# Generative Modeling by Estimating Gradients of the Data Distribution

This blog post focuses on a promising new direction for generative modeling. We can learn score functions (gradients of log probability density functions) on a large number of noise-perturbed data distributions, then generate samples with Langevin-type sampling. The resulting generative models, often called *score-based generative models*, has several important advantages over existing model families: GAN-level sample quality without adversarial training, flexible model architectures, exact log-likelihood computation, and inverse problem solving without re-training models. In this blog post, we will show you in more detail the intuition, basic concepts, and potential applications of score-based generative models.

AUTHORS
Yang Song

AFFILIATIONS
Stanford University

PUBLISHED
May 5, 2021

## Contents

Introduction

The score function, score-based models, and score matching

Langevin dynamics

Naive score-based generative modeling and its pitfalls

Score-based generative modeling with multiple noise perturbations

Score-based generative modeling with stochastic differential equations (SDEs)

- Perturbing data with an SDE
- Reversing the SDE for sample generation
- Estimating the reverse SDE with score-based models and score matching
- How to solve the reverse SDE
- Probability flow ODE
- Controllable generation for inverse problem solving

Connection to diffusion models and others

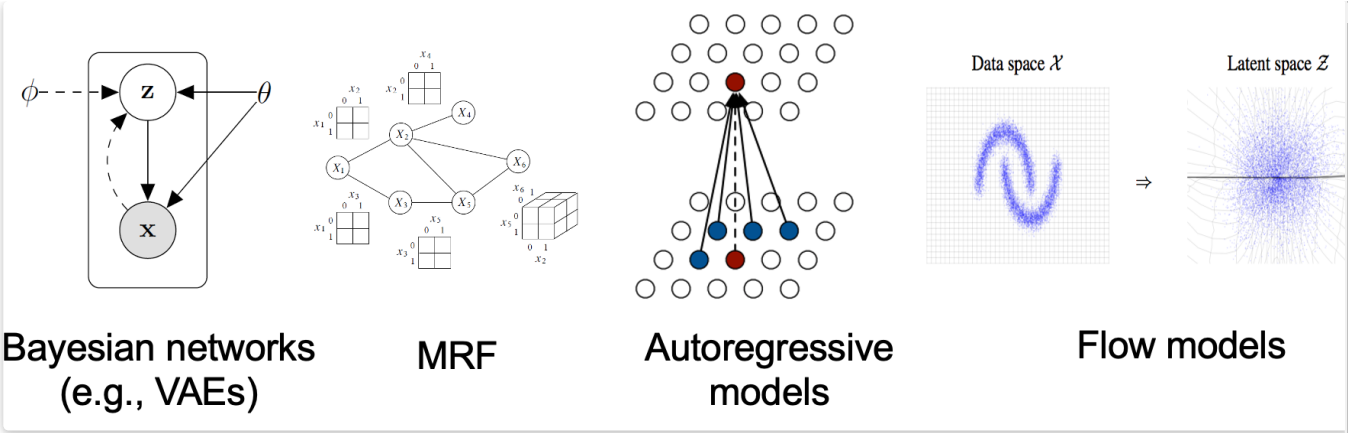
Concluding remarks

## Introduction

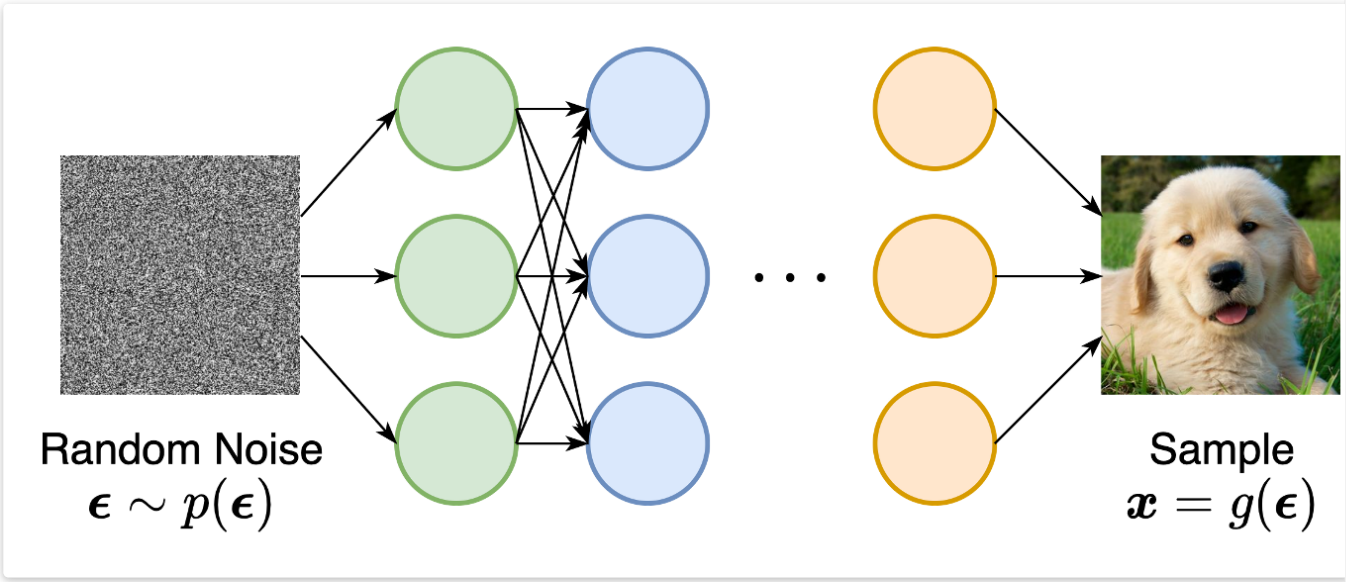
Existing generative modeling techniques can largely be grouped into two categories based on how they represent probability distributions.

1. **likelihood-based models**, which directly learn the distribution's probability density (or mass) function via (approximate) maximum likelihood. Typical likelihood-based models include autoregressive models [1, 2, 3], normalizing flow models [4, 5], energy-based models (EBMs) [6, 7], and variational auto-encoders (VAEs) [8, 9].

2. **implicit generative models** [10], where the probability distribution is implicitly represented by a model of its sampling process. The most prominent example is generative adversarial networks (GANs) [11], where new samples from the data distribution are synthesized by transforming a random Gaussian vector with a neural network.



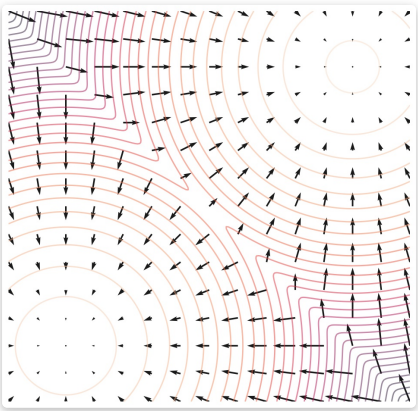

Bayesian networks, Markov random fields (MRF), autoregressive models, and normalizing flow models are all examples of likelihood-based models. All these models represent the proba density or mass function of a distribution.




GAN is an example of implicit models. It implicitly represents a distribution over all objects that can be produced by the generator network.

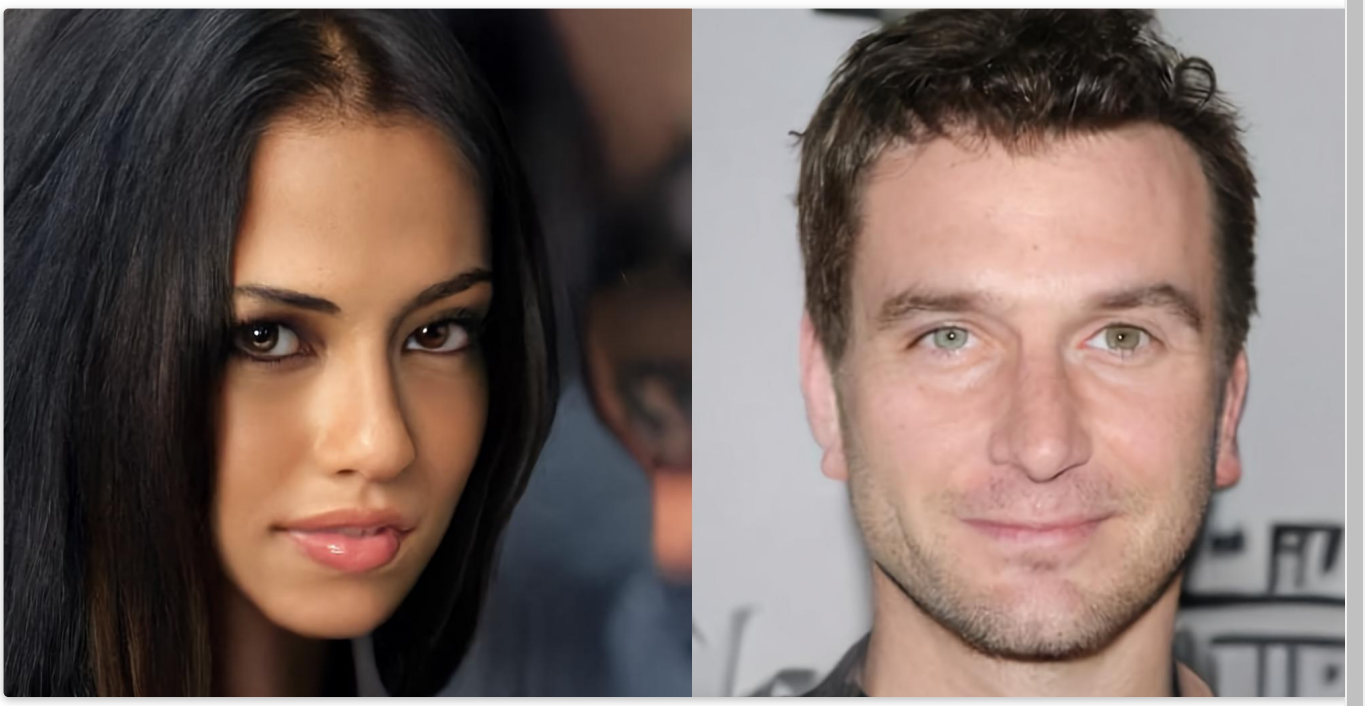
Likelihood-based models and implicit generative models, however, both have significant limitations. Likelihood-based models either require strong restrictions o the model architecture to ensure a tractable normalizing constant for likelihood computation, or must rely on surrogate objectives to approximate maximum likelihood training. Implicit generative models, on the other hand, often require adversarial training, which is notoriously unstable [12] and can lead to mode colla [13].

In this blog post, I will introduce another way to represent probability distributions that may circumvent several of these limitations. The key idea is to model *the gradient of the log probability density function*, a quantity often known as the (Stein) **score function** [14]. Such **score-based models** are not required to have a tractable normalizing constant, and can be directly learned by **score matching** [15, 16].



Score function (the vector field) and density function (contours) of a mixture of two Gaussians.

Score-based models have achieved state-of-the-art performance on many downstream tasks and applications. These tasks include, among others, image generation [17, 18, 19, 20, 21, 22] (Yes, better than GANs!), audio synthesis [23, 24, 25], shape generation [26], and music generation [27]. Moreover, score-based models have connections to normalizing flow models, therefore allowing exact likelihood computation and representation learning. Additionally, modeling and estimatin scores facilitates inverse problem solving, with applications such as image inpainting [17, 20], image colorization [20], compressive sensing, and medical image reconstruction (e.g., CT, MRI) [28].



1024 x 1024 samples generated from score-based models [20]

This post aims to show you the motivation and intuition of score-based generative modeling, as well as its basic concepts, properties and applications.

## The score function, score-based models, and score matching

Suppose we are given a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, where each point is drawn independently from an underlying data distribution $p(\mathbf{x})$. Given this dataset, the goal of generative modeling is to fit a model to the data distribution such that we can synthesize new data points at will by sampling from the distribution.

In order to build such a generative model, we first need a way to represent a probability distribution. One such way, as in likelihood-based models, is to directly model the probability density function (p.d.f.) or probability mass function (p.m.f.). Let $f_\theta(\mathbf{x}) \in \mathbb{R}$ be a real-valued function parameterized by a learnable parameter $\theta$. We can define a p.d.f. [1] via

$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta},$$

where $Z_\theta > 0$ is a normalizing constant dependent on $\theta$, such that $\int p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = 1$. Here the function $f_\theta(\mathbf{x})$ is often called an unnormalized probabilistic mod or energy-based model [7].

We can train $p_\theta(\mathbf{x})$ by maximizing the log-likelihood of the data

$$\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i).$$

However, equation (2) requires $p_\theta(\mathbf{x})$ to be a normalized probability density function. This is undesirable because in order to compute $p_\theta(\mathbf{x})$, we must evaluat the normalizing constant $Z_\theta$—a typically intractable quantity for any general $f_\theta(\mathbf{x})$. Thus to make maximum likelihood training feasible, likelihood-based model must either restrict their model architectures (e.g., causal convolutions in autoregressive models, invertible networks in normalizing flow models) to make $Z_\theta$ tractable, or approximate the normalizing constant (e.g., variational inference in VAEs, or MCMC sampling used in contrastive divergence [29]) which may be computationally expensive.

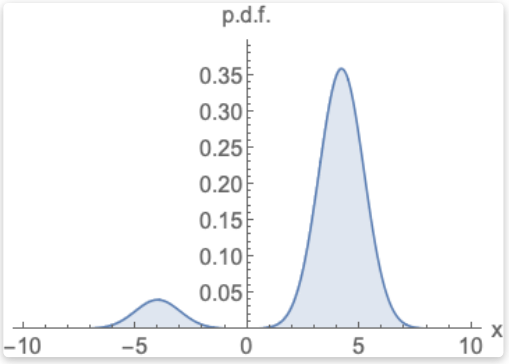
By modeling the score function instead of the density function, we can sidestep the difficulty of intractable normalizing constants. The **score function** of a distribution $p(\mathbf{x})$ is defined as

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}),$$

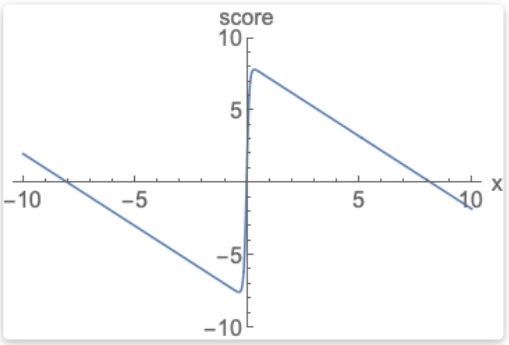
and a model for the score function is called a **score-based model** [17], which we denote as $\mathbf{s}_\theta(\mathbf{x})$. The score-based model is learned such that $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$, and can be parameterized without worrying about the normalizing constant. For example, we can easily parameterize a score-based m

$$\mathbf{s}_\theta(\mathbf{x}) = \nabla_\mathbf{x} \log p_\theta(\mathbf{x}) = -\nabla_\mathbf{x} f_\theta(\mathbf{x}) - \underbrace{\nabla_\mathbf{x} \log Z_\theta}_{=0} = -\nabla_\mathbf{x} f_\theta(\mathbf{x}).$$

Note that the score-based model $\mathbf{s}_\theta(\mathbf{x})$ is independent of the normalizing constant $Z_\theta$! This significantly expands the family of models that we can tractably u since we don't need any special architectures to make the normalizing constant tractable.




Parameterizing probability density functions. No matter how you change the model family and parameters, it has to be normalized (area under the curve must integrate to one).




Parameterizing score functions. No need to worry about normalization.

Similar to likelihood-based models, we can train score-based models by minimizing the **Fisher divergence** [2] between the model and the data distributions, defin as

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_\mathbf{x} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$$

Intuitively, the Fisher divergence compares the squared $\ell_2$ distance between the ground-truth data score and the score-based model. Directly computing this divergence, however, is infeasible because it requires access to the unknown data score $\nabla_\mathbf{x} \log p(\mathbf{x})$. Fortunately, there exists a family of methods called **scor matching** [3] [15, 16, 30] that minimize the Fisher divergence without knowledge of the ground-truth data score. Score matching objectives can directly be estimate on a dataset and optimized with stochastic gradient descent, analogous to the log-likelihood objective for training likelihood-based models (with known normalizing constants). We can train the score-based model by minimizing a score matching objective, **without requiring adversarial optimization**.

Additionally, using the score matching objective gives us a considerable amount of modeling flexibility. The Fisher divergence itself does not require $\mathbf{s}_\theta(\mathbf{x})$ to be actual score function of any normalized distribution—it simply compares the $\ell_2$ distance between the ground-truth data score and the score-based model, with additional assumptions on the form of $\mathbf{s}_\theta(\mathbf{x})$. In fact, the only requirement on the score-based model is that it should be a vector-valued function with the same input and output dimensionality, which is easy to satisfy in practice.

As a brief summary, we can represent a distribution by modeling its score function, which can be estimated by training a score-based model of free-form architectures with score matching.

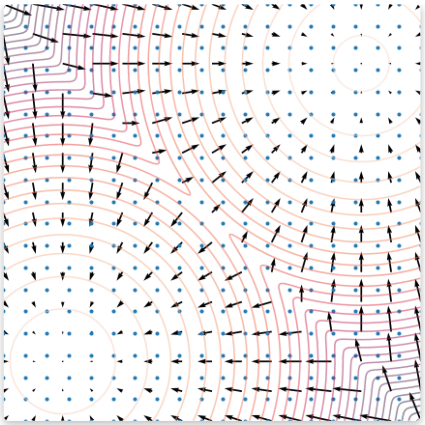
## Langevin dynamics

Once we have trained a score-based model $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$, we can use an iterative procedure called **Langevin dynamics** [31, 32] to draw samples from it

Langevin dynamics provides an MCMC procedure to sample from a distribution $p(\mathbf{x})$ using only its score function $\nabla_\mathbf{x} \log p(\mathbf{x})$. Specifically, it initializes the ch from an arbitrary prior distribution $\mathbf{x}_0 \sim \pi(\mathbf{x})$, and then iterates the following

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_\mathbf{x} \log p(\mathbf{x}) + \sqrt{2\epsilon}\, \mathbf{z}_i, \quad i = 0, 1, \cdots, K,$$

where $\mathbf{z}_i \sim \mathcal{N}(0, I)$. When $\epsilon \to 0$ and $K \to \infty$, $\mathbf{x}_K$ obtained from the procedure in (6) converges to a sample from $p(\mathbf{x})$ under some regularity conditions. practice, the error is negligible when $\epsilon$ is sufficiently small and $K$ is sufficiently large.

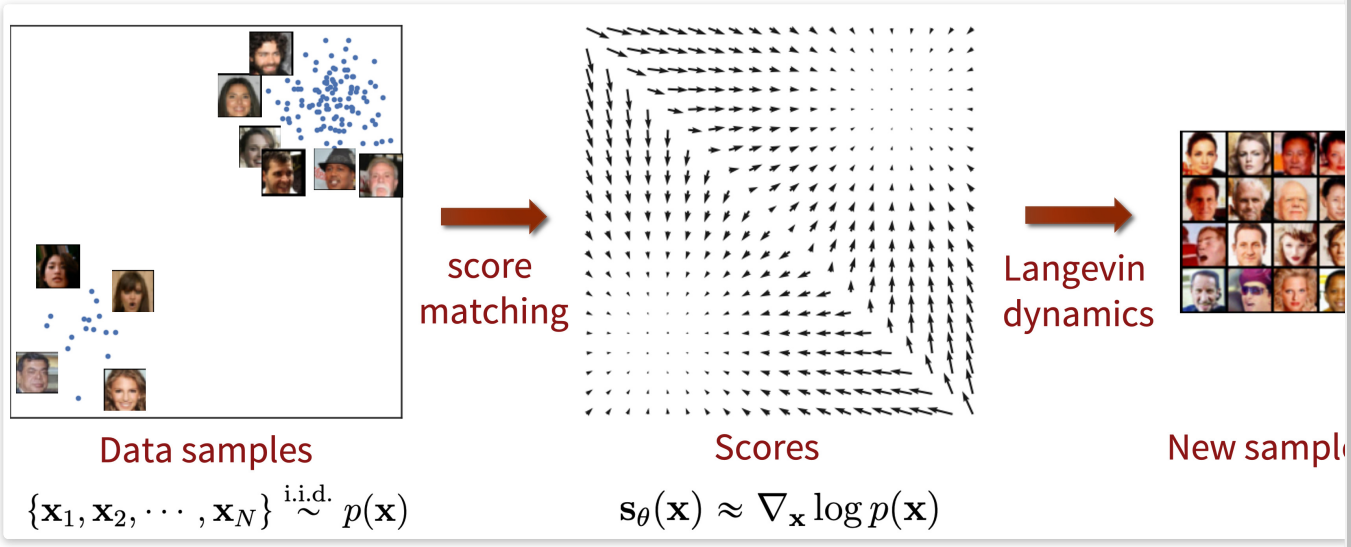


Using Langevin dynamics to sample from a mixture of two Gaussians.

Note that Langevin dynamics accesses $p(\mathbf{x})$ only through $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Since $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$, we can produce samples from our score-based model $\mathbf{s}_\theta(\mathbf{x})$ by plugging it into equation (6).

## Naive score-based generative modeling and its pitfalls

So far, we've discussed how to train a score-based model with score matching, and then produce samples via Langevin dynamics. However, this naive approach has had limited success in practice—we'll talk about some pitfalls of score matching that received little attention in prior works [17].

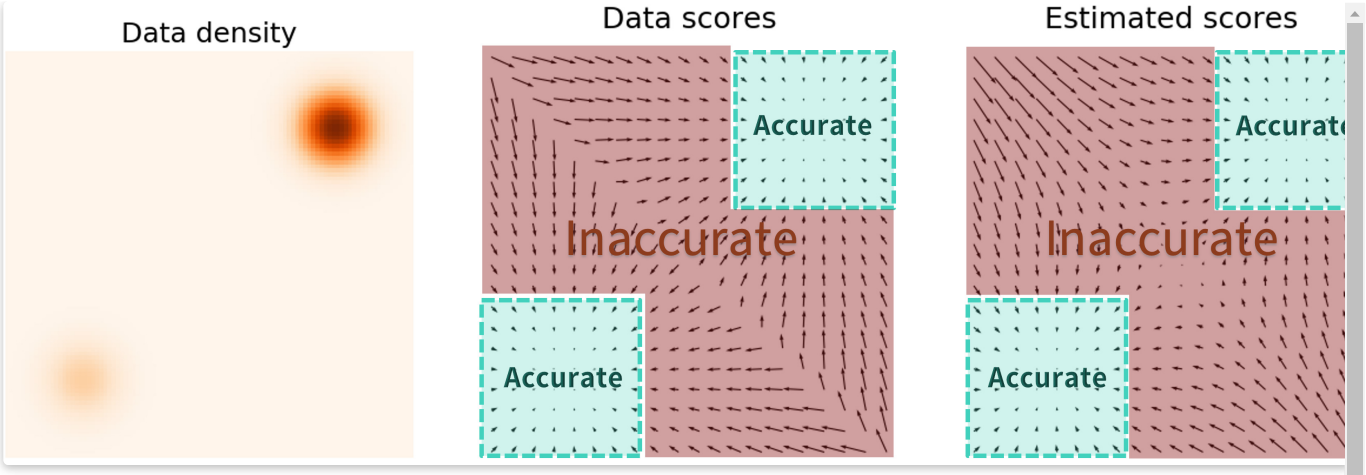



Score-based generative modeling with score matching + Langevin dynamics.

The key challenge is the fact that the estimated score functions are inaccurate in low density regions, where few data points are available for computing the score matching objective. This is expected as score matching minimizes the Fisher divergence

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] = \int p(\mathbf{x}) \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2 \mathrm{d}\mathbf{x}.$$

Since the $\ell_2$ differences between the true data score function and score-based model are weighted by $p(\mathbf{x})$, they are largely ignored in low density regions where $p(\mathbf{x})$ is small. This behavior can lead to subpar results, as illustrated by the figure below:

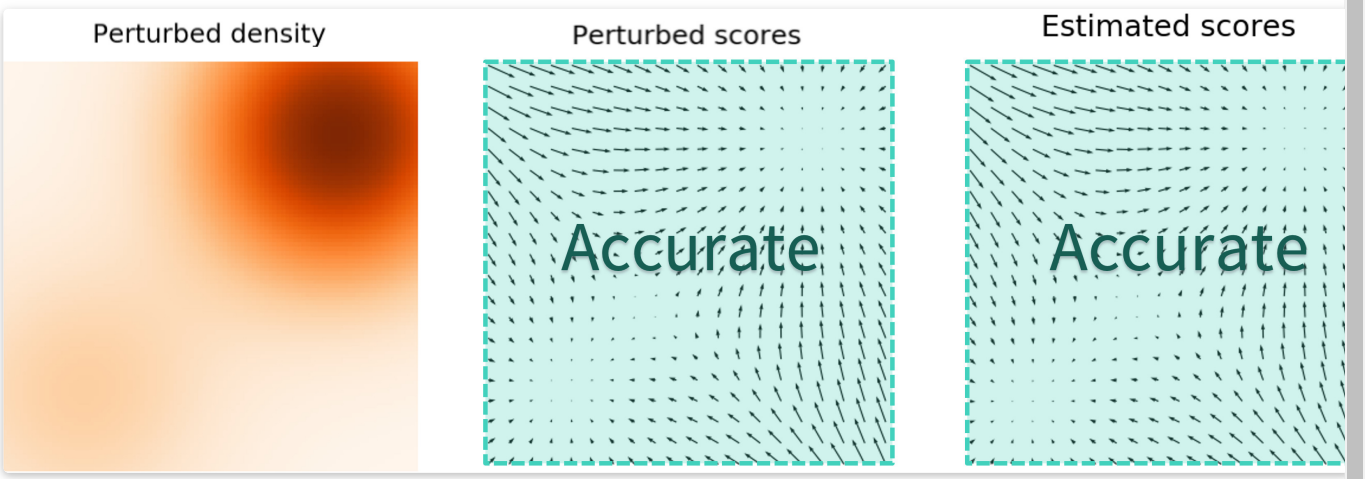



Estimated scores are only accurate in high density regions.

When sampling with Langevin dynamics, our initial sample is highly likely in low density regions when data reside in a high dimensional space. Therefore, having inaccurate score-based model will derail Langevin dynamics from the very beginning of the procedure, preventing it from generating high quality samples that a representative of the data.

## Score-based generative modeling with multiple noise perturbations

How can we bypass the difficulty of accurate score estimation in regions of low data density? Our solution is to **perturb** data points with noise and train score-based models on the noisy data points instead. When the noise magnitude is sufficiently large, it can populate low data density regions to improve the accuracy estimated scores. For example, here is what happens when we perturb a mixture of two Gaussians perturbed by additional Gaussian noise.




Estimated scores are accurate everywhere for the noise-perturbed data distribution due to reduced low data density regions.

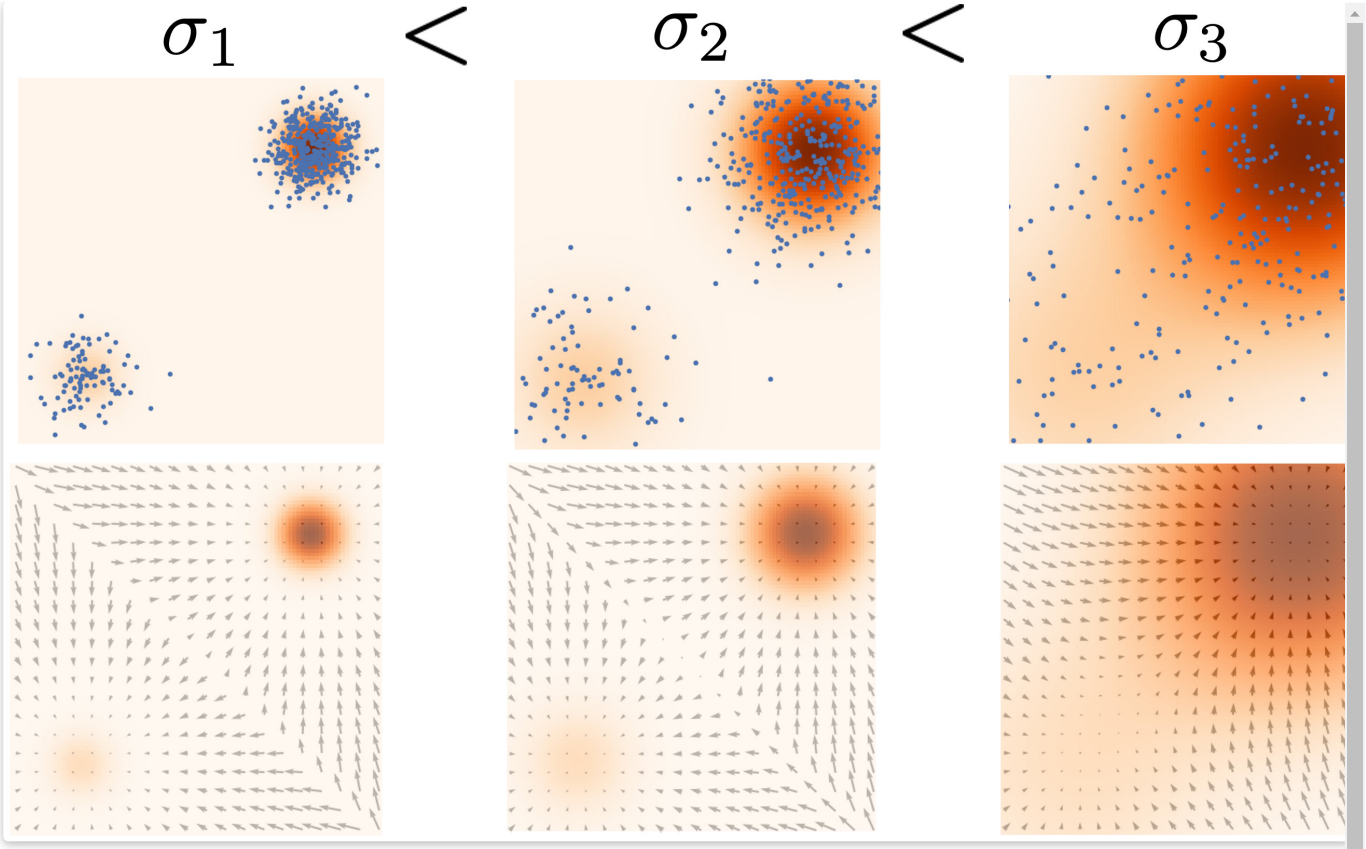
Yet another question remains: how do we choose an appropriate noise scale for the perturbation process? Larger noise can obviously cover more low density regions for better score estimation, but it over-corrupts the data and alters it significantly from the original distribution. Smaller noise, on the other hand, causes less corruption of the original data distribution, but does not cover the low density regions as well as we would like.

To achieve the best of both worlds, we use multiple scales of noise perturbations simultaneously [17, 18]. Suppose we always perturb the data with isotropic Gaussian noise, and let there be a total of $L$ increasing standard deviations $\sigma_1 < \sigma_2 < \cdots < \sigma_L$. We first perturb the data distribution $p(\mathbf{x})$ with each of the Gaussian noise $\mathcal{N}(0, \sigma_i^2 I), i = 1, 2, \cdots, L$ to obtain a noise-perturbed distribution

$$p_{\sigma_i}(\mathbf{x}) = \int p(\mathbf{y}) \mathcal{N}(\mathbf{x}; \mathbf{y}, \sigma_i^2 I) \mathrm{d}\mathbf{y}.$$

Note that we can easily draw samples from $p_{\sigma_i}(\mathbf{x})$ by sampling $\mathbf{x} \sim p(\mathbf{x})$ and computing $\mathbf{x} + \sigma_i \mathbf{z}$, with $\mathbf{z} \sim \mathcal{N}(0, I)$.

Next, we estimate the score function of each noise-perturbed distribution, $\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$, by training a **Noise Conditional Score-Based Model** $\mathbf{s}_\theta(\mathbf{x}, i)$ (also ca a Noise Conditional Score Network, or NCSN [17, 18, 20], when parameterized with a neural network) with score matching, such that $\mathbf{s}_\theta(\mathbf{x}, i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$ fo all $i = 1, 2, \cdots, L$.




We apply multiple scales of Gaussian noise to perturb the data distribution (**first row**), and jointly estimate the score functions for all of them (**second row**).

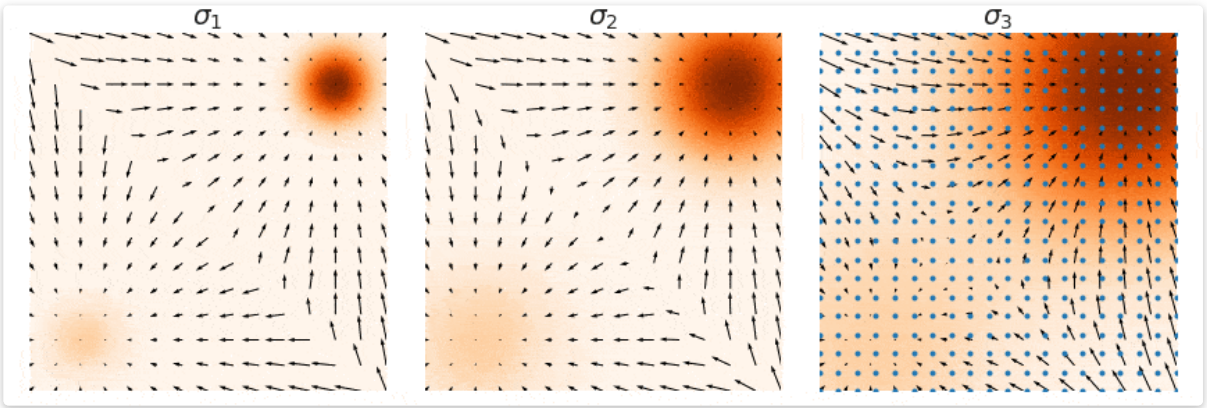
Perturbing an image with multiple scales of Gaussian noise.

The training objective for $\mathbf{s}_\theta(\mathbf{x}, i)$ is a weighted sum of Fisher divergences for all noise scales. In particular, we use the objective below:

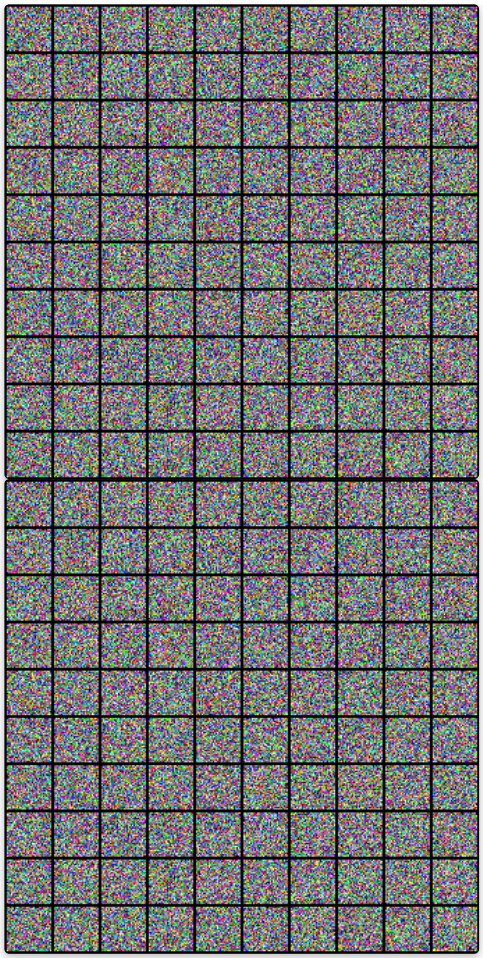
$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, i)\|_2^2],$$

where $\lambda(i) \in \mathbb{R}_{>0}$ is a positive weighting function, often chosen to be $\lambda(i) = \sigma_i^2$. The objective (7) can be optimized with score matching, exactly as in optimizing the naive (unconditional) score-based model $\mathbf{s}_\theta(\mathbf{x})$.

After training our noise-conditional score-based model $\mathbf{s}_\theta(\mathbf{x}, i)$, we can produce samples from it by running Langevin dynamics for $i = L, L-1, \cdots, 1$ in sequence. This method is called **annealed Langevin dynamics** (defined by Algorithm 1 in [17], and improved by [18, 33]), since the noise scale $\sigma_i$ decreases (anneals) gradually over time.




Annealed Langevin dynamics combine a sequence of Langevin chains with gradually decreasing noise scales.

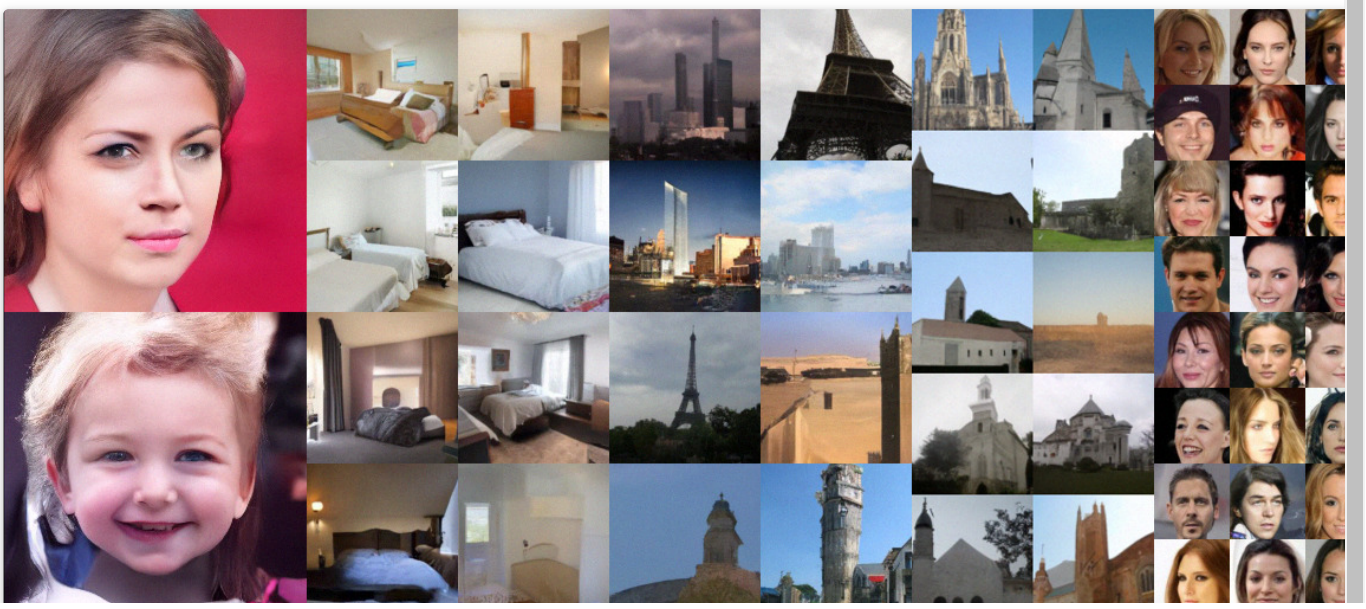


Annealed Langevin dynamics for the Noise Conditional Score Network (NCSN) model (from ref. [17]) trained on CelebA (**left**) and CIFAR-10 (**right**). We can start from unstructured nois modify images according to the scores, and generate nice samples. The method achieved state-of-the-art Inception score on CIFAR-10 at its time.

Here are some practical recommendations for tuning score-based generative models with multiple noise scales:

- Choose $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ as a geometric progression, with $\sigma_1$ being sufficiently small and $\sigma_L$ comparable to the maximum pairwise distance betwee training data points [18]. $L$ is typically on the order of hundreds or thousands.
- Parameterize the score-based model $\mathbf{s}_\theta(\mathbf{x}, i)$ with U-Net skip connections [17, 19].
- Apply exponential moving average on the weights of the score-based model when used at test time [18, 19].

With such best practices, we are able to generate high quality image samples with comparable quality to GANs on various datasets, such as below:



Samples from the NCSNv2 [18] model. From left to right: FFHQ 256x256, LSUN bedroom 128x128, LSUN tower 128x128, LSUN church_outdoor 96x96, and CelebA 64x64.

## Score-based generative modeling with stochastic differential equations (SDEs)

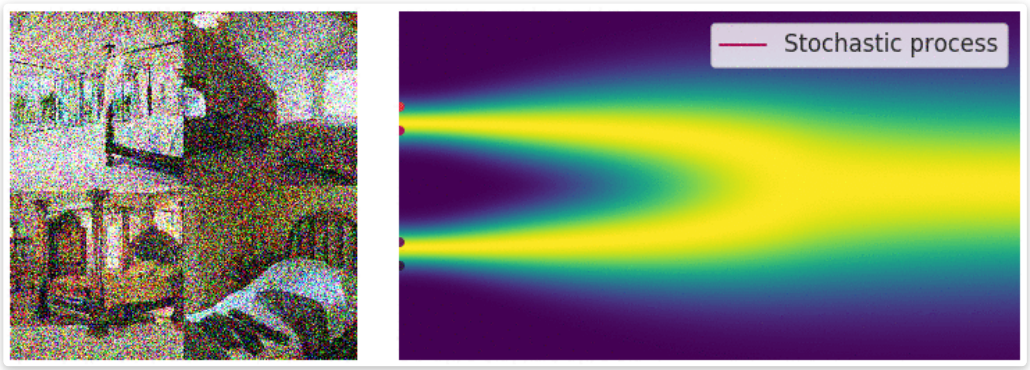
As we already discussed, adding multiple noise scales is critical to the success of score-based generative models. By generalizing the number of noise scales t infinity [20], we obtain not only **higher quality samples**, but also, among others, **exact log-likelihood computation**, and **controllable generation for inverse probl solving**.

In addition to this introduction, we have tutorials written in Google Colab to provide a step-by-step guide for training a toy model on MNIST. We also have more advanced code repositories that provide full-fledged implementations for large scale applications.

| Link | Description |
|---|---|
| Open in Colab | Tutorial of score-based generative modeling with SDEs in JAX + FLAX |
| Open in Colab | Load our pretrained checkpoints and play with sampling, likelihood computation, and controllable synthesis (JAX + FLAX) |
| Open in Colab | Tutorial of score-based generative modeling with SDEs in PyTorch |
| Open in Colab | Load our pretrained checkpoints and play with sampling, likelihood computation, and controllable synthesis (PyTorch) |
| Code in JAX | Score SDE codebase in JAX + FLAX |
| Code in PyTorch | Score SDE codebase in PyTorch |

### Perturbing data with an SDE

When the number of noise scales approaches infinity, we essentially perturb the data distribution with continuously growing levels of noise. In this case, the noi perturbation procedure is a continuous-time stochastic process, as demonstrated below




Perturbing data to noise with a continuous-time stochastic process.

How can we represent a stochastic process in a concise way? Many stochastic processes (diffusion processes in particular) are solutions of stochastic differe equations (SDEs). In general, an SDE possesses the following form:

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

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the drift coefficient, $g(t) \in \mathbb{R}$ is a real-valued function called the diffusion coefficient, $\mathbf{w}$ denotes a standard Brownian motion, and $\mathrm{d}\mathbf{w}$ can be viewed as infinitesimal white noise. The solution of a stochastic differential equation is a continuous collection of random variables $\{\mathbf{x}(t)\}_{t\in[0,T]}$. These random variables trace stochastic trajectories as the time index $t$ grows from the start time $0$ to the end time $T$. Let $p_t($ denote the (marginal) probability density function of $\mathbf{x}(t)$. Here $t \in [0, T]$ is analogous to $i = 1, 2, \cdots, L$ when we had a finite number of noise scales, and $p_t(\mathbf{x})$ is analogous to $p_{\sigma_i}(\mathbf{x})$. Clearly, $p_0(\mathbf{x}) = p(\mathbf{x})$ is the data distribution since no perturbation is applied to data at $t = 0$. After perturbing $p(\mathbf{x})$ with the stochastic process for a sufficiently long time $T$, $p_T(\mathbf{x})$ becomes close to a tractable noise distribution $\pi(\mathbf{x})$, called a **prior distribution**. We note that $p_T(\mathbf{x})$ is analogous to $p_{\sigma_L}(\mathbf{x})$ in the case of finite noise scales, which corresponds to applying the largest noise perturbation $\sigma_L$ to the data.

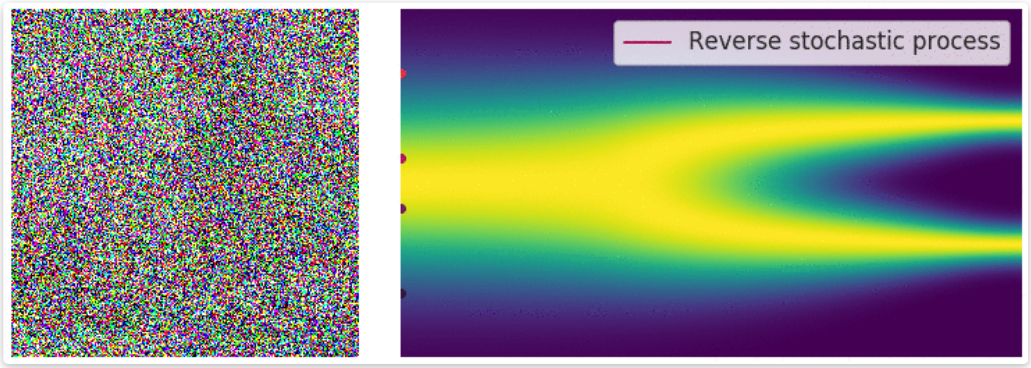
The SDE in (8) is **hand designed**, similarly to how we hand-designed $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ in the case of finite noise scales. There are numerous ways to add noise perturbations, and the choice of SDEs is not unique. For example, the following SDE

$$\mathrm{d}\mathbf{x} = e^t \mathrm{d}\mathbf{w}$$

perturbs data with a Gaussian noise of mean zero and exponentially growing variance, which is analogous to perturbing data with $\mathcal{N}(0, \sigma_1^2 I), \mathcal{N}(0, \sigma_2^2 I), \cdots, \mathcal{N}(0, \sigma_L^2 I)$ when $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ is a geometric progression. Therefore, the SDE should be viewed as part of the model, much like $\{\sigma_1, \sigma_2, \cdots, \sigma_L\}$. In [20], we provide three SDEs that generally work well for images: the Variance Exploding SDE (VE SDE), the Variance Preserving S (VP SDE), and the sub-VP SDE.

### Reversing the SDE for sample generation

Recall that with a finite number of noise scales, we can generate samples by reversing the perturbation process with **annealed Langevin dynamics**, i.e., sequentially sampling from each noise-perturbed distribution using Langevin dynamics. For infinite noise scales, we can analogously reverse the perturbation process for sample generation by using the reverse SDE.

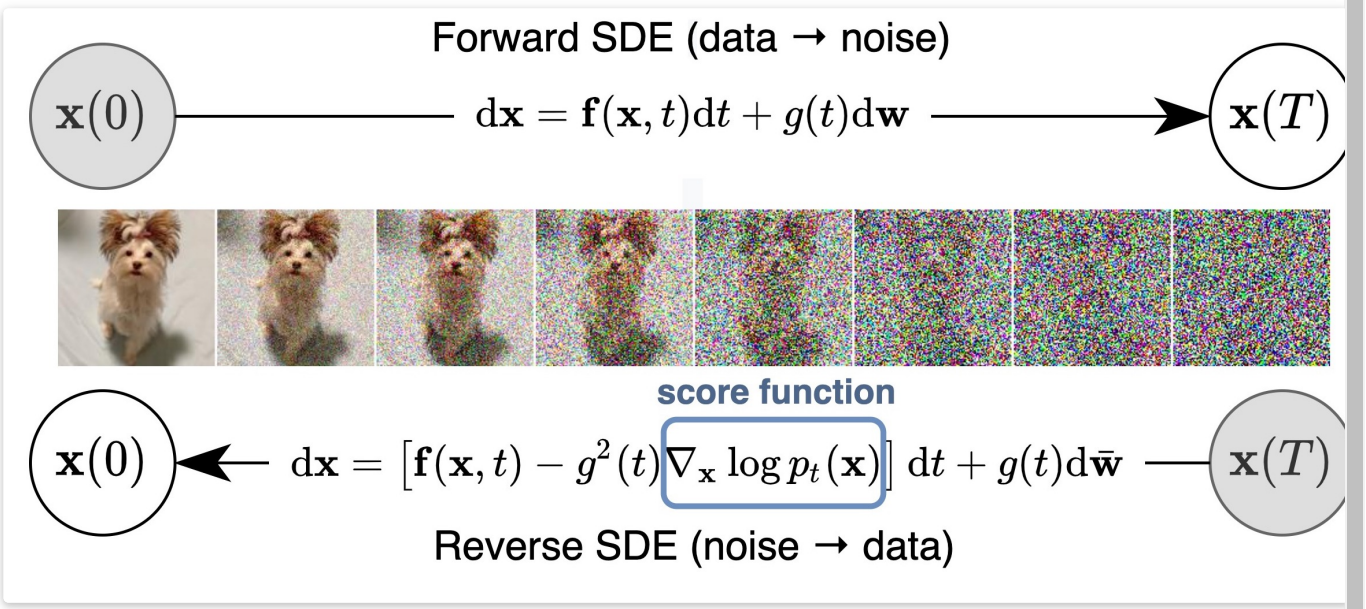



Generate data from noise by reversing the perturbation procedure.

Importantly, any SDE has a corresponding reverse SDE [34], whose closed form is given by

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

Here $\mathrm{d}t$ represents a negative infinitesimal time step, since the SDE (10) needs to be solved backwards in time (from $t = T$ to $t = 0$). In order to compute the reverse SDE, we need to estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, which is exactly the **score function** of $p_t(\mathbf{x})$.




Solving a reverse SDE yields a score-based generative model. Transforming data to a simple noise distribution can be accomplished with an SDE. It can be reversed to generate samples noise if we know the score of the distribution at each intermediate time step.

### Estimating the reverse SDE with score-based models and score matching

Solving the reverse SDE requires us to know the terminal distribution $p_T(\mathbf{x})$, and the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. By design, the former is close to the prior distribution $\pi(\mathbf{x})$ which is fully tractable. In order to estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we train a **Time-Dependent Score-Based Model** $\mathbf{s}_\theta(\mathbf{x}, t)$, such that $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. This is analogous to the noise-conditional score-based model $\mathbf{s}_\theta(\mathbf{x}, i)$ used for finite noise scales, trained such that $\mathbf{s}_\theta(\mathbf{x}, i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$.

Our training objective for $\mathbf{s}_\theta(\mathbf{x}, t)$ is a continuous weighted combination of Fisher divergences, given by

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2],$$

where $\mathcal{U}(0, T)$ denotes a uniform distribution over the time interval $[0, T]$, and $\lambda : \mathbb{R} \to \mathbb{R}_{>0}$ is a positive weighting function. Typically we use

As before, our weighted combination of Fisher divergences can be efficiently optimized with score matching methods, such as denoising score matching [16] a sliced score matching [30]. Once our score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$ is trained to optimality, we can plug it into the expression of the reverse SDE in (10) to obtain estimated reverse SDE.

$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}. \quad (1$$

We can start with $\mathbf{x}(T) \sim \pi$, and solve the above reverse SDE to obtain a sample $\mathbf{x}(0)$. Let us denote the distribution of $\mathbf{x}(0)$ obtained in such way as $p_\theta$. Wh the score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$ is well-trained, we have $p_\theta \approx p_0$, in which case $\mathbf{x}(0)$ is an approximate sample from the data distribution $p_0$.

When $\lambda(t) = g^2(t)$, we have an important connection between our weighted combination of Fisher divergences and the KL divergence from $p_0$ to $p_\theta$ under sc regularity conditions [35]:

$$\mathrm{KL}(p_0(\mathbf{x}) \| p_\theta(\mathbf{x})) \leq \frac{T}{2}\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(\mathbf{x})}[\lambda(t)\|\nabla_{\mathbf{x}}\log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\|_2^2]$$
$$+ \mathrm{KL}(p_T \parallel \pi). \quad (2$$

Due to this special connection to the KL divergence and the equivalence between minimizing KL divergences and maximizing likelihood for model training, we c $\lambda(t) = g(t)^2$ the **likelihood weighting function**. Using this likelihood weighting function, we can train score-based generative models to achieve very high likelihoods, comparable or even superior to state-of-the-art autoregressive models [35].

## How to solve the reverse SDE

By solving the estimated reverse SDE with numerical SDE solvers, we can simulate the reverse stochastic process for sample generation. Perhaps the simplest numerical SDE solver is the Euler-Maruyama method. When applied to our estimated reverse SDE, it discretizes the SDE using finite time steps and small Gauss noise. Specifically, it chooses a small negative time step $\Delta t \approx 0$, initializes $t \leftarrow T$, and iterates the following procedure until $t \approx 0$:

$$\begin{aligned}\Delta\mathbf{x} &\leftarrow [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t\\ \mathbf{x} &\leftarrow \mathbf{x} + \Delta\mathbf{x}\\ t &\leftarrow t + \Delta t,\end{aligned}$$

Here $\mathbf{z}_t \sim \mathcal{N}(0, I)$. The Euler-Maruyama method is qualitatively similar to Langevin dynamics—both update $\mathbf{x}$ by following score functions perturbed with Gaussian noise.

Aside from the Euler-Maruyama method, other numerical SDE solvers can be directly employed to solve the reverse SDE for sample generation, including, for example, Milstein method, and stochastic Runge-Kutta methods. In [20], we provided a reverse diffusion solver similar to Euler-Maruyama, but more tailored for solving reverse-time SDEs. More recently, authors in [36] introduced adaptive step-size SDE solvers that can generate samples faster with better quality.

In addition, there are two special properties of our reverse SDE that allow for even more flexible sampling methods:

- We have an estimate of $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ via our time-dependent score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$.
- We only care about sampling from each marginal distribution $p_t(\mathbf{x})$. Samples obtained at different time steps can have arbitrary correlations and do not ha to form a particular trajectory sampled from the reverse SDE.

As a consequence of these two properties, we can apply MCMC approaches to fine-tune the trajectories obtained from numerical SDE solvers. Specifically, we propose **Predictor-Corrector samplers**. The **predictor** can be any numerical SDE solver that predicts $\mathbf{x}(t + \Delta t) \sim p_{t+\Delta t}(\mathbf{x})$ from an existing sample $\mathbf{x}(t) \sim p_t(\mathbf{x})$. The **corrector** can be any MCMC procedure that solely relies on the score function, such as Langevin dynamics and Hamiltonian Monte Carlo.

At each step of the Predictor-Corrector sampler, we first use the predictor to choose a proper step size $\Delta t < 0$, and then predict $\mathbf{x}(t + \Delta t)$ based on the curre sample $\mathbf{x}(t)$. Next, we run several corrector steps to improve the sample $\mathbf{x}(t + \Delta t)$ according to our score-based model $\mathbf{s}_\theta(\mathbf{x}, t + \Delta t)$, so that $\mathbf{x}(t + \Delta t)$ becomes a higher-quality sample from $p_{t+\Delta t}(\mathbf{x})$.

With Predictor-Corrector methods and better architectures of score-based models, we can achieve **state-of-the-art** sample quality on CIFAR-10 (measured in FI [37] and Inception scores [12]), outperforming the best GAN model to date (StyleGAN2 + ADA [38]).

| Method | FID ↓ | Inception score ↑ |
|---|---|---|
| StyleGAN2 + ADA [38] | 2.92 | 9.83 |
| Ours [20] | **2.20** | **9.89** |

The sampling methods are also scalable for extremely high dimensional data. For example, it can successfully generate high fidelity images of resolution $1024 \times 1024$.

1024 x 1024 samples from a score-based model trained on the FFHQ dataset.

Some additional (uncurated) samples for other datasets (taken from this GitHub repo):

256 x 256 samples on LSUN bedroom.

256 x 256 samples on CelebA-HQ.

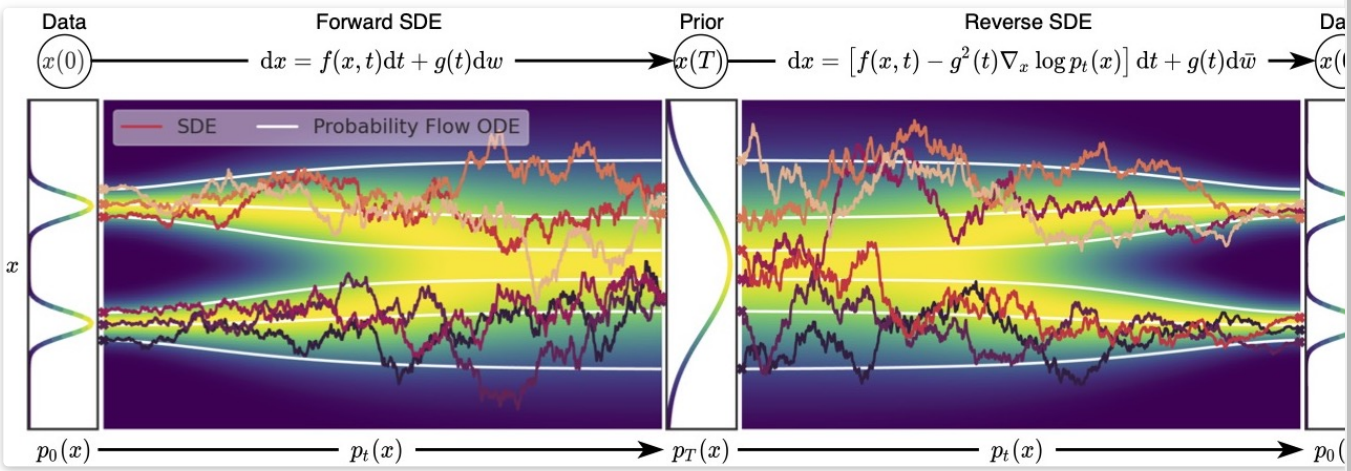
## Probability flow ODE

Despite capable of generating high-quality samples, samplers based on Langevin MCMC and SDE solvers do not provide a way to compute the exact log-likelih of score-based generative models. Below, we introduce a sampler based on ordinary differential equations (ODEs) that allow for exact likelihood computation.

In [20], we show t is possible to convert any SDE into an ordinary differential equation (ODE) without changing its marginal distributions $\{p_t(\mathbf{x})\}_{t\in[0,T]}$. Thus by solving this ODE, we can sample from the same distributions as the reverse SDE. The corresponding ODE of an SDE is named **probability flow ODE** [20], given b

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

The following figure depicts trajectories of both SDEs and probability flow ODEs. Although ODE trajectories are noticeably smoother than SDE trajectories, they convert the same data distribution to the same prior distribution and vice versa, sharing the same set of marginal distributions $\{p_t(\mathbf{x})\}_{t\in[0,T]}$. In other words, trajectories obtained by solving the probability flow ODE have the same marginal distributions as the SDE trajectories.




We can map data to a noise distribution (the prior) with an SDE, and reverse this SDE for generative modeling. We can also reverse the associated probability flow ODE, 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 score functions.

This probability flow ODE formulation has several unique advantages.

When $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is replaced by its approximation $\mathbf{s}_\theta(\mathbf{x}, t)$, the probability flow ODE becomes a special case of a neural ODE [39]. In particular, it is an examp of continuous normalizing flows [40], since the probability flow ODE converts a data distribution $p_0(\mathbf{x})$ to a prior noise distribution $p_T(\mathbf{x})$ (since it shares the sa marginal distributions as the SDE) and is fully invertible.

As such, the probability flow ODE inherits all properties of neural ODEs or continuous normalizing flows, including exact log-likelihood computation. Specifically, can leverage the instantaneous change-of-variable formula (Theorem 1 in [39], Equation (4) in [40]) to compute the unknown data density $p_0$ from the known pri density $p_T$ with numerical ODE solvers.

In fact, our model achieves the **state-of-the-art** log-likelihoods on uniformly dequantized [4] CIFAR-10 images [20], **even without maximum likelihood training**.

| Method | Negative log-likelihood (bits/dim) ↓ |
|---|---|
| RealNVP | 3.49 |
| iResNet | 3.45 |
| Glow | 3.35 |
| FFJORD | 3.40 |
| Flow++ | 3.29 |
| Ours | **2.99** |

When training score-based models with the **likelihood weighting** we discussed before, and using **variational dequantization** to obtain likelihoods on discrete images, we can achieve comparable or even superior likelihood to the state-of-the-art autoregressive models (all without any data augmentation) [35].

| Method | Negative log-likelihood (bits/dim) ↓ on CIFAR-10 | Negative log-likelihood (bits/dim) ↓ on ImageNet 32x32 |
|---|---|---|
| Sparse Transformer | **2.80** | - |
| Image Transformer | 2.90 | 3.77 |
| Ours | 2.83 | **3.76** |

## Controllable generation for inverse problem solving

Score-based generative models are particularly suitable for solving inverse problems. At its core, inverse problems are same as Bayesian inference problems. L and $\mathbf{y}$ be two random variables, and suppose we know the forward process of generating $\mathbf{y}$ from $\mathbf{x}$, represented by the transition probability distribution $p(\mathbf{y} \mid$ The inverse problem is to compute $p(\mathbf{x} \mid \mathbf{y})$. From Bayes' rule, we have $p(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x}) / \int p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})\mathrm{d}\mathbf{x}$. This expression can be greatly simpli by taking gradients with respect to $\mathbf{x}$ on both sides, leading to the following Bayes' rule for score functions:

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

Through score matching, we can train a model to estimate the score function of the unconditional data distribution, i.e., $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$. This will allow u to easily compute the posterior score function $\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y})$ from the known forward process $p(\mathbf{y} \mid \mathbf{x})$ via equation (15), and sample from it with Langevi type sampling [20].

A recent work from UT Austin [28] has demonstrated that score-based generative models can be applied to solving inverse problems in medical imaging, such a accelerating magnetic resonance imaging (MRI). Concurrently in [41], we demonstrated superior performance of score-based generative models not only on accelerated MRI, but also sparse-view computed tomography (CT). We were able to achieve comparable or even better performance than supervised or unrolle deep learning approaches, while being more robust to different measurement processes at test time.

Below we show some examples on solving inverse problems for computer vision.


Class-conditional generation with an unconditional time-dependent score-based model, and a pre-trained noise-conditional image classifier on CIFAR-10.

Image inpainting with a time-dependent score-based model trained on LSUN bedroom. The leftmost column is ground-truth. The second column shows masked images (y in our framew
The rest columns show different inpainted images, generated by solving the conditional reverse-time SDE.

Image colorization with a time-dependent score-based model trained on LSUN church_outdoor and bedroom. The leftmost column is ground-truth. The second column shows gray-sca
images (y in our framework). The rest columns show different colorizedimages, generated by solving the conditional reverse-time SDE.

We can even colorize gray-scale portrays of famous people in history (Abraham Lincoln) with a time-dependent score-based model trained on FFHQ. The image resolution is 1024 x 10

## Connection to diffusion models and others

I started working on score-based generative modeling since 2019, when I was trying hard to make score matching scalable for training deep energy-based mod on high-dimensional datasets. My first attempt at this led to the method sliced score matching [30]. Despite the scalability of sliced score matching for training energy-based models, I found to my surprise that Langevin sampling from those models fails to produce reasonable samples even on the MNIST dataset. I star investigating this issue and discovered three crucial improvements that can lead to extremely good samples: (1) perturbing data with multiple scales of noise, a training score-based models for each noise scale; (2) using a U-Net architecture (we used RefineNet since it is a modern version of U-Nets) for the score-based model; (3) applying Langevin MCMC to each noise scale and chaining them together. With those methods, I was able to obtain the state-of-the-art Inception Sc on CIFAR-10 in [17] (even better than the best GANs!), and generate high-fidelity image samples of resolution up to $256 \times 256$ in [18].

The idea of perturbing data with multiple scales of noise is by no means unique to score-based generative models though. It has been previously used in, for example, simulated annealing, annealed importance sampling [42], diffusion probabilistic models [43], infusion training [44], and variational walkback [45] for generative stochastic networks [46]. Out of all these works, diffusion probabilistic modeling is perhaps the closest to score-based generative modeling. Diffusion probabilistic models are hierachical latent variable models first proposed by Jascha and his colleagues [43] in 2015, which generate samples by learning a variational decoder to reverse a discrete diffusion process that perturbs data to noise. Without awareness of this work, score-based generative modeling was proposed and motivated independently from a very different perspective. Despite both perturbing data with multiple scales of noise, the connection between sc based generative modeling and diffusion probabilistic modeling seemed superficial at that time, since the former is trained by score matching and sampled by Langevin dynamics, while the latter is trained by the evidence lower bound (ELBO) and sampled with a learned decoder.

In 2020, Jonathan Ho and colleagues [19] significantly improved the empirical performance of diffusion probabilistic models and first unveiled a deeper connec to score-based generative modeling. They showed that the ELBO used for training diffusion probabilistic models is essentially equivalent to the weighted combination of score matching objectives used in score-based generative modeling. Moreover, by parameterizing the decoder as a sequence of score-based models with a U-Net architecture, they demonstrated for the first time that diffusion probabilistic models can also generate high quality image samples comparable or superior to GANs.

Inspired by their work, we further investigated the relationship between diffusion models and score-based generative models in an ICLR 2021 paper [20]. We fou that the sampling method of diffusion probabilistic models can be integrated with annealed Langevin dynamics of score-based models to create a unified and more powerful sampler (the Predictor-Corrector sampler). By generalizing the number of noise scales to infinity, we further proved that score-based generative models and diffusion probabilistic models can both be viewed as discretizations to stochastic differential equations determined by score functions. This work bridges both score-based generative modeling and diffusion probabilistic modeling into a unified framework.

Collectively, these latest developments seem to indicate that both score-based generative modeling with multiple noise perturbations and diffusion probabilistic models are different perspectives of the same model family, much like how wave mechanics and matrix mechanics are equivalent formulations of quantum mechanics in the history of physics [5]. The perspective of score matching and score-based models allows one to calculate log-likelihoods exactly, solve inverse problems naturally, and is directly connected to energy-based models, Schrödinger bridges and optimal transport [47]. The perspective of diffusion models is naturally connected to VAEs, lossy compression, and can be directly incorporated with variational probabilistic inference. This blog post focuses on the first perspective, but I highly recommend interested readers to learn about the alternative perspective of diffusion models as well (see a great blog by Lilian Weng).

Many recent works on score-based generative models or diffusion probabilistic models have been deeply influenced by knowledge from both sides of research (see a website curated by researchers at the University of Oxford). Despite this deep connection between score-based generative models and diffusion models, hard to come up with an umbrella term for the model family that they both belong to. Some colleagues in DeepMind propose to call them "Generative Diffusion Processes". It remains to be seen if this will be adopted by the community in the future.

## Concluding remarks

This blog post gives a detailed introduction to score-based generative models. We demonstrate that this new paradigm of generative modeling is able to produc high quality samples, compute exact log-likelihoods, and perform controllable generation for inverse problem solving. It is a compilation of several papers we published in the past few years. Please visit them if you are interested in more details:

- Yang Song*, Sahaj Garg*, Jiaxin Shi, and Stefano Ermon. *Sliced Score Matching: A Scalable Approach to Density and Score Estimation*. UAI 2019 (Oral)
- Yang Song, and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. NeurIPS 2019 (Oral)
- Yang Song, and Stefano Ermon. *Improved Techniques for Training Score-Based Generative Models*. NeurIPS 2020
- Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. *Score-Based Generative Modeling through Stochas Differential Equations*. ICLR 2021 (Outstanding Paper Award)
- Yang Song*, Conor Durkan*, Iain Murray, and Stefano Ermon. *Maximum Likelihood Training of Score-Based Diffusion Models*. NeurIPS 2021 (Spotlight)
- Yang Song*, Liyue Shen*, Lei Xing, and Stefano Ermon. *Solving Inverse Problems in Medical Imaging with Score-Based Generative Models*. ICLR 2022

For a list of works that have been influenced by score-based generative modeling, researchers at the University of Oxford have built a very useful (but necessari incomplete) website: https://scorebasedgenerativemodeling.github.io/.

There are two major challenges of score-based generative models. First, the sampling speed is slow since it involves a large number of Langevin-type iterations Second, it is inconvenient to work with discrete data distributions since scores are only defined on continuous distributions.

The first challenge can be partially solved by using numerical ODE solvers for the probability flow ODE with lower precision (a similar method, denoising diffusio implicit modeling, has been proposed in [48]). It is also possible to learn a direct mapping from the latent space of probability flow ODEs to the image space, as shown in [49]. However, all such methods to date result in worse sample quality.

The second challenge can be addressed by learning an autoencoder on discrete data and performing score-based generative modeling on its continuous latent space [27, 50]. Jascha's original work on diffusion models [43] also provides a discrete diffusion process for discrete data distributions, but its potential for large scale applications remains yet to be proven.

It is my conviction that these challenges will soon be solved with the joint efforts of the research community, and score-based generative models/ diffusion-bas models will become one of the most useful tools for data generation, density estimation, inverse problem solving, and many other downstream tasks in machin learning.

### Footnotes

1. Hereafter we only consider probability density functions. Probability mass functions are similar. [↩]
2. Fisher divergence is typically between two distributions p and q, defined as

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \nabla_{\mathbf{x}} \log q(\mathbf{x})\|_2^2]. \tag{4}$$

   Here we slightly abuse the term as the name of a closely related expression for score-based models. [↩]
3. Commonly used score matching methods include denoising score matching [16] and sliced score matching [30]. Here is an introduction to score matching and sliced score matching. [↩]
4. It is typical for normalizing flow models to convert discrete images to continuous ones by adding small uniform noise to them. [↩]
5. Goes without saying that the significance of score-based generative models/diffusion probabilistic models is in no way comparable to quantum mechanics. [↩]

### References

1. **The neural autoregressive distribution estimator**
   Larochelle, H. and Murray, I., 2011. International Conference on Artificial Intelligence and Statistics, pp. 29–37.
2. **Made: Masked autoencoder for distribution estimation**
   Germain, M., Gregor, K., Murray, I. and Larochelle, H., 2015. International Conference on Machine Learning, pp. 881–889.

3. **Pixel recurrent neural networks**
   Van Oord, A., Kalchbrenner, N. and Kavukcuoglu, K., 2016. International Conference on Machine Learning, pp. 1747--1756.

4. **NICE: Non-linear independent components estimation**
   Dinh, L., Krueger, D. and Bengio, Y., 2014. arXiv preprint arXiv:1410.8516.

5. **Density estimation using Real NVP**
   Dinh, L., Sohl-Dickstein, J. and Bengio, S., 2017. International Conference on Learning Representations.

6. **A tutorial on energy-based learning**
   LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M. and Huang, F., 2006. Predicting structured data, Vol 1(0).

7. **How to Train Your Energy-Based Models**
   Song, Y. and Kingma, D.P., 2021. arXiv preprint arXiv:2101.03288.

8. **Auto-encoding variational bayes**
   Kingma, D.P. and Welling, M., 2014. International Conference on Learning Representations.

9. **Stochastic backpropagation and approximate inference in deep generative models**
   Rezende, D.J., Mohamed, S. and Wierstra, D., 2014. International conference on machine learning, pp. 1278--1286.

10. **Learning in implicit generative models**
    Mohamed, S. and Lakshminarayanan, B., 2016. arXiv preprint arXiv:1610.03483.

11. **Generative adversarial nets**
    Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Advances in neural information processing systems, pp. 2672--2680.

12. **Improved techniques for training gans**
    Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. and Chen, X., 2016. Advances in Neural Information Processing Systems, pp. 2226--2234.

13. **Unrolled Generative Adversarial Networks** [link]
    Metz, L., Poole, B., Pfau, D. and Sohl-Dickstein, J., 2017. 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

14. **A kernelized Stein discrepancy for goodness-of-fit tests**
    Liu, Q., Lee, J. and Jordan, M., 2016. International conference on machine learning, pp. 276--284.

15. **Estimation of non-normalized statistical models by score matching**
    Hyvarinen, A., 2005. Journal of Machine Learning Research, Vol 6(Apr), pp. 695--709.

16. **A connection between score matching and denoising autoencoders**
    Vincent, P., 2011. Neural computation, Vol 23(7), pp. 1661--1674. MIT Press.

17. **Generative Modeling by Estimating Gradients of the Data Distribution** [PDF]
    Song, Y. and Ermon, S., 2019. Advances in Neural Information Processing Systems, pp. 11895--11907.

18. **Improved Techniques for Training Score-Based Generative Models** [PDF]
    Song, Y. and Ermon, S., 2020. Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

19. **Denoising diffusion probabilistic models**
    Ho, J., Jain, A. and Abbeel, P., 2020. arXiv preprint arXiv:2006.11239.

20. **Score-Based Generative Modeling through Stochastic Differential Equations** [link]
    Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S. and Poole, B., 2021. International Conference on Learning Representations.

21. **Diffusion models beat gans on image synthesis**
    Dhariwal, P. and Nichol, A., 2021. arXiv preprint arXiv:2105.05233.

22. **Cascaded Diffusion Models for High Fidelity Image Generation**
    Ho, J., Saharia, C., Chan, W., Fleet, D.J., Norouzi, M. and Salimans, T., 2021.

23. **WaveGrad: Estimating Gradients for Waveform Generation** [link]
    Chen, N., Zhang, Y., Zen, H., Weiss, R.J., Norouzi, M. and Chan, W., 2021. International Conference on Learning Representations.

24. **DiffWave: A Versatile Diffusion Model for Audio Synthesis** [link]
    Kong, Z., Ping, W., Huang, J., Zhao, K. and Catanzaro, B., 2021. International Conference on Learning Representations.

25. **Grad-tts: A diffusion probabilistic model for text-to-speech**
    Popov, V., Vovk, I., Gogoryan, V., Sadekova, T. and Kudinov, M., 2021. arXiv preprint arXiv:2105.06337.

26. **Learning Gradient Fields for Shape Generation**
    Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N. and Hariharan, B., 2020. Proceedings of the European Conference on Computer Vision (ECCV).

27. **Symbolic Music Generation with Diffusion Models**
    Mittal, G., Engel, J., Hawthorne, C. and Simon, I., 2021. arXiv preprint arXiv:2103.16091.

28. **Robust Compressed Sensing MRI with Deep Generative Priors**
    Jalal, A., Arvinte, M., Daras, G., Price, E., Dimakis, A.G. and Tamir, J.I., 2021. Advances in neural information processing systems.

29. **Training products of experts by minimizing contrastive divergence**
    Hinton, G.E., 2002. Neural computation, Vol 14(8), pp. 1771--1800. MIT Press.

30. **Sliced score matching: A scalable approach to density and score estimation** [PDF]

31. **Correlation functions and computer simulations**
Parisi, G., 1981. Nuclear Physics B, Vol 180(3), pp. 378--384. Elsevier.

32. **Representations of knowledge in complex systems**
Grenander, U. and Miller, M.I., 1994. Journal of the Royal Statistical Society: Series B (Methodological), Vol 56(4), pp. 549--581. Wiley Online Library.

33. **Adversarial score matching and improved sampling for image generation** [link]
Jolicoeur-Martineau, A., Piche-Taillefer, R., Mitliagkas, I. and Combes, R.T.d., 2021. International Conference on Learning Representations.

34. **Reverse-time diffusion equation models**
Anderson, B.D., 1982. Stochastic Processes and their Applications, Vol 12(3), pp. 313--326. Elsevier.

35. **Maximum Likelihood Training of Score-Based Diffusion Models**
Song, Y., Durkan, C., Murray, I. and Ermon, S., 2021. Advances in Neural Information Processing Systems (NeurIPS).

36. **Gotta Go Fast When Generating Data with Score-Based Models**
Jolicoeur-Martineau, A., Li, K., Piche-Taillefer, R., Kachman, T. and Mitliagkas, I., 2021. arXiv preprint arXiv:2105.14080.

37. **GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium**
Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S., 2017. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, {USA}, pp. 6626--6637.

38. **Training Generative Adversarial Networks with Limited Data**
Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. and Aila, T., 2020. Proc. NeurIPS.

39. **Neural Ordinary Differential Equations**
Chen, T.Q., Rubanova, Y., Bettencourt, J. and Duvenaud, D., 2018. Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montr{\'{e}}al, Canada, pp. 6572--6583.

40. **Scalable Reversible Generative Models with Free-form Continuous Dynamics** [link]
Grathwohl, W., Chen, R.T.Q., Bettencourt, J. and Duvenaud, D., 2019. International Conference on Learning Representations.

41. **Solving Inverse Problems in Medical Imaging with Score-Based Generative Models** [PDF]
Song, Y., Shen, L., Xing, L. and Ermon, S., 2022. International Conference on Learning Representations.

42. **Annealed importance sampling**
Neal, R.M., 2001. Statistics and computing, Vol 11(2), pp. 125--139. Springer.

43. **Deep unsupervised learning using nonequilibrium thermodynamics**
Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. and Ganguli, S., 2015. International Conference on Machine Learning, pp. 2256--2265.

44. **Learning to generate samples from noise through infusion training**
Bordes, F., Honari, S. and Vincent, P., 2017. arXiv preprint arXiv:1703.06975.

45. **Variational walkback: Learning a transition operator as a stochastic recurrent net**
Goyal, A., Ke, N.R., Ganguli, S. and Bengio, Y., 2017. arXiv preprint arXiv:1711.02282.

46. **GSNs: generative stochastic networks**
Alain, G., Bengio, Y., Yao, L., Yosinski, J., Thibodeau-Laufer, E., Zhang, S. and Vincent, P., 2016. Information and Inference: A Journal of the IMA, Vol 5(2), pp. 210--249. Oxford University Press.

47. **Diffusion Schrödinger Bridge with Applications to Score-Based Generative Modeling**
De Bortoli, V., Thornton, J., Heng, J. and Doucet, A., 2021. Advances in Neural Information Processing Systems (NeurIPS).

48. **Denoising Diffusion Implicit Models** [link]
Song, J., Meng, C. and Ermon, S., 2021. International Conference on Learning Representations.

49. **Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed**
Luhman, E. and Luhman, T., 2021. arXiv e-prints, pp. arXiv--2101.

50. **Score-based Generative Modeling in Latent Space**
Vahdat, A., Kreis, K. and Kautz, J., 2021. Advances in Neural Information Processing Systems (NeurIPS).

Muhammad Firmansyah Kasim
a year ago

Thank you for this great blog post! Why did you model the score **s** directly? Why not model the unnormalized log(p), and then take its derivative with respect to **x** to compute **s**? The second one produces the score profile that is automatically a differential 1-form (i.e. its Jacobian is symmetric), while modelling **s** directly does not necessarily produce a differential 1-form (its Jacobian is not necessarily symmetric).

3 1 Reply • Share ›

C Carl ↗ Muhammad Firmansyah Kasim
6 months ago

For very good reasons: 1) score can be directly used in Langevin sampling & reverse SDE in diffusion models 2) Taking derivative is extremely expensive for large neural networks. 3) Most realistic data distribution has compact support, the log of 0 is minus infinity, modeling log(p) can lead to numerical instability .

0 0 Reply • Share ›

J jiahao Lu
a year ago

Such an inspiring and revealing blog ! It really helps understanding. Many thanks !

1 0 Reply • Share ›

J jiahao Lu ↗ jiahao Lu
a year ago

Actually I have one question here .. Why score-based models have exact log-likelihood computation ? From my understanding, matching the gradients of log-likelihood is kind of approximating the exact log-likelihood computation ...

0 0 Reply • Share ›

Carl Thomé
a year ago

Amazing blog post! Thanks so much for this.

1 0 Reply • Share ›

J Jaechul Lee
3 months ago

It makes me much easier to figure out your great job. I look forward to your next publication. Thank you so much for the awesome post!

**I** **ivan**
6 months ago

Hi, thanks for the post. I am looking at your notebook example (Pytorch version). You rescale your ScoreNet output by multiplying 1/std[:, None, None, None], then in the loss_fn, you multiply std[:, None, None, None] back again, what's the point? It seems to do nothing with the output if you don't normalize and rescale your output.

0 0 Reply • Share ›

**S** **Sukjun Hwang** ↗ ivan
2 months ago edited

$\frac{1}/{\sigma}$ gets multiplied to the ScoreNet due to the characteristic that the output of optimally trained $s_\{theta}(x, \sigma)$ is proportional to $\frac{1}/{\sigma}$.
Say it is not multiplied at the end. Then, parameters of the ScoreNet has to control l2 norm of its output with respect to the given time step $t$.
Because $\frac{1}/{\sigma}$ gets multiplied at the end, the ScoreNet can be better trained as now it does not have to control l2 norm of its output, getting independent to $t$.

The latter multiplication of $\sigma$ in the loss function is driven from the denoising score matching, which you can refer to Section 4.2. of the NCSN paper.

1 0 Reply • Share ›

**Z** **Zweig Shao**
a year ago

Why does the pitfall of naive score-based generative (SBG) modeling (mentioned in the post) exist and can be solved by noise perturbing? I mean, when training SBG model, we use the denoising score matching technique, the noise perturbing is already involved. So does the SBG model trained by denoising score matching still have that pitfall? If yes, is there a conflict that we sitll use noise perturbing to eliminate this problem?
感谢您提供了可读性如此高的post，为了表达清晰，我重新用中文表达一遍我的困惑。文章中提到的naive score-based generative modeling是用什么方式训练的呢，如果是denoising score matching的方法，这里不是已经引入了噪声了嘛？为什么后面又讲到存在“低概率区域训练不充分”的问题，并使用噪声扰动加以解决。

0 0 Reply • Share ›

**M** **Miao Song** ↗ Zweig Shao
3 months ago

Use the sliced score matching to estimate scores without noise, which is written on P4 of the paper

0 0 Reply • Share ›

**Yuanzhi** ↗ Zweig Shao
a year ago edited

First of all, denoising score matching is introduced to circumvent trace of gradient of score function, rather than handle the pitfalls (they pop up later). Indeed, denoising score matching corresponding to only one specific noise level: if the noise is large, we can't get accurate score estimation; if the noise if small, we still have the low density regions pitfall (trade-off here). What we want at last, is to both perturbate the data sufficiently (large noise) to better estimate the low density regions and be able to converge to the true data distribution (small noise). In this case, a sequence of noise-levels is proposed for both accurate training and sampling (not one score network but a set of noise conditional score networks (though they can be represented by only one neural network)).
Please correct me if my understand is wrong :)

0 0 Reply • Share ›