Score-Based Generative Modeling through Stochastic Differential Equations
SGM through SDE

Yang Song, et al., ICLR 2021.
Stanford University | Google Brain

Suk-Hwan Lee

Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering &
Artificial Intelligence

References

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, Ben Poole, "**Score-Based Generative Modeling through Stochastic Differential Equations**," ICLR 2021.
Stanford University | Google Brain

https://github.com/yang-song/score_sde
https://yang-song.net/blog/2021/score/

[blog]
https://kimjy99.github.io/논문리뷰/sbgm/
https://blog.si-analytics.ai/49
https://junia3.github.io/blog/scoresde

Score-Based Generative Modeling through SDE

Abstract (Paper)

- 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**,
  - **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 leveraing the score-based generative modeling, we can accurately **estimate these scores with neural networks**, and **use numerical SDE solvers to generate samples.**

- This framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities.

- Particularly introduce **a predictor-corrector framework** to **correct errors in the evolution of the discretized reverse-time SDE.**

- 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.**

- Provide **a new way to solve inverse problems with score-based models**, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization.

- 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**,
- Demonstrate high fidelity generation of **1024x1024 images** for the first time from a score-based generative model.

Score-Based Generative Modeling through SDE

Overview

https://yang-song.net/blog/2021/score/

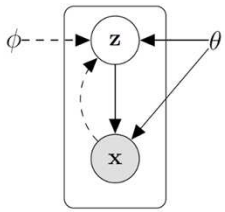
- We can **learn score functions** (**gradients of log probability density functions**) on **a large number of noise-perturbed data distributions**, then **generate samples with Langevin-type sampling**.
- The resulting generative models, often called **score-based generative models**, has several important advantages over existing model families:
  - **GAN-level sample quality without adversarial training**
  - **Flexible model architectures**
  - **Exact log-likelihood computation**
  - **Inverse problem solving without re-training models**

Introduction

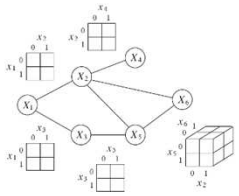
- **Existing generative modeling techniques** can largely be grouped into **two categories** based on **how they represent probability distributions.**

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].

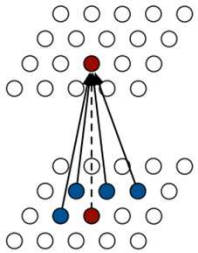
② **Implicit generative models** [10], where **the probability distribution is implicitly represented by a model of its sampling process**. The most prominent example is **generative adversarial networks (GANs)** [11], where new samples from the data distribution are synthesized by transforming a random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



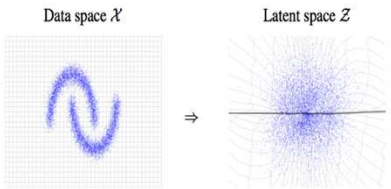
Bayesian networks (e.g., VAEs)



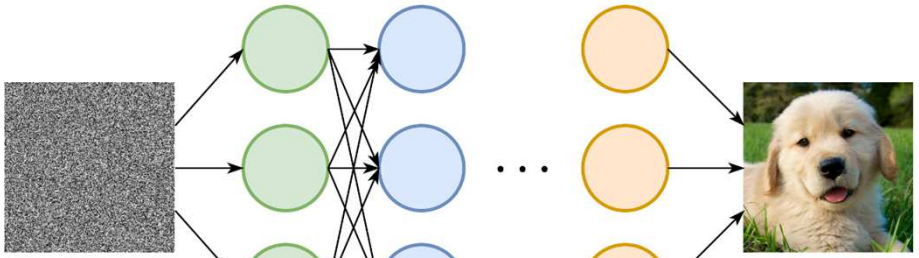
MRF



Autoregressive models



Flow models



Bayesian networks, Markov random fields (MRF), autoregressive models, and normalizing flow models are all examples of likelihood-based models. All these models represent the probability 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.

Score-Based Generative Modeling through SDE

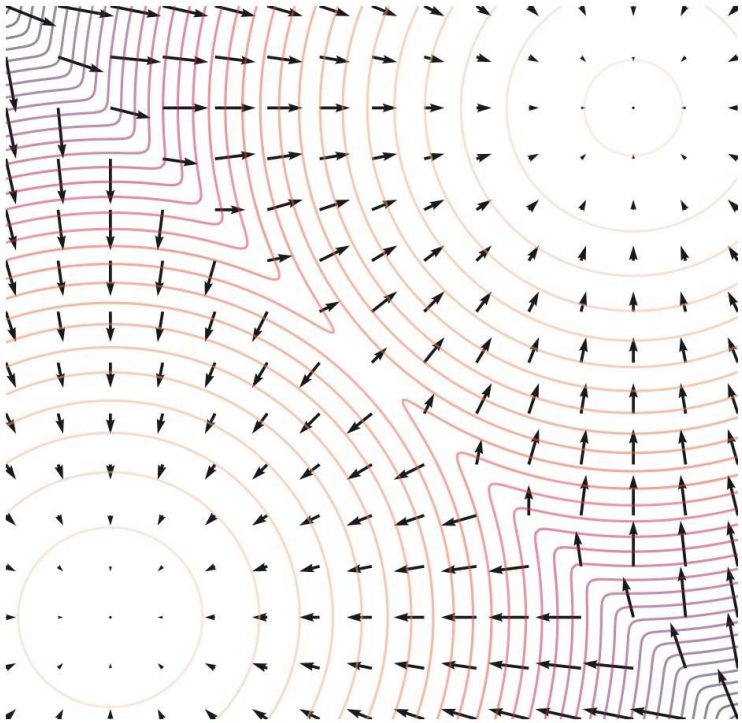
Introduction

➢ Significant limitations.

- **Likelihood-based models :**
  - Require 1) **strong restrictions on the model architecture to ensure a tractable normalizing constant for likelihood computation**,
  - or 2) **must rely on surrogate objectives to approximate maximum likelihood training**.
- **Implicit generative models**
  - Often require **adversarial training**, which is **notoriously unstable** [12] and can lead to **mode collapse** [13].

➢ **Score-based model**

- 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*** [14,15].
- Such score-based models are **not required to have a tractable normalizing constant**, and can be **directly learned by score matching** [16,17].



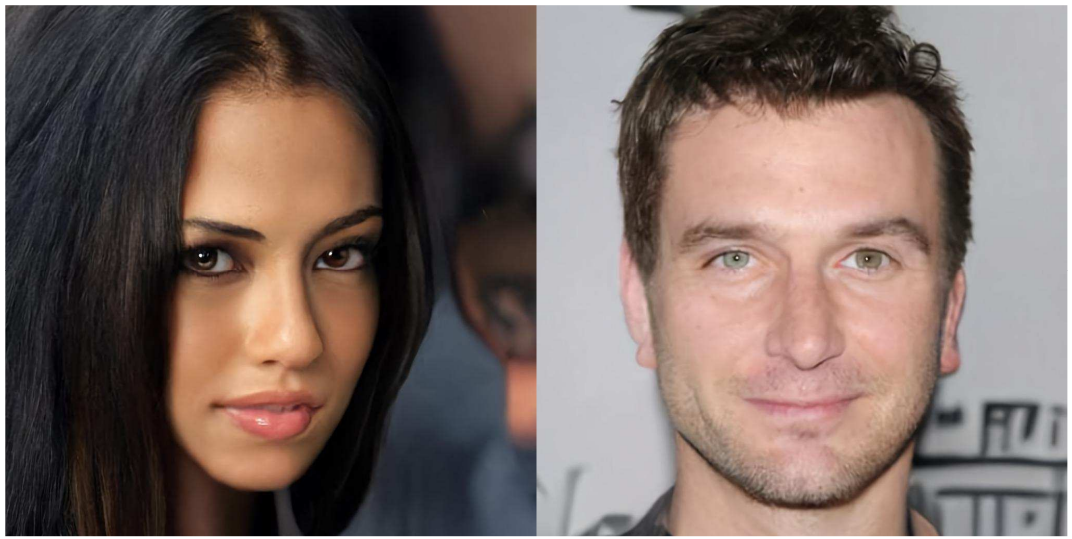
Score function (the vector field) and ***density function*** (contours) of a mixture of two Gaussians.

Score-Based Generative Modeling through SDE

Introduction

➢ **Score-based model**

- Achieve State-of-the-art performance on many downstream tasks and applications.
  - ✓ These tasks include, among others, **image generation** [18,19,20,21,22,23] (Yes, better than GANs!), **audio synthesis** [24,25,26], **shape generation** [27], and **music generation** [28].
- Have connections to normalizing flow models, therefore allowing **exact likelihood computation** and **representation learning.**
- **Modeling and estimating scores** facilitates **inverse problem solving**, with applications such as image inpainting [18,21], image colorization [21], compressive sensing, and medical image reconstruction (e.g., CT, MRI)[29].



1024x1024 samples generated from score-based models [21]

Score-Based Generative Modeling through SDE

Score function, score-based models, and score matching

- Suppose given a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, 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**.

➢ Build such a generative model,

○ 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 or probability mass function.
- Let $f_\theta(\mathbf{x}) \in \mathbb{R}$ be a real-valued function parameterized by a learnable parameter θ.
- Define a pdf via

$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta} \qquad (1)$$

✓ Z_θ : A normalizing constant dependent on θ, such that $\int p_\theta(\mathbf{x}) d\mathbf{x} = 1$

✓ $f_\theta(\mathbf{x})$ is often called an unnormalized probabilistic model, or energy-based model [7].

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

$$\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i). \qquad (2)$$

- It **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 evaluate the normalizing constant Z_θ —a typically intractable quantity for any general $f_\theta(\mathbf{x})$.
- Thus to make maximum likelihood training feasible, likelihood-based models must either restrict their model architectures (e.g., causal convolutions in autoregressive models, invertible networks in normalizing flow models) to make Z_θ tractable, or approximate the normalizing constant (e.g., variational inference in VAEs, or MCMC sampling used in contrastive divergence[30]) which may be computationally expensive.

Score-Based Generative Modeling through SDE

Score function, score-based models, and score matching

➢ Build such a generative model,

○ 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

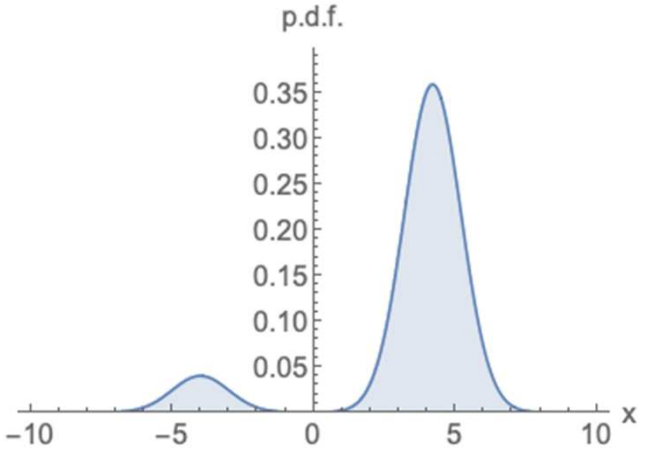
$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

- A model for the score function is called a **score-based model** [18], $s_\theta(\mathbf{x})$.
- The score-based model is learned such that $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$, and can be parameterized without worrying about the normalizing constant.
- For example, we can easily parameterize a score-based model with *the energy-based model* defined in equation (1)

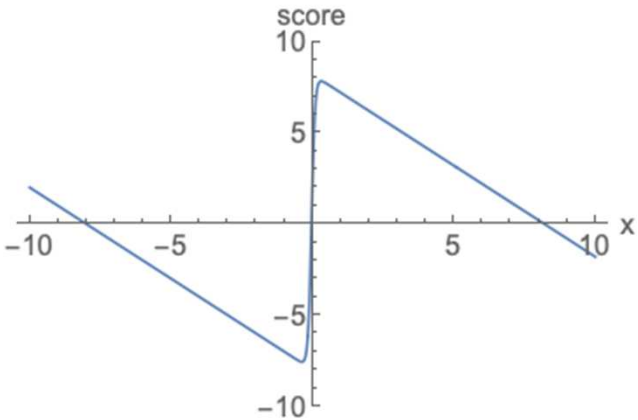
$$\mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_\theta}_{=0} = -\nabla_{\mathbf{x}} f_\theta(\mathbf{x}). \quad (3)$$

❖ $s_\theta(\mathbf{x})$ is independent of the normalizing constant !.

☞ This significantly expands the family of models that we can tractably use, 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.

Score-Based Generative Modeling through SDE

Score function, score-based models, and score matching

➢ We can **train score-based models** by **minimizing the Fisher divergence** between the **model** and the **data distributions**

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] \qquad (5)$$

- Intuitivley, compares the squared l_2 distance between the ground-truth data score and the score-based model.

❖ Fisher divergence between two distributions p and q, defined as

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

➢ Score matching

- It requires access to the unknown data score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. There exists a family of methods called ***score matching*** [16,17,31] that **minimize the Fisher divergence without knowledge of the ground-truth data score**.
- We can **train the score-based model by minimizing a score matching objective**, **without requiring adversarial optimization**.

- **Score matching objectives** can directly **be estimated on a dataset and optimized with stochastic gradient descent**, analogous to the log-likelihood objective for training likelihood-based models (with known normalizing constants).

→ Gives us a considerable amount of **modeling flexibility**.

- Does not require $s_\theta(\mathbf{x})$ to be an actual score function of any normalized distribution—it simply compares the l_2 distance with $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ and $s_\theta(\mathbf{x})$ no additional assumptions on the form of $s_\theta(\mathbf{x})$.
- **Only requirement on the score-based model** is that it should be **a vector-valued function with the same input and output dimensionality**.

Score-Based Generative Modeling through SDE

Langevin dynamics

- Once we have trained a score-based model $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$, we can use an iterative procedure called **Langevin dynamics** [32,33] to **draw samples from it**.

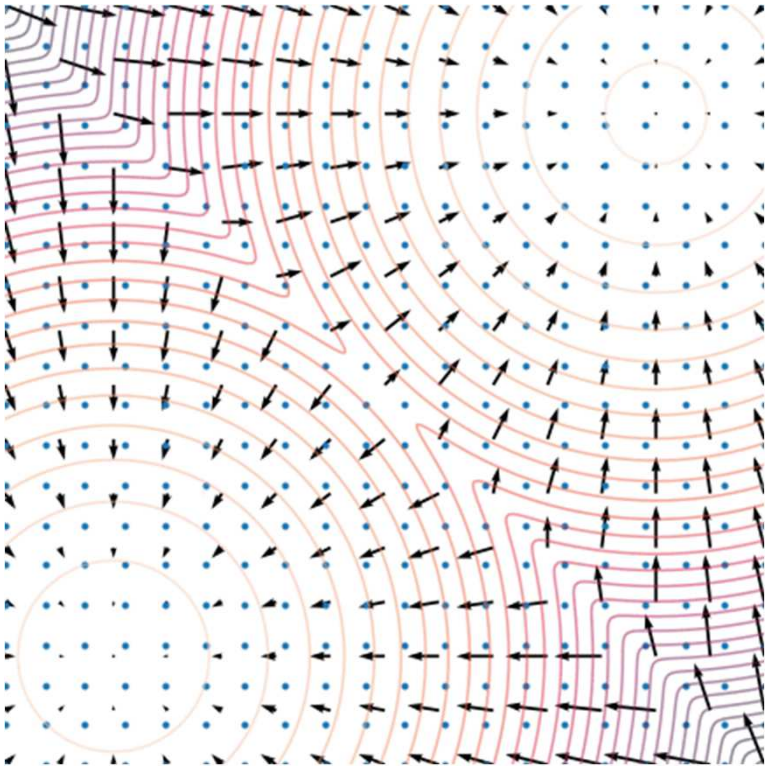
- Langevin dynamics provides an **MCMC** (Markov chain Monte Carlo) procedure to sample from a distribution $p(\mathbf{x})$ using only its score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$.

- Specifically, it initializes the chain from an arbitrary prior distribution $\mathbf{x}_0 \sim \pi(\mathbf{x})$, and then iterates the following

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\mathbf{z}_i, \quad i = 0,1,\dots,K \qquad (6)$$

$$\mathbf{z}_i \sim N(0, I)$$

  - 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.
  - In practice, the error is negligible when ϵ is sufficiently small and K is sufficiently large.



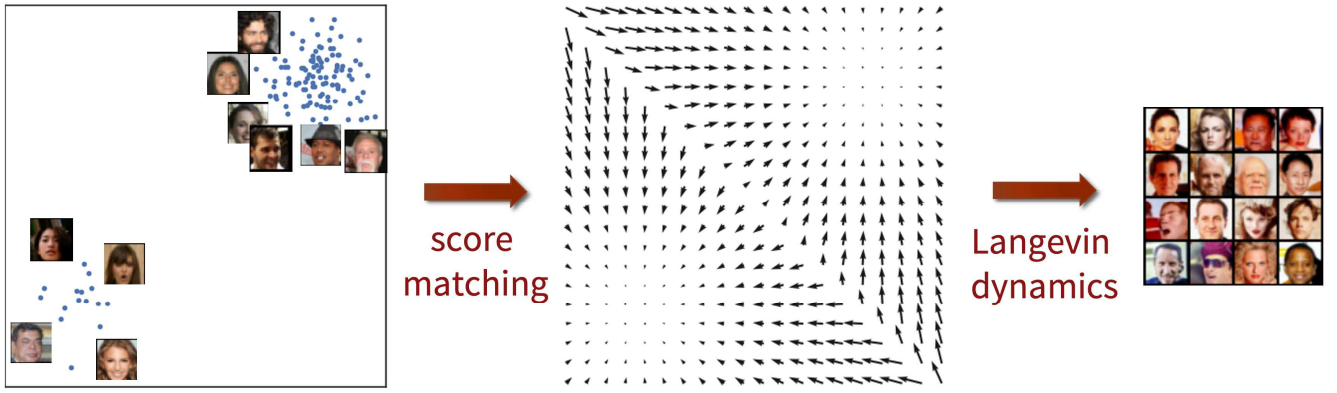
Using Langevin dynamics to sample from a mixture of two Gaussians.

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

Score-Based Generative Modeling through SDE

Naive score-based generative modeling and its pitfalls

- How to train a score-based model with score matching, and then produce samples via Langevin dynamics
- This naive approach has had limited success in practice



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

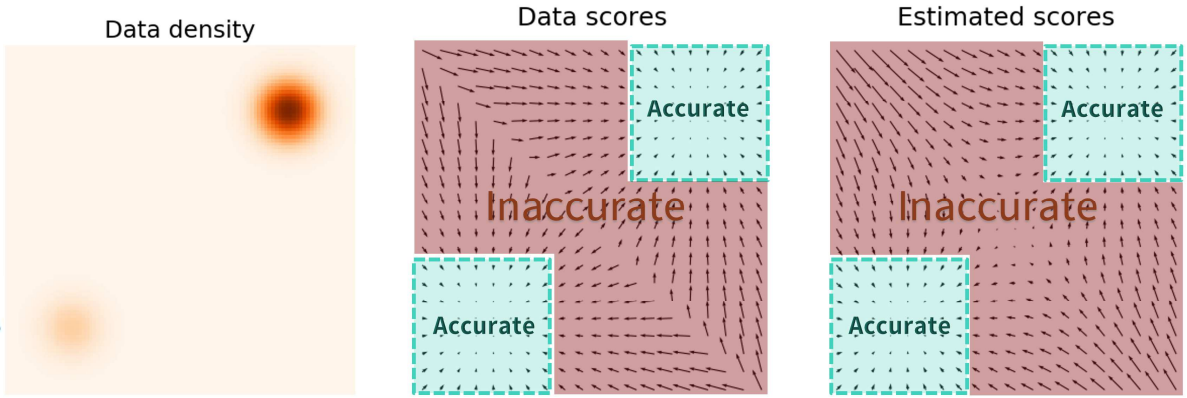
Score-Based Generative Modeling through SDE

Naive score-based generative modeling and its pitfalls

➢ The key challenge

- 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

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



Estimated scores are only accurate in high density regions.

- Since the l_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.

[Note That]

- 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 an inaccurate score-based model will derail Langevin dynamics from the very beginning of the procedure, preventing it from generating high quality samples that are representative of the data.

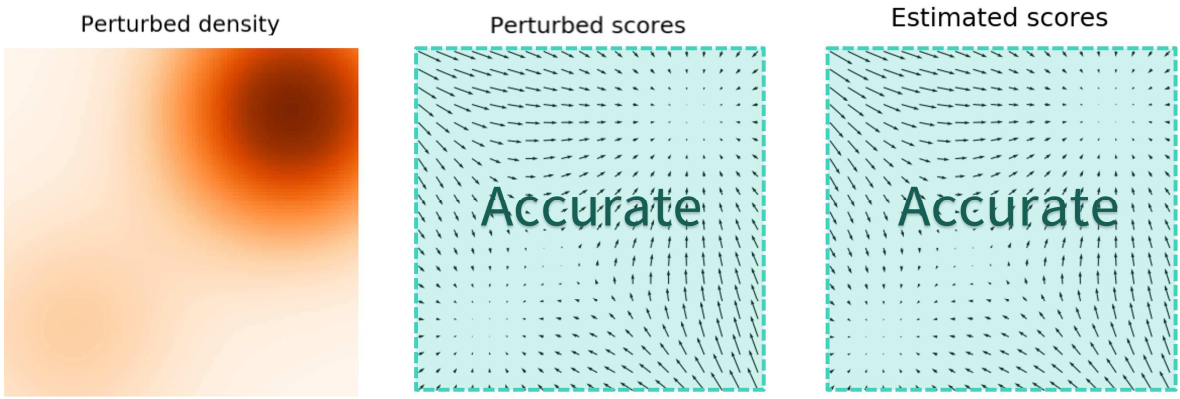
Score-Based Generative Modeling through SDE

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 of 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.

Score-Based Generative Modeling through SDE

Score-based generative modeling with multiple noise perturbations

➢ How do we **choose an appropriate noise scale for the perturbation process**?

- **Larger noise** : 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** : Causes less corruption of the original data distribution, but does not cover the low density regions.

To achieve the best of both worlds

➢ **Multiple scales of noise perturbations simultaneously**

- Suppose we always perturb the data with isotropic Gaussian noise, and let there be a total of L increasing standard deviations $\sigma_1 < \sigma_2 < \cdots < \sigma_L$.
- First, perturb the data distribution $p(\mathbf{x})$ with each of the Gaussian noise $N(0, \sigma_i^2 I)$, $i = 1,2, \ldots, L$ to obtain a noise-perturbed distribution

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

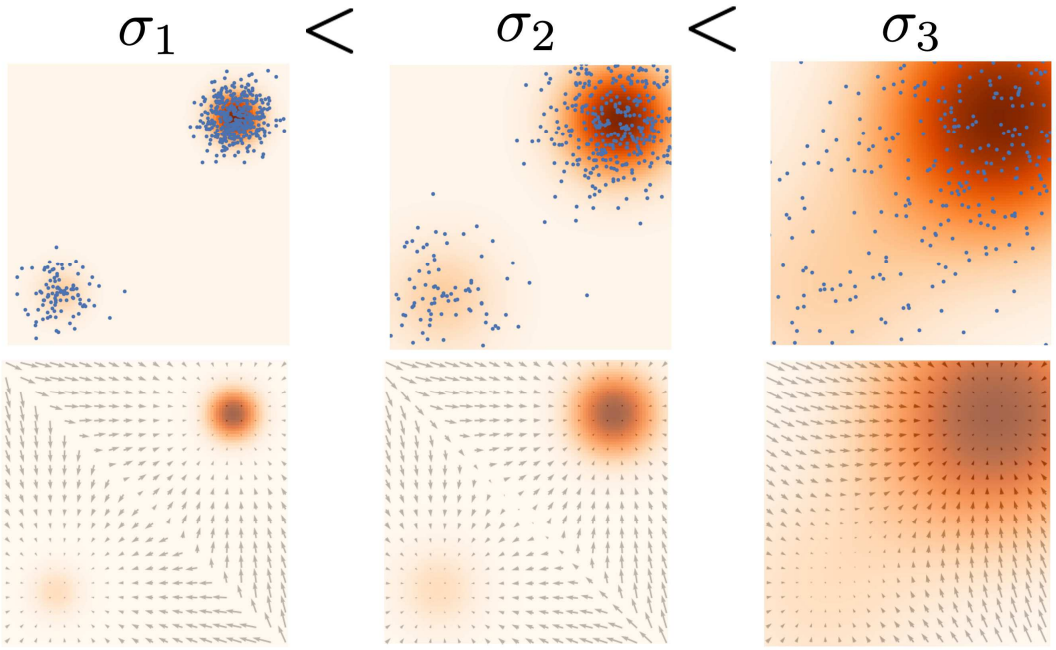
- Note that we can easily draw samples from $p_{\sigma_i}(\mathbf{x})$ by sampling $\mathbf{x} \sim p(\mathbf{x})$ and computing $\mathbf{x} + \sigma_i \mathbf{z}$ with $\boldsymbol{z} \sim N(0, I)$.

Score-Based Generative Modeling through SDE

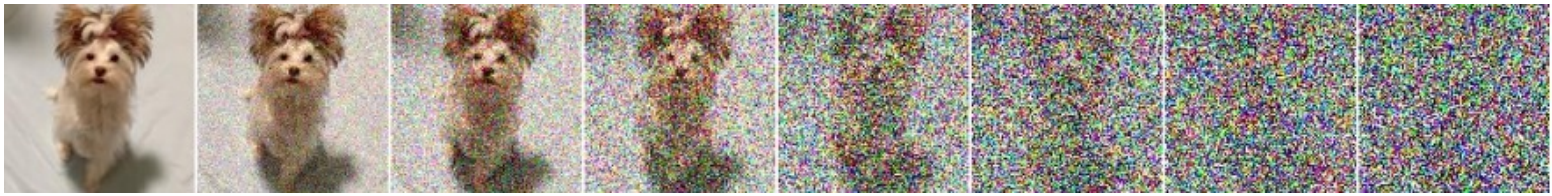
Score-based generative modeling with multiple noise perturbations

➢ Multiple scales of noise perturbations simultaneously

- Next, 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** $s_\theta(\mathbf{x}, i)$ (also called a **Noise Conditional Score Network**, or **NCSN** [18,19,21], when parameterized with a neural network) with score matching, such that $s_\theta(\mathbf{x}, i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$ for all $i = 1,2,\dots,L$.



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.

Score-Based Generative Modeling through SDE

Score-based generative modeling with multiple noise perturbations

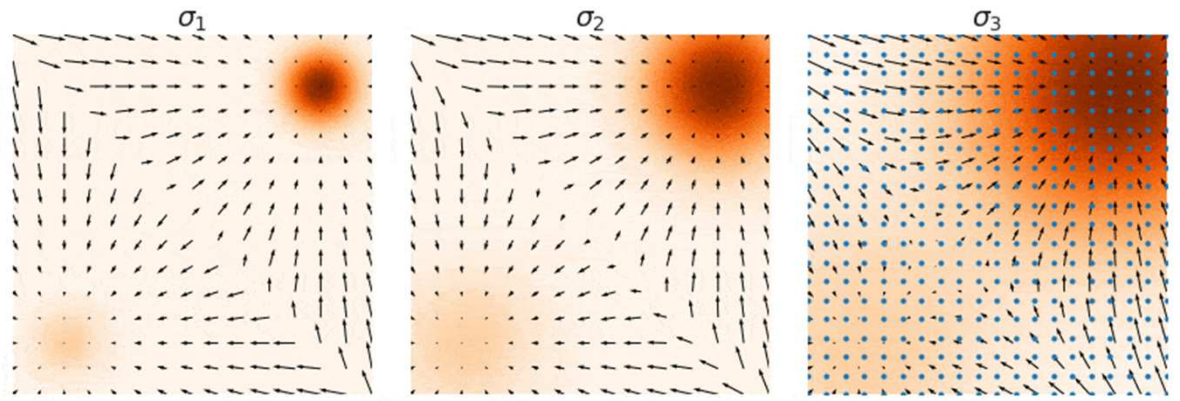
➢ Noise-conditional score-based model $s_\theta(\mathbf{x}, i)$ - Training objective

- The training objective for $s_\theta(\mathbf{x}, i)$ Is a weighted sum of Fisher divergences for all noise scales.

$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, i)\|_2^2], \qquad (7)$$

- $\lambda(i) \in \mathbb{R}_{>0}$ 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 $s_\theta(\mathbf{x})$

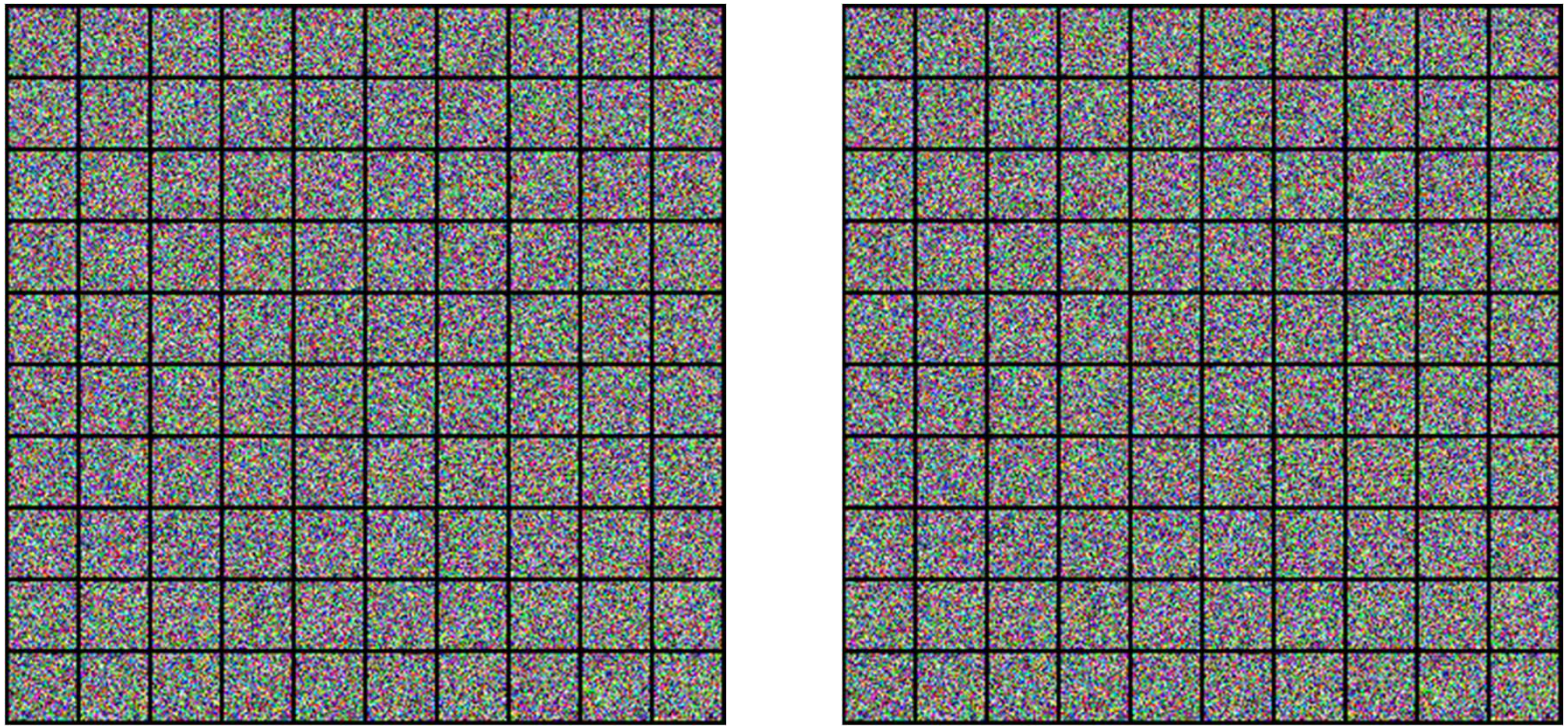
- After training $s_\theta(\mathbf{x}, i)$, we can produce samples from it by running Langevin dynamics for $i = L, L-1, \ldots, 1$ in sequence.
- This method is called **annealed Langevin dynamics** (defined by Algorithm 1 in [18], and improved by [19, 34]), since the noise scale σ_i decreases (anneals) gradually over time



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

Score-Based Generative Modeling through SDE

Score-based generative modeling with multiple noise perturbations

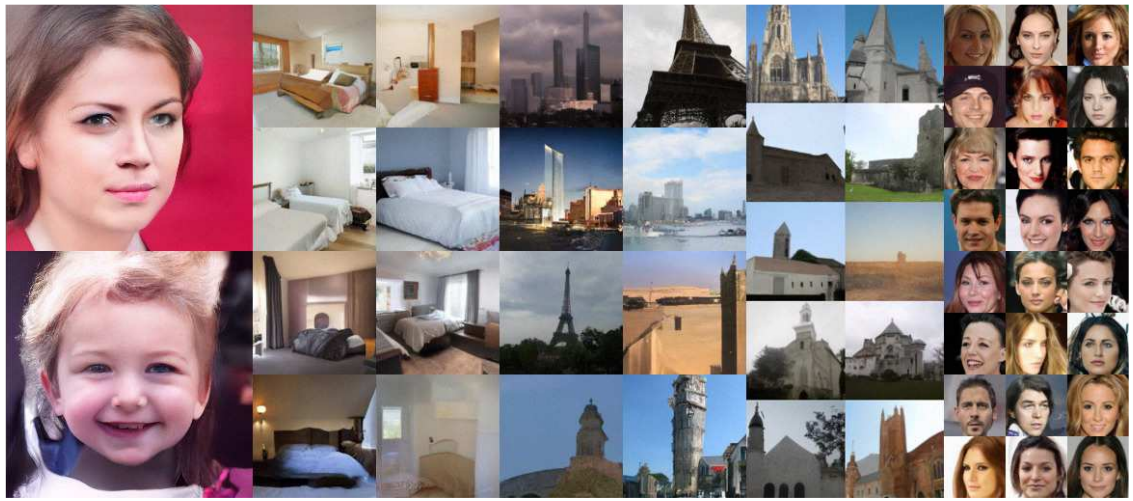


Annealed Langevin dynamics for the Noise Conditional Score Network (NCSN) model (from ref. [18]) trained on CelebA (left) and CIFAR-10 (right). We can start from unstructured noise, modify images according to the scores, and generate nice samples. The method achieved state-of-the-art Inception score on CIFAR-10 at its time.

Score-based generative modeling with multiple noise perturbations

- 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 σ_i being sufficiently small and σ_L comparable to the maximum pairwise distance between all training data points [19]. L is typically on the order of hundreds or thousands.
  - Parameterize the score-based model $s_\theta(\mathbf{x}, i)$ with **U-Net skip connections** [18, 20]
  - Apply exponential moving average on the weights of the score-based model when used at test time [19, 20].

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



Samples from the NCSNv2 [19] 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)

- **Adding multiple noise scales** is critical to the success of score-based generative models.
- By **generalizing the number of noise scales to infinity** [21], we obtain not only **higher quality samples**, but also, among others, **exact log-likelihood computation**, and **controllable generation for inverse problem solving**.

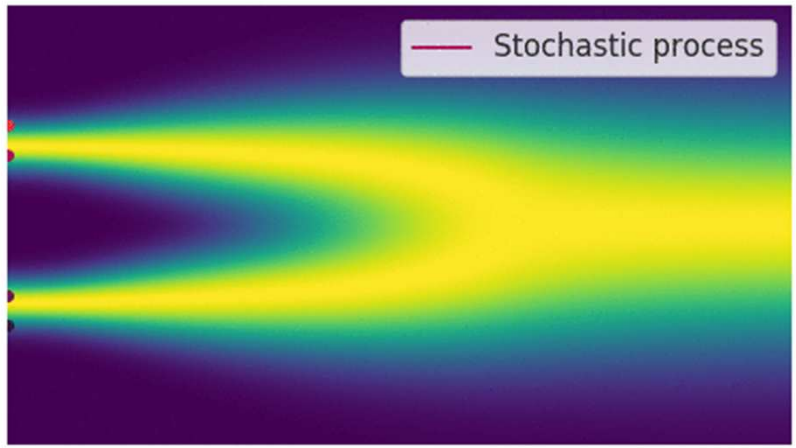
Link	Description
https://colab.research.google.com/drive/1SeXMpILhkJPjXUaesvzEhc3Ke6Zl_zxJ?usp=sharing	• Tutorial of score-based generative modeling with SDEs in JAX + FLAX
https://colab.research.google.com/drive/1dRR_0gNRmfLtPavX2APzUggBuXyjWW55?usp=sharing	• Load our pretrained checkpoints and play with sampling, likelihood computation, and controllable synthesis (JAX + FLAX)
https://colab.research.google.com/drive/120kYYBOVa1i0TD85RjlEkFjaWDxSFUx3?usp=sharing	• Tutorial of score-based generative modeling with SDEs in PyTorch
https://colab.research.google.com/drive/17lTrPLTt_0EDXa4hkbHmbAFQEkpRDZnh?usp=sharing	• 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

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

Perturbing data with an SDE

➢ The number of noise scales to infinity

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



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 differential equations (SDEs).**

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

Perturbing data with an SDE

➢ In general, an **SDE** possesses the following form:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w} \qquad (8)$$

- $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$, a vector-valued function called the **drift coefficient**
- $g(t) \in \mathbb{R}^d$: a real-valued function called the **diffusion coefficient**
- $\mathbf{w}$: Standard Brownian motion
- $d\mathbf{w}$: Can be viewed as **infinitesimal white noise**.

- The solution of a SDE 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 $p_t(\mathbf{x})$ denote the (marginal) probability density function of $\mathbf{x}(t)$.
- Here $t \in [0, T]$ is analogous to $i = 1,2, \dots, L$ 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 σ_L to the data.

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

Perturbing data with an SDE ; Forward SDE

➢ In general, an **SDE** possesses the following form:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w} \qquad (8)$$

- The SDE 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 perturbs data with a Gaussian noise of mean zero and exponentially growing variance

$$d\mathbf{x} = e^t d\mathbf{w} \qquad (9)$$

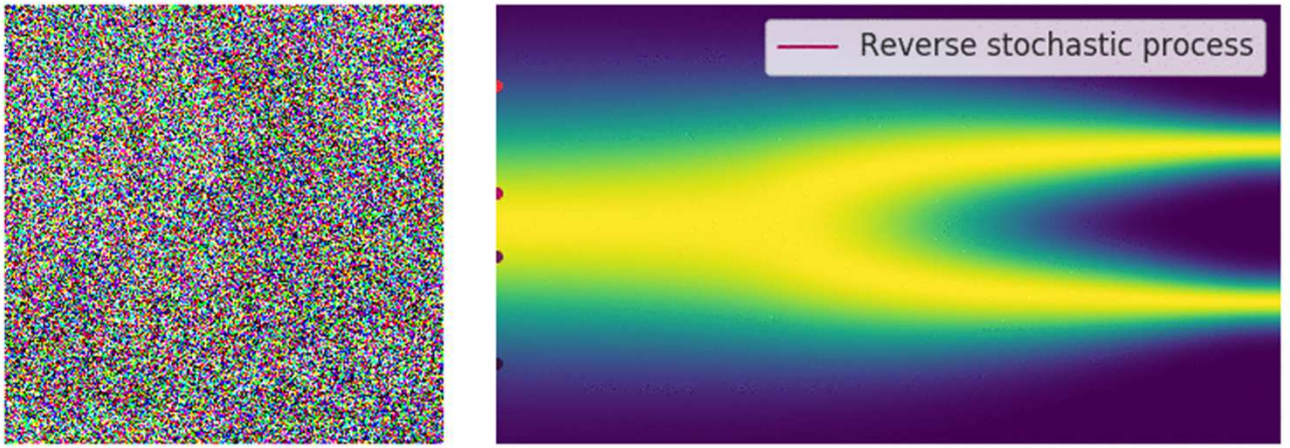
✓ which is analogous to perturbing data with $N(0, \sigma_1^2 I), N(0, \sigma_2^2 I), \ldots, N(0, \sigma_L^2 I)$ when $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ is a geometric progression.

- Therefore, the SDE should be viewed as part of the model, much like $\{\sigma_1, \sigma_2, \ldots, \sigma_L\}$.
- In [21], we provide three SDEs that generally work well for images: the **Variance Exploding SDE** (**VE SDE**), the **Variance Preserving SDE** (**VP SDE**), and the **sub-VP SDE.**

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

Reversing the SDE for sample generation; Reverse SDE

- 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.

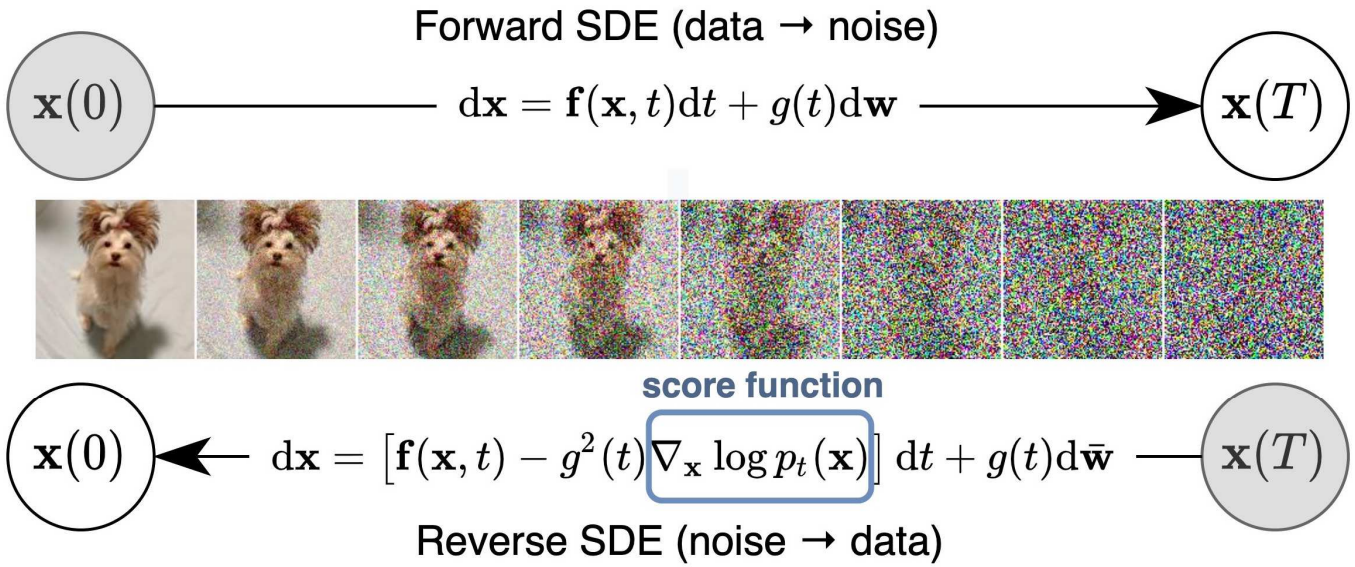
- Importantly, any SDE has a corresponding reverse SDE [35], whose closed form is given by

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

- dt : 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 $p_t(\mathbf{x})$.

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

Reversing the SDE for sample generation; Reverse SDE



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 from noise if we know **the score of the distribution at each intermediate time step.**

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

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

➢ Requirement for solving the Reverse SDE

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

➢ Training objective of score-based model

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

$$\mathbb{E}_{t \in \mathcal{U}(0,T)} \mathbb{E}_{p_t(\mathbf{x})} [\lambda(t) \| \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t) \|_2^2], \qquad (11)$$

  - ✓ $U(0, T)$: Uniform distribution over the time interval $[0, T]$
  - ✓ $\lambda: \mathbb{R} \to \mathbb{R}_{>0}$: A positive weighting function
  - ✓ We use $\lambda(t) \propto 1/\mathbb{E}[\left\| \nabla_{\mathbf{x}(t)} \log p(\mathbf{x}(t) | \mathbf{x}(0)) \right\|_2^2]$ to balance the magnitude of different score matching losses across time.
- Our weighted combination of Fisher divergences can be efficiently optimized with **score matching methods**, such as denoising score matching [17] and sliced score matching [31]

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

➢ Expression of the reverse SDE

- Once our score-based model $s_\theta(\mathbf{x}, t)$ is trained to optimality, we can plug it into the expression of the reverse SDE in (10) to obtain **an estimated reverse SDE**.

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)s_\theta(\mathbf{x}, t)]dt + g(t)d\mathbf{w} \qquad (12)$$

- 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_θ. When the score-based model $s_\theta(\mathbf{x}, t)$ 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.

➢ KL Divergence

- 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_θ under some regularity conditions.

$$\mathbf{KL}\big(p_0(\mathbf{x}) \parallel p_\theta(\mathbf{x})\big) \qquad (13)$$
$$\leq \frac{T}{2}\mathbb{E}_{t\in U(0,T)}\mathbb{E}_{p_t(\mathbf{x})}\big[\lambda(t)\|\nabla_{\mathbf{x}}\log p_t(\mathbf{x}) - s_\theta(\mathbf{x}, t)\|_2^2\big] + \mathbf{KL}(p_T \parallel \pi)$$

Training objective for $s_\theta(\mathrm{x}, t)$ is a continuous weighted combination of Fisher divergences

- Due to this special **connection to the KL divergence** and the **equivalence between minimizing KL divergences and maximizing likelihood for model training**, we call $\lambda(t) = g^2(t)$ **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

How to solve the reverse SDE

➢ Expression of the reverse SDE

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

$$\Delta \mathbf{x} \leftarrow [\mathbf{f}(\mathbf{x}, t) - g^2(t) s_\theta(\mathbf{x}, t)]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta \mathbf{x} \qquad t \leftarrow t + \Delta t$$

Here $\mathbf{z}_t \leftarrow N(0, I)$

- The **Euler-Maruyama method** is qualitatively similar to **Langevin dynamics**—both update $\mathbf{x}$ by following score functions perturbed with Gaussian noise.

- Other **numerical SDE solvers** can be directly employed to solve the reverse SDE for sample generation;
  - ✓ **Milstein method**, and **stochastic Runge-Kutta methods**.
  - ✓ A **reverse diffusion solver** [21] similar to Euler-Maruyama, but more tailored for solving reverse-time SDEs.
  - ✓ **Adaptive step-size SDE solvers** that can generate samples faster with better quality.

How to solve the reverse SDE

➢ Special properties of our reverse SDE

- Two special properties of our reverse SDE that allow for even more flexible sampling methods:

  ① We have an estimate of $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ via our time-dependent score-based model $s_\theta(\mathbf{x}, t)$.

  ② 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 have 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**.

➢ 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,
  - First, we use the **predictor** to choose a proper step size $\Delta t < 0$ and then predict $\mathbf{x}(t + \Delta t)$ based on the current 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** $s_\theta(\mathbf{x}, t + \Delta t)$, so that $\mathbf{x}(t + \Delta t)$ becomes a higher-quality sample from $p_{t+\Delta t}(\mathbf{x})$.

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

How to solve the reverse SDE

➢ Propose **Predictor-Corrector samplers**.

- 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 FID [38] and Inception scores [12])

Method	FID ↓	Inception score ↑
StyleGAN2 + ADA [39]	2.92	9.83
Ours [21]	**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 1024×1024

1024x1024 samples from a score-based model trained on the FFHQ dataset. (원본 그림의 25% 축소)

256 x 256 samples on LSUN bedroom. (원본 그림의 50%)

Probability flow ODE

➢ **Probability flow ODE (Ordinary differential equations)**

- 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-likelihood of score-based generative models.
- Introduce **a sampler based on ordinary differential equations (ODEs)** that allow **for exact likelihood computation**.
- It 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** [21], given by

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x},t) - \frac{1}{2}g^2(t)\,\nabla_{\mathbf{x}}\log p_t(\mathbf{x})\right]dt \qquad (14)$$

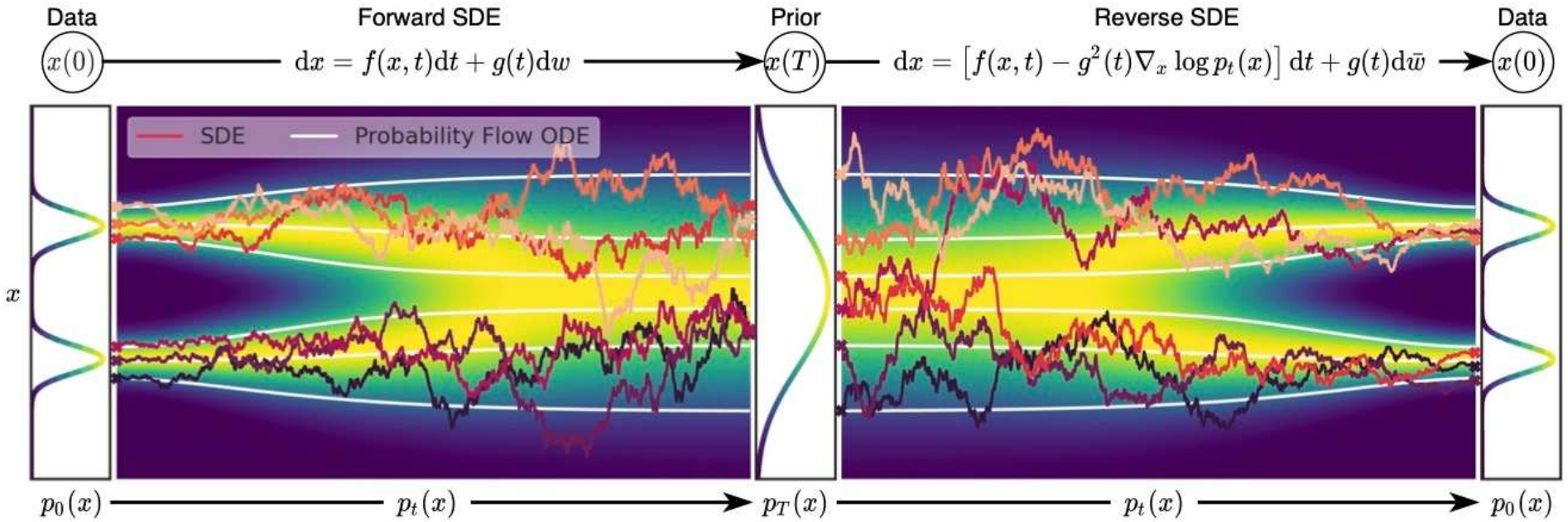
➢ **Trajectories of both SDEs and probability flow ODEs.**

- 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** $\{\boldsymbol{p_t}(\mathbf{x})\}_{\boldsymbol{t}\in[\mathbf{0},\boldsymbol{T}]}$.
- In other words, trajectories obtained by solving the probability flow ODE have the same marginal distributions as the SDE trajectories.

☞ 다음 페이지에 그림

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

Probability flow ODE



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.**

Probability flow ODE

➢ **Unique advantages of Probability flow ODE formulation**

- When $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is replaced by its approximation $s_\theta(\mathbf{x}, t)$, the **probability flow ODE** becomes a special case of a **neural ODE** [40].
- In particular, it is an example of **continuous normalizing flows** [41], 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 same 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, we can leverage the instantaneous change-of-variable formula (Theorem 1 in [40], Equation (4) in [41]) to compute the unknown data density p_0 from the known prior density p_T with numerical ODE solvers.
- 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** $\{\boldsymbol{p_t}(\mathbf{x})\}_{t\in[0,T]}$.
- Our model achieves SOTA **log-likelihoods** on *uniformly dequantized* CIFAR-10 images [21], 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**

Controllable generation for inverse problem solving

➢ Score-based generative models are particularly **suitable for solving inverse problems**.

➢ **Inverse problems** are same as **Bayesian inference problems**

- Let $\mathbf{x}$ 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** $p(\mathbf{y}|\mathbf{x})$.
- The inverse problem is to compute $p(\mathbf{x}|\mathbf{y})$.
- From Bayes' rule

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})} = \frac{p(\mathbf{x})p(\mathbf{y}|\mathbf{x})}{\int p(\mathbf{x})p(\mathbf{y}|\mathbf{x})d\mathbf{x}}$$

- This expression can be greatly simplified by taking gradients with respect to $\mathbf{x}$ on both sides, leading to the following Bayes' rule for score functions:

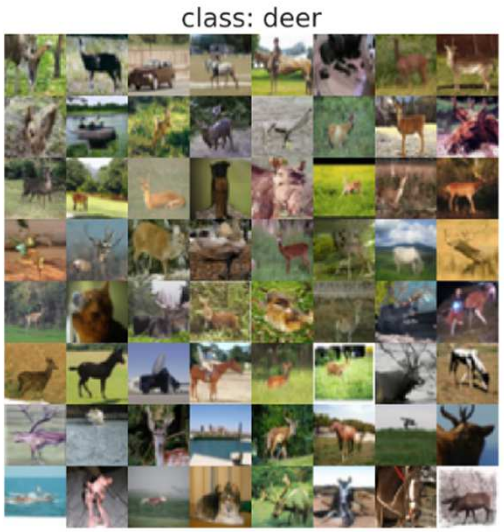
$$\nabla_{\mathbf{x}}\log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}}\log p(\mathbf{x}) + \nabla_{\mathbf{x}}\log p(\mathbf{y}|\mathbf{x}) \qquad (15)$$

- Through **score matching**, we can train a model to estimate the score function of the unconditional data distribution, i.e.,$s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}}\log p(\mathbf{x})$.
- This will allow us to easily compute the posterior score function $\nabla_{\mathbf{x}}\log p(\mathbf{x}|\mathbf{y})$ from the known forward process $p(\mathbf{y}|\mathbf{x})$ via equation (15) , and sample from it with Langevin-type sampling [21].

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

Controllable generation for inverse problem solving

➢ 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 framework). The rest columns show different inpainted images, generated by solving the conditional reverse-time SDE.

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

Controllable generation for inverse problem solving

- Some examples on solving inverse problems for computer vision.

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-scale 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 1024.

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

Connection to diffusion models and others

➢ 3 crucial improvements that can lead to extremely good samples:

1) Perturbing data with **multiple scales of noise**, and 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.

➢ Obtain the state-of-the-art Inception Score on CIFAR-10 in [18] (even better than the best GANs!), and generate high-fidelity image samples of resolution up to 256×256 in [19]

➢ The idea of perturbing data with multiple scales of noise

- Score-based generative models
- Previously used : Simulated annealing, Annealed importance sampling [43], Diffusion probabilistic models [44], Infusion training [45], Variational walkback [46] for generative stochastic networks [47].

✓ Score-based generative modeling : Trained by score matching and sampled by Langevin dynamics

✓ Diffusion probabilistic modeling : Trained by the evidence lower bound (ELBO) and sampled with a learned decoder.

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

Connection to diffusion models and others

➢ Deepr connection between **score-based generative modeling (SGM)** and **diffusion probabilistic modeling (DPM)** By Jonathan Ho and colleagues [20] in 2020

- **ELBO** used for training DPM is essentially equivalent to **the weighted combination of score matching objectives** used in SGM.
- By parameterizing the **decoder** as a sequence of score-based models with a **U-Net** architecture, DPM can also generate high quality image samples comparable or superior to GANs.

➢ Further investigated the relationship between SGM and DPM (ICLR2021)

- **Sampling method** of DPM can be integrated with **annealed Langevin dynamics** of SGM to create a unified and more powerful sampler (the Predictor-Corrector sampler).
- By generalizing the **number of noise scales** to **infinity**, SGM and DPM can both be **viewed as discretizations to stochastic differential equations determined by score functions**.

❖ This work bridges both SGM and DPM into a unified framework**.**

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

Connection to diffusion models and others

- 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** [48].
- The **perspective of diffusion models** is naturally connected to **VAEs**, **lossy compression**, and can be directly incorporated with **variational probabilistic inference**.
- Alternative perspective of diffusion models; Lilian Weng blog https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- *Call as Generative Diffusion Processes* (Proposed to Deepmind researcher)
  - Despite this deep connection between score-based generative models and diffusion models, it is hard to come up with an umbrella term for the model family that they both belong to.
- *For a list of works that have been influenced by score-based generative modeling, researchers at the University of Oxford*

  https://scorebasedgenerativemodeling.github.io/

Remarks

➢ Two major challenges of score-based generative models

1) **Low Sampling speed** since it involves **a large number of Langevin-type iterations**

   - Can be partially solved by using numerical ODE solvers for the probability flow ODE with lower precision (a similar method, denoising diffusion implicit modeling [49]).
   - Possible to learn a direct mapping from the latent space of probability flow ODEs to the image space, as shown in [50].
   - All such methods to date result in **worse sample quality**.

2) **Inconvenient to work with discrete data distributions** since scores are only defined on continuous distributions.

   - Can be addressed by learning an autoencoder on discrete data and performing score-based generative modeling on its continuous latent space [28, 51]