Score-Based Generative Modeling and Guided Diffusion

Contents

1 Introduction **2**

2 Score-Based Modeling **2**

2.1 Score Function 2

2.2 Training Score Networks 2

2.2.1 Parzen Estimator (Kernel Density Estimation) 3

2.2.2 Modified Loss Function 3

2.2.3 Why Use Different Noise Variances? 4

2.3 Langevin Dynamics 4

2.4 Annealed Langevin Dynamics (ALD) 5

3 Stochastic Differential Equations (SDEs) **5**

3.1 Forward Diffusion Process 5

3.2 Reverse Diffusion Process 6

3.3 Connection to ODEs 6

3.4 Denoising Diffusion Implicit Models (DDIM) 6

3.4.1 Derivation of DDIM 6

3.4.2 Interpretation as ODE Discretization 7

3.4.3 Advantages of DDIM 7

3.5 Implementation of DDIM Sampling 8

4 Guided Diffusion **8**

4.1 Simplest Approach to Conditioning 8

4.2 Classifier Guidance 9

4.2.1 Guidance Scale 9

4.2.2 Limitations of Classifier Guidance 9

4.3 Classifier-Free Guidance 9

5 Latent Diffusion Model **10**

6 Cascaded Diffusion **10**

7 Summary **11**
7.1 Score-Based Modeling 11
7.2 Stochastic Differential Equations (SDEs) 11
7.3 Guided Diffusion 11
7.4 Advanced Techniques 12

1 Introduction

Score-based generative modeling is a powerful framework for data generation that uses the gradients of log-probability densities, called *scores*, to guide the generation process. It forms the basis for various modern generative models, including diffusion models and guided diffusion methods.

This document explores the principles of score-based modeling, derivations for kernel density estimation, and guided diffusion techniques, including classifier and classifier-free guidance.

2 Score-Based Modeling

2.1 Score Function

The *score function* of a probability density $p(x)$ is defined as:

$$\nabla_x \ln p(x), \tag{1}$$

which points in the direction of the steepest increase of the log-density. Learning the score function allows us to approximate the data distribution $p(x)$ up to a constant factor.

2.2 Training Score Networks

The score network $s(x, w)$, parameterized by w, is trained to match the true score function $\nabla_x \ln p(x)$. The initial loss function is given by:

$$\mathcal{J}(w) = \frac{1}{2} \int \|s(x, w) - \nabla_x \ln p(x)\|^2 p(x)\, dx. \tag{2}$$

However, in practice, the true data distribution $p(x)$ is not directly available. Instead, we approximate it using the *empirical distribution* derived from a finite dataset $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$:

$$p_D(x) = \frac{1}{N} \sum_{n=1}^{N} \delta(x - x_n), \tag{3}$$

where $\delta(x)$ is the Dirac delta function. The Dirac delta is defined as:

$$\delta(x) = 0 \quad \text{for } x \neq 0, \tag{4}$$

$$\int \delta(x)\, dx = 1. \tag{5}$$

Substituting $p_D(x)$ into the loss:

$$\mathcal{J}(w) = \frac{1}{2N} \sum_{n=1}^{N} \int \|s(x, w) - \nabla_x \ln p(x)\|^2 \delta(x - x_n) dx. \tag{6}$$

The gradient $\nabla_x \ln p_D(x)$, however, is not well-defined because $p_D(x)$ is non-differentiable. To address this, we introduce a noise kernel to smooth out the data distribution.

2.2.1 Parzen Estimator (Kernel Density Estimation)

The smoothed density $q_\sigma(z)$ is obtained by convolving the original data distribution $p(x)$ with the kernel $q(z|x, \sigma)$:

$$q_\sigma(z) = \int q(z|x, \sigma) p(x)\, dx, \tag{7}$$

For a Gaussian kernel:

$$q(z|x, \sigma) = \mathcal{N}(z|x, \sigma^2 I), \tag{8}$$

this corresponds to adding Gaussian noise to the data.

By substituting the empirical distribution $p_D(x)$ into the Parzen estimator:

$$q_\sigma(z) = \int q(z|x, \sigma) \frac{1}{N} \sum_{n=1}^{N} \delta(x - x_n)\, dx, \tag{9}$$

we obtain:

$$q_\sigma(z) = \frac{1}{N} \sum_{n=1}^{N} q(z|x_n, \sigma). \tag{10}$$

2.2.2 Modified Loss Function

Replacing $p(x)$ with $q_\sigma(z)$ in the loss, we get:

$$\mathcal{J}(w) = \frac{1}{2} \int \|s(z, w) - \nabla_z \ln q_\sigma(z)\|^2 q_\sigma(z)\, dz. \tag{11}$$

Using the definition of $q_\sigma(z)$, this becomes:

$$\mathcal{J}(w) = \frac{1}{2N} \sum_{n=1}^{N} \int \|s(z, w) - \nabla_z \ln q(z|x_n, \sigma)\|^2 q(z|x_n, \sigma)\, dz. \tag{12}$$

2.2.3 Why Use Different Noise Variances?

In score-based modeling, we use different noise variances to ensure that the score network learns robustly across the entire data distribution, including regions of low density. This is important for the following reasons:

- **Smoothing Low-Density Regions:** In areas where the data distribution $p(x)$ is sparse, the score function $\nabla_x \ln p(x)$ may be poorly estimated. Adding noise smooths the distribution and ensures that the score network focuses on more well-defined regions.
- **Handling Complex Distributions:** Many real-world data distributions are multimodal or lie on lower-dimensional manifolds within the data space. Adding noise spreads the data points, making the distribution easier to model.
- **Progressive Refinement:** By progressively reducing the noise during sampling, we can generate samples that start from a coarse approximation and become increasingly refined.

The choice of noise levels $\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_L^2$ ensures that the model learns to handle varying levels of uncertainty in the data, making it more robust during sampling.

2.3 Langevin Dynamics

Langevin dynamics is a sampling technique based on the idea of simulating a particle moving under the influence of both deterministic and stochastic forces. For a target distribution $p(x)$, the updates are given by:

$$z_{t+1} = z_t + \eta \nabla_x \ln p(x_t) + \sqrt{2\eta}\xi_t, \tag{13}$$

where:

- η: Step size, controlling the scale of updates.
- $\nabla_x \ln p(x_t)$: The score function, which guides the sample toward regions of high probability density.
- $\xi_t \sim \mathcal{N}(0, I)$: Gaussian noise, which adds stochasticity to explore the distribution.

The score function $\nabla_x \ln p(x)$ ensures that the updates push the sample toward the modes of the distribution, while the Gaussian noise ensures sufficient exploration.

2.4 Annealed Langevin Dynamics (ALD)

Annealed Langevin dynamics extends Langevin dynamics by introducing a schedule of noise variances $\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_L^2$. This method gradually refines the samples by starting with high noise (to explore broadly) and reducing the noise over time (to focus on specific regions). The update rule becomes:

$$z_{t+1} = z_t + \eta \cdot s(z_t, w, \sigma_i^2) + \sqrt{2\eta}\xi_t, \tag{14}$$

where $s(z_t, w, \sigma_i^2)$ is the score network trained at noise level σ_i^2.

The steps in ALD are:

1. Initialize z_0 with a random sample (e.g., Gaussian noise).
2. For each noise level σ_i^2 (from highest to lowest):
   (a) Perform several iterations of Langevin dynamics.
   (b) Update the sample z_t based on the score function $s(z_t, w, \sigma_i^2)$.
3. Return the final refined sample at the lowest noise level σ_L^2.

Annealed Langevin dynamics is particularly effective for score-based generative modeling because it balances global exploration with local refinement.

3 Stochastic Differential Equations (SDEs)

In diffusion models, the generative process can be interpreted as solving a stochastic differential equation (SDE) that gradually transforms a simple distribution (e.g., Gaussian noise) into the data distribution. Conversely, the forward process adds noise to the data, which can also be described by an SDE.

3.1 Forward Diffusion Process

The forward diffusion process progressively adds noise to the data according to an SDE of the form:

$$d\mathbf{z} = f(\mathbf{z}, t)\, dt + g(t)\, d\mathbf{w}_t, \tag{15}$$

where:

- $\mathbf{z} \in \mathbb{R}^d$ is the data vector at time t,
- $f(\mathbf{z}, t)$ is the **drift** coefficient (deterministic part),
- $g(t)$ is the **diffusion** coefficient (scale of the stochastic part),
- $d\mathbf{w}_t$ is the standard Wiener process (Brownian motion).

3.2 Reverse Diffusion Process

To generate samples from the data distribution, we need to reverse the diffusion process. The reverse-time SDE is derived using probability theory, specifically the time reversal of stochastic processes. The reverse-time SDE is given by:

$$d\mathbf{z} = \left[f(\mathbf{z}, t) - g^2(t)\nabla_{\mathbf{z}} \ln p_t(\mathbf{z})\right] dt + g(t)\, d\bar{\mathbf{w}}_t, \tag{16}$$

where:

- $p_t(\mathbf{z})$ is the marginal distribution of $\mathbf{z}$ at time t,
- $\nabla_{\mathbf{z}} \ln p_t(\mathbf{z})$ is the score function at time t,
- $d\bar{\mathbf{w}}_t$ is the standard Wiener process in reverse time.

By estimating the time-dependent score function $\nabla_{\mathbf{z}} \ln p_t(\mathbf{z})$ using a neural network $s(\mathbf{z}, t, w)$, we can simulate the reverse SDE to generate data samples.

3.3 Connection to ODEs

An alternative to simulating the reverse-time SDE is to consider a deterministic process governed by an ordinary differential equation (ODE). The **probability flow ODE** associated with the diffusion process is:

$$\frac{d\mathbf{z}}{dt} = f(\mathbf{z}, t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{z}} \ln p_t(\mathbf{z}). \tag{17}$$

This ODE describes the evolution of the probability density without the stochastic term and can be solved using standard ODE solvers. Solving this ODE allows us to deterministically map a simple distribution to the data distribution.

3.4 Denoising Diffusion Implicit Models (DDIM)

Denoising Diffusion Implicit Models (DDIM) provide a non-stochastic sampling method that accelerates the generation process by leveraging the ODE formulation. Instead of simulating the stochastic reverse diffusion process, DDIMs define a family of discrete-time mappings that approximate the ODE solution.

3.4.1 Derivation of DDIM

Consider the forward diffusion process in discrete time steps $t = 0, 1, \ldots, T$, where Gaussian noise is added to the data:

$$\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{18}$$

with $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$, and β_s is the variance schedule.

The reverse process aims to reconstruct $\mathbf{x}$ from $\mathbf{z}_T$. In DDIM, we define a deterministic mapping from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$:

$$\mathbf{z}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{z}_t - \sqrt{1-\alpha_t}\,\boldsymbol{g}(\mathbf{z}_t, t, w)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}}\,\boldsymbol{g}(\mathbf{z}_t, t, w). \tag{19}$$

Here, $\boldsymbol{g}(\mathbf{z}_t, t, w)$ is a neural network predicting the noise component at time t.

3.4.2 Interpretation as ODE Discretization

The DDIM update rule can be interpreted as a discretization of the probability flow ODE. By rearranging terms, we have:

$$\mathbf{z}_{t-1} = \mathbf{z}_t - \sqrt{\alpha_t}\left(\sqrt{1-\alpha_{t-1}} - \sqrt{1-\alpha_t}\right) \cdot \frac{\boldsymbol{g}(\mathbf{z}_t, t, w)}{\sqrt{1-\alpha_t}}. \tag{20}$$

Assuming $\sqrt{1-\alpha_{t-1}} - \sqrt{1-\alpha_t} \approx \frac{d}{dt}\sqrt{1-\alpha_t} \cdot \Delta t$, the update becomes:

$$\mathbf{z}_{t-1} = \mathbf{z}_t - \left(\frac{d}{dt}\sqrt{1-\alpha_t} \cdot \sqrt{\alpha_t} \cdot \Delta t\right) \cdot \frac{\boldsymbol{g}(\mathbf{z}_t, t, w)}{\sqrt{1-\alpha_t}}. \tag{21}$$

This aligns with the ODE formulation:

$$\frac{d\mathbf{z}}{dt} = -\frac{d}{dt}\left(\sqrt{1-\alpha_t}\right) \cdot \frac{\sqrt{\alpha_t}}{\sqrt{1-\alpha_t}}\boldsymbol{g}(\mathbf{z}_t, t, w) \tag{22}$$

3.4.3 Advantages of DDIM

DDIMs offer several benefits:

- **Deterministic Sampling:** Enables consistent outputs for the same inputs.
- **Faster Inference:** Requires fewer steps compared to stochastic methods.
- **Flexible Trajectory:** Allows interpolation between different sampling paths.

3.5 Implementation of DDIM Sampling

The DDIM sampling algorithm proceeds as follows:

1. **Initialization:** Sample $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
2. **Iterative Updates:** For $t = T, T-1, \ldots, 1$:
   (a) Predict noise $\boldsymbol{g}(\mathbf{z}_t, t, w)$.
   (b) Compute $\mathbf{z}_{t-1}$ using the deterministic update rule.
3. **Output:** Return $\mathbf{x}$ as the generated sample.

By following this procedure, DDIMs efficiently generate high-quality samples without the need for stochastic noise at each step.

4 Guided Diffusion

In many applications, we aim to sample from a conditional distribution $p(x|c)$, where c is a conditioning variable. This variable can represent a class label, a textual description, or some other modality. Applications include:

- **Image Super-Resolution:** Improving the resolution of low-quality images.
- **Image Inpainting:** Filling in missing parts of an image.
- **Text-to-Image Generation:** Creating images based on textual prompts.
- **Video Generation:** Generating videos conditioned on specific inputs.

To incorporate conditioning, a mechanism called *guidance* is used. This involves adding pressure to the generative process to match the conditioning information c while balancing sample diversity.

4.1 Simplest Approach to Conditioning

The simplest approach is to treat c as an additional input to the score network:

$$s(z, w, t, c),$$

where the network is trained on matched pairs (x_n, c_n). However, this method often gives insufficient weight to c or may even ignore it entirely. To address this, guided diffusion methods explicitly control the influence of c during sampling.

There are two main approaches to guidance: Classifier Guidance and Classifier-Free Guidance.

4.2 Classifier Guidance

Classifier guidance leverages a pre-trained classifier $p(c|x)$ to adjust the score function. Using Bayes' theorem:

$$p(x|c) \propto p(c|x)p(x).$$

Taking the log and gradient with respect to x, we obtain:

$$\nabla_x \ln p(x|c) = \nabla_x \ln p(x) + \nabla_x \ln p(c|x).$$

Here:

- $\nabla_x \ln p(x)$: The **unconditional score function**, trained on unconditional data.
- $\nabla_x \ln p(c|x)$: The **classifier gradient**, which adjusts the score toward regions likely to correspond to the conditioning c.

4.2.1 Guidance Scale

To control the importance of the conditioning variable c, we introduce a guidance scale λ:

$$\nabla_x \ln p(x|c, \lambda) = \nabla_x \ln p(x) + \lambda \nabla_x \ln p(c|x). \tag{23}$$

- $\lambda = 0$: Ignores the class label, resulting in unconditional generation.
- $\lambda = 1$: Balances the class label normally.
- $\lambda \gg 1$: Strongly enforces the class label but reduces sample diversity.

4.2.2 Limitations of Classifier Guidance

Classifier guidance requires a separate classifier $p(c|x)$ that can handle noisy inputs (since the data distribution during sampling includes noise). Additionally, large values of λ can overfit the model to c, reducing the diversity of samples.

4.3 Classifier-Free Guidance

Classifier-free guidance avoids the need for a separate classifier by training a single score network $s(x, c, t)$ that learns both conditional and unconditional scores. The score function is:

$$\nabla_x \ln p(x|c) = \nabla_x \ln p(x) + \nabla_x \ln p(c|x).$$

To create a balance between conditional and unconditional generation, we define a convex combination:

$$\text{score}(x, c, \lambda) = \lambda \nabla_x \ln p(x|c) + (1 - \lambda) \nabla_x \ln p(x).$$

Substituting $\nabla_x \ln p(x|c) = \nabla_x \ln p(x) + \nabla_x \ln p(c|x)$, we have:

$$\begin{aligned}
\text{score}(x, c, \lambda) &= \lambda\big(\nabla_x \ln p(x) + \nabla_x \ln p(c|x)\big) + (1 - \lambda)\nabla_x \ln p(x), & (24)\\
&= \lambda\nabla_x \ln p(c|x) + \nabla_x \ln p(x). & (25)
\end{aligned}$$

Training procedure of classifier-free guidance is as follows:

- During training, the conditioning variable c is randomly set to null with a fixed probability (e.g., 10–20%).
- When c is null, the network learns the unconditional score $\nabla_x \ln p(x)$.
- When c is provided, the network learns the conditional score $\nabla_x \ln p(x|c)$.

Classifier-free guidance has several key benefits:

- **Simplified Training:** A single network is trained, eliminating the need for a separate classifier.
- **Improved Diversity:** Convex interpolation between conditional and unconditional scores prevents over-conditioning.
- **Flexibility:** The guidance scale λ can be adjusted dynamically during sampling.

5 Latent Diffusion Model

Instead of applying diffusion directly on pixel-space images, techniques such as latent diffusion operate on a compressed latent space:

1. Train a variational autoencoder (VAE) to map images to a latent representation.
2. Apply diffusion in the latent space using a U-Net architecture.

This approach is more computationally efficient and allows for higher-resolution image generation.

6 Cascaded Diffusion

In some applications, model cascades are used. Following is an example of using model cascades for generating super resolution images. For high-resolution generation, cascading techniques are used:

- Generate low-resolution images (e.g., 64×64).
- Progressively refine the resolution (e.g., $64 \times 64 \to 128 \times 128 \to 256 \times 256$).

7 Summary

This document provides an in-depth exploration of score-based generative modeling and guided diffusion techniques, foundational frameworks in modern generative models like diffusion models.

7.1 Score-Based Modeling

- The **score function** $\nabla_x \ln p(x)$ represents the gradient of the log-probability density, guiding the generation process toward regions of higher data density.
- Training score networks involves approximating the true score function. Due to the non-differentiable nature of the empirical distribution $p_D(x)$, a **Parzen estimator** (kernel density estimation) with Gaussian kernels is used to smooth the data distribution.
- Employing different noise variances $\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_L^2$ ensures robust learning across the entire data distribution, particularly in low-density regions.
- **Langevin Dynamics** and **Annealed Langevin Dynamics (ALD)** are sampling techniques that iteratively refine samples using the score function and progressively decreasing noise levels.

7.2 Stochastic Differential Equations (SDEs)

- The forward diffusion process adds noise to data according to an SDE: $d\mathbf{z} = f(\mathbf{z}, t)\,dt + g(t)\,d\mathbf{w}_t$.
- The reverse diffusion process removes noise, described by the reverse-time SDE: $d\mathbf{z} = [f(\mathbf{z}, t) - g^2(t)\nabla_{\mathbf{z}} \ln p_t(\mathbf{z})]dt + g(t)\,d\bar{\mathbf{w}}_t$.
- The **Probability Flow ODE** provides a deterministic alternative to the reverse SDE: $\frac{d\mathbf{z}}{dt} = f(\mathbf{z}, t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{z}} \ln p_t(\mathbf{z})$.
- **Denoising Diffusion Implicit Models (DDIM)** utilize this ODE formulation for faster, deterministic sampling by defining discrete-time mappings that approximate the ODE solution.

7.3 Guided Diffusion

- Incorporating conditioning variables c (e.g., class labels, textual descriptions) enhances control over the generation process.
- The simplest conditioning approach includes c as an additional input to the score network but may inadequately enforce the conditioning.

- **Classifier Guidance** modifies the score function using gradients from a pre-trained classifier $p(c|x)$, controlled by a guidance scale λ. While effective, it requires a separate classifier and can reduce sample diversity.
- **Classifier-Free Guidance** trains a single score network for both conditional and unconditional generation by randomly omitting c during training. This approach improves diversity and allows dynamic adjustment of the guidance scale.

7.4 Advanced Techniques

- **Latent Diffusion Models** perform diffusion in a compressed latent space obtained via a variational autoencoder (VAE), enhancing computational efficiency and enabling high-resolution image generation.
- **Cascaded Diffusion** involves a sequence of models that progressively apply similar transformations to the data.