explore several new architecture designs for score-based models using both VE and VP SDEs (details in Appendix H), where we train models with the same discrete objectives as in SMLD/DDPM. We directly transfer the architectures for VP SDEs to sub-VP SDEs due to their similarity. Our optimal architecture for the VE SDE, named NCSN++, achieves an FID of 2.45 on CIFAR-10 with PC samplers, while our optimal architecture for the VP SDE, called DDPM++, achieves 2.78.

By switching to the continuous training objective in Eq. (7), and increasing the network depth, we can further improve sample quality for all models. The resulting architectures are denoted as NCSN++ cont. and DDPM++ cont. in Table 3 for VE and VP/sub-VP SDEs respectively. Results reported in Table 3 are for the checkpoint with the smallest FID over the course of training, where samples are generated with PC samplers. In contrast, FID scores and NLL values in Table 2 are reported for the last training checkpoint, and samples are obtained with black-box ODE solvers. As shown in Table 3, VE SDEs typically provide better sample quality than VP/sub-VP SDEs, but we also empirically observe that their likelihoods are worse than VP/sub-VP SDE counterparts. This indicates that practitioners likely need to experiment with different SDEs for varying domains and architectures.

Our best model for sample quality, NCSN++ cont. (deep, VE), doubles the network depth and 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. With all improvements together, we also obtain the first set of high-fidelity samples on CelebA-HQ 1024×1024 from score-based models (see Appendix H.3). Our best model for likelihoods, DDPM++ cont. (deep, sub-VP), similarly doubles the network depth and achieves a log-likelihood of 2.99 bits/dim with the continuous objective in Eq. (7). To our best knowledge, this is the highest likelihood on uniformly dequantized CIFAR-10.

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

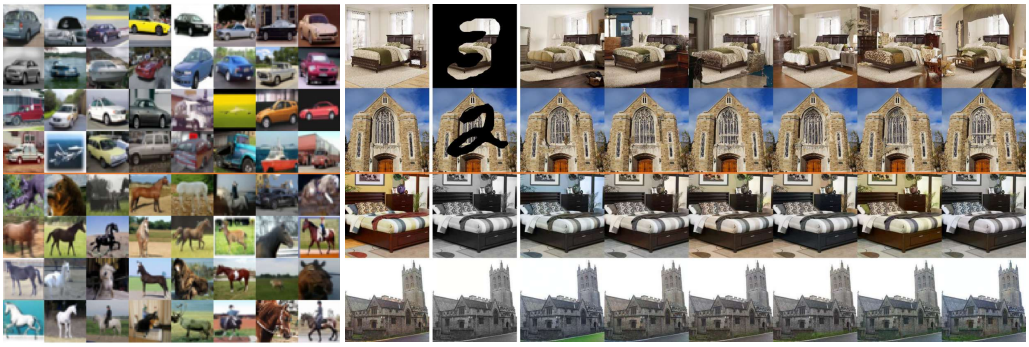

Figure 4: *Left*: Class-conditional samples on 32 × 32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right:* Inpainting (top two rows) and colorization (bottom two rows) results on 256 × 256 LSUN. First column is the original image, second column is the masked/gray-scale image, remaining columns are sampled image completions or colorizations.

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:

$$\mathrm{d}\mathbf{x} = \{\mathbf{f}(\mathbf{x}, t) - g(t)^2[\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})]\}\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}. \tag{14}$$

In general, we can use Eq. (14) to solve a large family of *inverse problems* with score-based generative models, once given an estimate of the gradient of the forward process, $\nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x}(t))$. In some cases, it is possible to train a separate model to learn the forward process $\log p_t(\mathbf{y} \mid \mathbf{x}(t))$ and compute its gradient. Otherwise, we may estimate the gradient with heuristics and domain knowledge. In Appendix I.4, we provide a broadly applicable method for obtaining such an estimate without the need of training auxiliary models.

We consider three applications of controllable generation with this approach: class-conditional generation, image imputation and colorization. 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 $p_t(\mathbf{y} \mid \mathbf{x}(t))$. We provide class-conditional CIFAR-10 samples in Fig. 4 (left), and relegate more details and results to 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. 4 (right) shows results for inpainting and colorization achieved with unconditional time-dependent score-based models.

6 Conclusion

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.

ACKNOWLEDGEMENTS

We would like to thank Nanxin Chen, Ruiqi Gao, Jonathan Ho, Kevin Murphy, Tim Salimans and Han Zhang for their insightful discussions during the course of this project. This research was partially supported by NSF (#1651565, #1522054, #1733686), ONR (N000141912145), AFOSR (FA95501910024), and TensorFlow Research Cloud. Yang Song was partially supported by the Apple PhD Fellowship in AI/ML.

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.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 3608–3618. Curran Associates, Inc., 2019.

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.