Score-Based Generative Modeling Through Stochastic Differential Equations

Ben Eliav
Advised by Prof. Tamir Hazan

July 2024

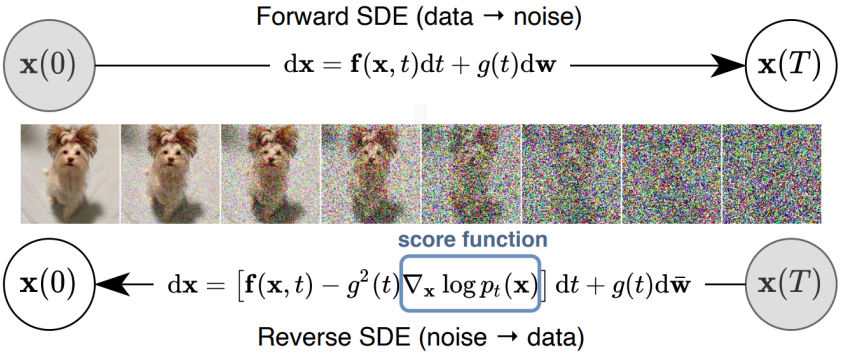

Figure 1: Diffusion models randomly add Gaussian noise to their input until the input is unrecognizable. Afterwards, they learn how to gradually remove the added noise [15].

1 Motivation And Background

Rapid advancements in generative modeling have significantly impacted various fields including computer vision and natural language processing. Generative models such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAE) are remarkably successful at image generation but lack a strong theoretical foundation. This shortcoming is one of the reasons alternative approaches have been tested, such as diffusion models in recent years.

Diffusion models [13] are one of the most prominent modern forms of generative models. Diffusion models are processes where training data ($\mathbf{x}$) is gradually corrupted by adding small units of random noise until it is almost unrecognizable. Evidently, the density of $\mathbf{x}$ starts from being very concentrated and complex, gradually losing its complexity the more noise is being added, until $\mathbf{x}$ is of some prior distribution (generally Gaussian). The model then tries to reverse the process, starting with a random point from the prior distribution and sequentially removing noise until a distinct image is returned from the original image distribution. While all diffusion models follow this paradigm, there had not been a single framework that generalized all diffusion models until 2020. In 2020, Song et al. proposed a novel framework, modeling diffusion processes as continuous time stochastic processes that are the solution to stochastic differential equations (SDEs). Figure 1 illustrates the diffusion process.

Since 2020, the field of diffusion models has grown rapidly. Models such as GLIDE [7] and later StableDiffusion [10] offered the possibility of generating any image based on a query, using conditional generation based on the idea of classifier-free guidance in diffusion models. This opened the door for many new research directions in the field, as diffusion models are now being used for almost any generation task, including inpainting, sharpening, class-based generation and more [7, 10] as shown in Figure 2, allowing generation of other types of information such as audio, point clouds and videos [19].

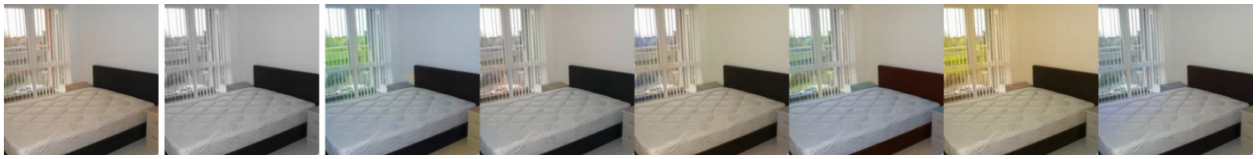

Figure 2: Conditional diffusion. The first image to the left is the original but the input to the model is the grayscale version, second to the left. The diffusion model generates a several possible colored versions, all which appear to be reasonable [15].

2 Research Stages

1. **Literature Review and Problem Understanding**
   - Conduct a literature review to better understand the current state of research in the field of score-based generative modeling and diffusion models.
   - Understand shortcomings of previous literature, understand where different papers come together and what separates their ideas.
2. **Basic Implementation of Baseline Models**
   - Gather simple data sets of images, making sure they are properly preprocessed.
   - Using the PyTorch module of Python, model the diffusion processes on simple datasets to see the different models' capabilities.
3. **Mathematical Analysis of Current Models**
   - Compare the different models from a theoretical perspective, see how they can all be modeled under the same SDE paradigm.
   - See how the SDE framework relates to previous frameworks and confirm its theoretical foundations.
4. **Application in Various Fields**
   - Explore the use of diffusion models in different areas.
5. **Enhancement of the Model**
   - Attempt to use different forward and backward methods than those proposed in Song et al. [15], perhaps attempting more complex models.
   - Attempt to shorten the sampling process to make the model less computationally heavy.
   - Attempt different functions as f and g in the original differential equation, as well as different weighting functions λ.
   - Attempt to specifically improve conditional generation using advances in other fields such as Natural Language Processing or Image Segmentation.
6. **Evaluation of Enhancements**
   - Compare enhanced models to the baseline proposed by Song et al. [15] on different, more diverse data sets than in step 2, using different evaluation metrics such as the Inception Score (IS) and Fréchet Inception Distance (FID).
7. **Future Work**
   - Reflect on shortcomings of our research due to limited time and how we can address them in future work.
   - Identify potential future directions in the field of score-based generative modeling.

3 Score-Based Generative Modeling

Score matching is learning the *score function* of a distribution with an unknown density. Note that in classical statistics, the score function is usually the gradient of the density function with respect to its parameters. In our case, the score function is the gradient of the density function with respect to the input: $\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x})$, where we will typically treat $\mathbf{x}$ as a vector in $\mathbb{R}^d$. If it were possible to know the exact density of the input, one could easily generate new samples from the input distribution by sampling according to the density function. Because this is impossible, we must instead try to *match* the distribution

by learning a network to approximate the value of the density function. Many times, matching the *score* function is preferable to the density as the true unknown density $p_{\text{data}}(\mathbf{x})$ often has the following form:

$$p_{\text{data}}(\mathbf{x}) = \frac{f(\mathbf{x})}{\int_{\mathbb{R}^n} f(\mathbf{x}) d\mathbf{x}} \tag{3.1}$$

meaning the numerator can be easily computed but the denominator is intractable as it requires going over all values of f in $\mathbb{R}^n$. However:

$$\log p_{\text{data}}(\mathbf{x}) = \log f(\mathbf{x}) - \underbrace{\log\left(\int_{\mathbb{R}^n} f(\mathbf{x}) d\mathbf{x}\right)}_{\text{constant}} \tag{3.2}$$

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \nabla_x \log f(\mathbf{x}) \tag{3.3}$$

Training a network $\mathbf{s}_\theta$ to match the score is as simple as minimizing the mean of squared errors between $\mathbf{s}_\theta$ and the original score function $\mathbf{s}$:

$$\text{MSE} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\left\|\mathbf{s}_\theta(\mathbf{x}) - \mathbf{s}(\mathbf{x})\right\|^2\right] \tag{3.4}$$

From this equation, it is possible to derive the following equivalent loss function that removes constant values [17]:

$$\tilde{L} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\frac{1}{2}\left\|\mathbf{s}_\theta(\mathbf{x})\right\|^2 + \text{tr}\left(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})\right)\right] \tag{3.5}$$

which is intractable due to the computation of the divergence of the complicated network $\mathbf{s}_\theta$ which could be of a very high dimension. Hence, one popular solution to this problem is "denoising" by slightly perturbing $\mathbf{x}$ by the distribution $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \sim \mathcal{N}(\mathbf{x}, \sigma^2)$. We can define the marginal density as $q_\sigma(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}$. The reasoning behind this method is that if one estimates the score of q_σ instead of p_{data}, it will still point in the same general direction as the original data before the perturbation. It can be shown that training according to the loss function in Equation 3.6 is equivalent to finding a score matcher for the distribution $q_\sigma(\mathbf{x})$, meaning their optimal networks are the same. However, $\nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ is only true if the noise scale is small enough that for all $\mathbf{x}$, $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ [17].

$$L_{\text{denoise}}(\theta) := \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \tilde{\mathbf{x}}\sim q_\sigma(\cdot,\mathbf{x})}\left[\left\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\right\|^2\right] \tag{3.6}$$

Once given a score-matcher, it is possible under certain conditions (continuous density, defined over entire space) to sample from any distribution using Langevin Dynamics [4], which are the realization of the Fokker-Planck Equation [9]. The Fokker Planck equation defines the evolution over time of the probability distribution of a particle under stochastic motion, such as Brownian motion.

Continuous Langevin Dynamics claim that if we allow $\mathbf{x}_0 \sim \pi$ (an arbitrary prior distribution) to advance according to the SDE in Equation 3.7 then after a long enough time, $\mathbf{x}_t$ will eventually be distributed according to the density function p_{data}:

$$d\mathbf{x}_t = \nabla_{\mathbf{x}_t} \log p_{\text{data}}(\mathbf{x}_t) dt + \sqrt{2}\, d\mathbf{w} \tag{3.7}$$

where $\mathbf{w}$ denotes standard Brownian motion. A discretized version of this SDE can be derived as follows, using the Euler-Maruyama method [6]:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\epsilon}{2} \nabla_{\mathbf{x}_t} \log p_{\text{data}}(\mathbf{x}_t) + \sqrt{\epsilon}\, \mathbf{z}_t, \tag{3.8}$$

where $z_t \sim \mathcal{N}(0, \mathbf{I})$.

Theorem 1. *If Langevin Dynamics reach the ground-truth distribution p_{data}, they have converged.*

Proof. Assume $d\mathbf{x}_t = \nabla_{\mathbf{x}_t} \log p_{\text{data}}(\mathbf{x}_t)dt + \sqrt{2}d\mathbf{w}$. Define ρ_t the marginal distribution of $\mathbf{x}_t$ at time t based on $\mathbf{x}_0 \sim \pi$ and the provided SDE. According to the Fokker-Planck equation [9]:

$$\frac{\partial \rho_t(\mathbf{x}_t)}{\partial t} = -\nabla \cdot (\rho_t(\mathbf{x}_t)\nabla_{\mathbf{x}_t} \log p_{\text{data}}(\mathbf{x}_t)) + \Delta \rho_t(\mathbf{x}_t)$$

where $\nabla\cdot$ denotes the divergence (trace of the Jacobian matrix) and Δ denotes the Laplacian (trace of the Hessian matrix). All gradients refer to the partial derivatives according to $\mathbf{x}_t$, unless explicity denoted by $\frac{\partial}{\partial t}$. Using the identity $\Delta f = \nabla \cdot (\nabla f)$, we can further develop the Fokker-Planck equation:

$$\begin{aligned}
\frac{\partial \rho_t(\mathbf{x}_t)}{\partial t} &= -\nabla \cdot (\rho_t(\mathbf{x}_t)\nabla \log p_{\text{data}}(\mathbf{x}_t)) + \Delta \rho_t(\mathbf{x}_t) \\
&= -\nabla \cdot (\rho_t(\mathbf{x}_t)\nabla \log p_{\text{data}}(\mathbf{x}_t)) + \nabla \cdot (\nabla \rho_t(\mathbf{x}_t)) \\
&= \nabla \cdot (\nabla \rho_t(\mathbf{x}_t) - \rho_t(\mathbf{x}_t)\nabla \log p_{\text{data}}(\mathbf{x}_t) \\
&= \nabla \cdot (\rho_t(\mathbf{x}_t)\nabla \log \rho_t(\mathbf{x}_t) - \rho_t(\mathbf{x}_t)\nabla \log p_{\text{data}}(\mathbf{x}_t) \\
&= \nabla \cdot (\rho_t(\mathbf{x}_t) (\nabla \log \rho_t(\mathbf{x}_t) - \nabla \log p_{\text{data}}(\mathbf{x}_t))) \\
&= \nabla \cdot \left(\rho_t(\mathbf{x}_t)\nabla \log \frac{\rho_t(\mathbf{x}_t)}{p_{\text{data}}(\mathbf{x}_t)} \right)
\end{aligned} \tag{3.9}$$

Hence, if the process reaches $\rho_t = p_{\text{data}}$, there will be no change in ρ_t, as $\frac{\partial \rho_t}{\partial t} = \nabla \cdot (\rho_t \nabla \log 1) = \nabla \cdot 0 = 0$. Note this goes in both directions, as the above equality holds for all $\mathbf{x}$ and the only way to make the distribution stationary is if the final expression is 0, which is only possible if $\rho_t = p_{\text{data}}$. □

Theorem 2. *Langevin Dynamics approach p_{data}. If p_{data} satisfies the Log-Sobolev Inequality (LSI) with constant $\alpha > 0$, then $D_{KL}(\rho_t \,||\, p_{data}) \leq e^{-2\alpha t} D_{KL}(\rho_0 \,||\, p_{data})$, providing exponentially fast convergence.*

Proof. We will analyze the change in the KL divergence between the distribution at time t – ρ_t – and the ground truth distribution p_{data}. Assume the input data is a vector of dimension d, $\mathbf{x} \in \mathbb{R}^d$. Our proof is based on the proof provided in [16].

$$\begin{aligned}
\frac{\partial}{\partial t} D_{\text{KL}}(\rho_t \,||\, p_{\text{data}}) &= \frac{\partial}{\partial t} \int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \log \frac{\rho_t(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} \, d\mathbf{x} \\
&= \int_{\mathbb{R}^d} \frac{\partial}{\partial t} \left(\rho_t(\mathbf{x}) \log \frac{\rho_t(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} \right) d\mathbf{x} \\
&= \int_{\mathbb{R}^d} \frac{\partial}{\partial t} (\rho_t(\mathbf{x})) \log \frac{\rho_t(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} \, d\mathbf{x} + \int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \frac{\partial}{\partial t} \log \frac{\rho_t(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} \, d\mathbf{x}
\end{aligned}$$

For the second element of the sum:

$$\begin{aligned}
\int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \frac{\partial}{\partial t} \log \frac{\rho_t(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} \, d\mathbf{x} &= \int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \frac{\frac{\partial}{\partial t} \rho_t(\mathbf{x})}{\rho_t(\mathbf{x})} \, d\mathbf{x} \\
&= \int_{\mathbb{R}^d} \frac{\partial}{\partial t} \rho_t(\mathbf{x}) = \frac{\partial}{\partial t} \int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \, d\mathbf{x} = \frac{\partial}{\partial t} 1 = 0
\end{aligned}$$

Meaning we remain only with the first element.

$$\begin{aligned}
\frac{\partial}{\partial t} D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right) &= \int_{\mathbb{R}^d} \frac{\partial}{\partial t}\left(\rho_t(\mathbf{x})\right) \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}\, d\mathbf{x} + \underbrace{\int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \frac{\partial}{\partial t} \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}\, d\mathbf{x}}_{=0} \\
&\overset{3.9}{=} \int_{\mathbb{R}^d} \nabla \cdot \left(\rho_t(\mathbf{x}) \nabla \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}\right) \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}\, d\mathbf{x} \\
&= \int_{\mathbb{R}^d} \sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{x}_i} \left(\rho_t(\mathbf{x}) \frac{\partial \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}}{\partial \mathbf{x}_i}\right) \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}\, d\mathbf{x} \\
&= \sum_{i=1}^{n} \int_{\mathbb{R}^d} \frac{\partial}{\partial \mathbf{x}_i} \left(\rho_t(\mathbf{x}) \frac{\partial \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}}{\partial \mathbf{x}_i}\right) \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}\, d\mathbf{x} \\
&\overset{(*)}{=} \sum_{i=1}^{n} - \int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \frac{\partial \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}}{\partial \mathbf{x}_i} \left(\frac{\partial \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})}}{\partial \mathbf{x}_i}\right) d\mathbf{x} \\
&= - \int_{\mathbb{R}^d} \rho_t(\mathbf{x}) \left\| \nabla \log \frac{\rho_t(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x})} \right\|^2 d\mathbf{x} = -J_{p_{\mathrm{data}}}(\rho_t),
\end{aligned}$$

where $-J_{p_{\mathrm{data}}}(\rho_t)$ is the negative Fisher information, which is always negative as the expression in the integral is always positive. $(*)$ is due to integration by parts, assuming boundary conditions, $\lim_{\|\mathbf{x}\|\to\infty} \rho_t(\mathbf{x}) = \lim_{\|\mathbf{x}\|\to\infty} p_{\mathrm{data}}(\mathbf{x}) = 0$. This means the KL Divergence decreases the longer we allow the Langevin Dynamics to run. A smaller KL Divergence means the density ρ_t is becoming more similar to p_{data}, and we know that it will only be 0 if $\rho_t = p_{\mathrm{data}}$. If p_{data} satisfies the LSI condition, the equivalent inequality can be derived [16]: $\frac{1}{2\alpha} J_{p_{\mathrm{data}}}(\rho_t) \geq D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right)$. Plugging this into what we've shown previously:

$$\begin{aligned}
&\frac{\partial}{\partial t} D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right) \leq -2\alpha D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right) \\
&\Leftrightarrow \frac{\frac{\partial}{\partial t} D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right)}{D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right)} \leq -2\alpha \\
&\Leftrightarrow \log D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right) - \log D_{\mathrm{KL}}\left(\rho_0 \,||\, p_{\mathrm{data}}\right) \leq -2\alpha t \\
&\Leftrightarrow \log D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right) \leq \log D_{\mathrm{KL}}\left(\rho_0 \,||\, p_{\mathrm{data}}\right) - 2\alpha t \\
&\Leftrightarrow D_{\mathrm{KL}}\left(\rho_t \,||\, p_{\mathrm{data}}\right) \leq e^{-2\alpha t} D_{\mathrm{KL}}\left(\rho_0 \,||\, p_{\mathrm{data}}\right),
\end{aligned}$$

where the justification for the second step is integration of both sides from 0 to t.

□

The two theorems above combined demonstrate how Langevin Dynamics rely on the Fokker Planck equation and are essentially the application of the Fokker Planck equation. Overall, the pipeline of a Score Matching Langevin Dynamics (SMLD) model is to train a score matching model $\mathbf{s}_\theta \approx \nabla_{\mathbf{x}} \log p_{\mathrm{data}}$ using any loss function, such as L_{denoise} as shown in Equation 3.6 [14, 15]. Once the model is trained, we can use it as a substitute for $\nabla_{\mathbf{x}} \log p_{\mathrm{data}}$ in Equation 3.8.

3.1 Annealed Langevin Dynamics

While Langevin Dynamics have the desired properties required to sample from p_{data} given just its score function $\nabla \log p_{\mathrm{data}}$, there are still practical limitations that hinder the convergence of Score-Matching Langevin Dynamics to the target distribution, such as *low-density regions* in the support of p_{data} and *slow mixing* in the case of mixture models [14, 18].

Regions in the support with low density most likely will not have representation in training data. This makes the objective function in Equation 3.6 unable to take these regions in consideration, and a score

matching model could map points in this region to any value without suffering from any loss. This is a problem because the model will effectively be mapping these points to random new points in the space, potentially taking much longer to converge or not converging at all. We can see in Figure 3 that the model does a good job predicting the score in areas with high density, but does a poor job in areas with low density due to there being little to no training data in those areas to begin with.

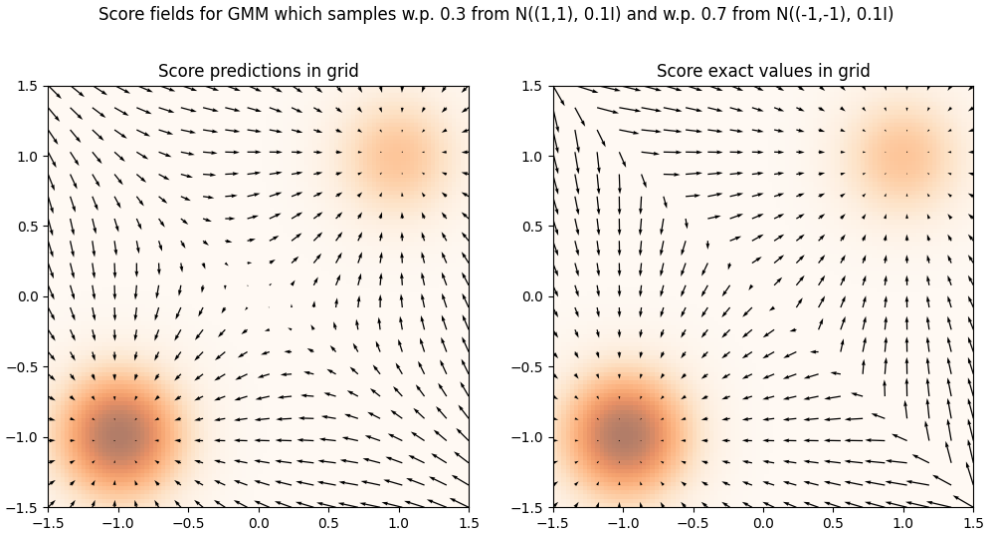

Figure 3: Predicted scores vs. actual score values in a GMM. Areas with higher density are colored darker orange.

Additionally, if the given probability distribution is divided into different supports with very little overlap, such as a GMM with probability 0.1 for a point to be sampled from $p_1 := \mathcal{N}((1,1), 0.01\mathbf{I})$ and probability 0.9 to be sampled from $p_2 := \mathcal{N}((-1,-1), 0.01\mathbf{I})$, the model will not be able to accurately compute the score. In this case, the score is equal to $\nabla_{\mathbf{x}} \log p(\mathbf{x}) = \nabla_{\mathbf{x}} \log (0.1p_1(\mathbf{x}) + 0.9p_2(\mathbf{x}))$. In the region around $(1,1)$, the function p_2 is practically constant making $\nabla_{\mathbf{x}} \log p(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log(0.1p_1(\mathbf{x}))$. The gradient of the logarithm removes all constant factors, such that $\nabla_{\mathbf{x}} \log p(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_1(\mathbf{x})$. Practical limitations essentially strip information from the model, such that the model cannot recover the probability of sampling from each member of the GMM. An example of this is provided in Figure 4, and this phenomenon is known as *slow mixing* of Langevin Dynamics.

Annealed Langevin Dynamics is a practical solution to the above limitations, which separates the sampling into several steps. Instead of training a Score Network that attempts to approximate the function $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$, we train a Noise Conditional Score Network (NCSN) that approximates the score of the *perturbed* data distribution $\nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x})$. The network $\mathbf{s}_\theta$ will be a function of both the input $\mathbf{x}$ and the noise scale σ. As $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ is a normal distribution with mean $\mathbf{x}$ and covariance matrix $\sigma^2\mathbf{I}$, its score function can be calculated as $-\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}$. We can use the following MSE-based loss function to train the NCSN:

$$\ell_{\text{denoise}}(\theta\,;\sigma) := \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\mathbb{E}_{\tilde{\mathbf{x}}\sim q_\sigma(\cdot|\mathbf{x})}\left[\left\|\mathbf{s}_\theta(\tilde{\mathbf{x}},\sigma) + \frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}\right\|_2^2\right] \tag{3.10}$$

Given a set of decreasing noise scales, $\{\sigma_t\}_{t=1}^T$, we can define the final loss function as a weighted average of all noise-specific denoising loss values:

$$L_{\text{denoise}}(\theta; \{\sigma_t\}_{t=1}^T) := \frac{1}{T}\sum_{t=1}^{T} \lambda(\sigma_t)\ell_{\text{denoise}}(\theta\,;\sigma), \tag{3.11}$$

where λ is a scaling function that balances the contributions of each noise scale to the overall loss. With this adjustment, for low values of σ, ℓ_{denoise} won't dominate the value of the loss function due to it being a much smaller denominator.

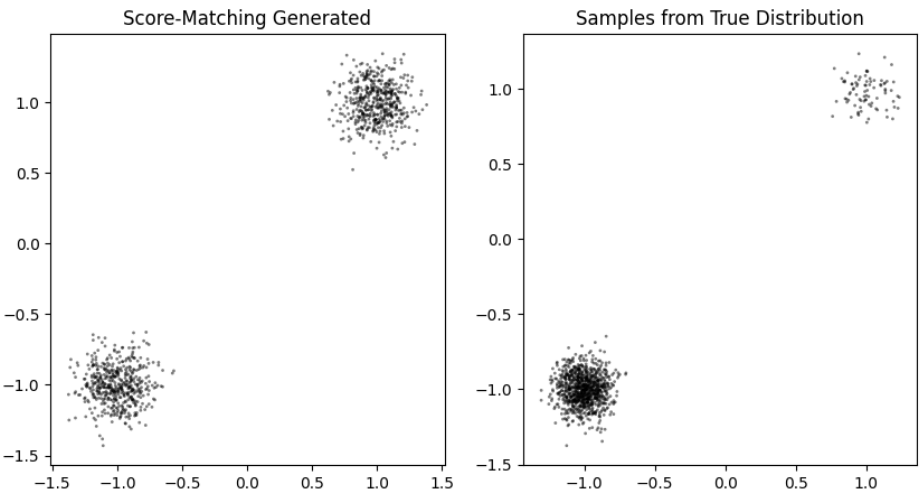

Figure 4: Sampling using Langevin dynamics on a trained score-matcher vs. from the true distribution in the case of an imbalanced GMM.

Once given an NCSN, the process of generating samples from the distribution p_{data} is similar to the process proposed in regular Langevin Dynamics. The sampling begins with the largest noise scale σ_1 and a large step size $\epsilon_1 = \epsilon \cdot \sigma_1^2/\sigma_T^2$. After conducting Langevin Dynamics with q_{σ_1} and ϵ_1, the output is used as input for the next stage of Annealed Langevin Dynamics, using σ_2 and $\epsilon_2 = \epsilon \cdot \sigma_2^2/\sigma_T^2$. This process is repeated K times, each step of Annealed Langevin Dynamics using the output of the previous step as its input and using step size $\epsilon_t = \epsilon \cdot \sigma_t^2/\sigma_T^2$.

The goal of separating the Langevin Dynamics into a process with several step sizes and perturbations is to solve the two problems that rise from Langevin Dynamics which were addressed in the beginning of this section. The low-density region issue is addressed in the first iterations, where the distributions are greatly perturbed using high values of σ_t. This causes the entire domain to have a more evenly divided distribution. For large enough values of K, this will cause more smooth sampling, as shown in Section 6.2. The slow mixing is addressed when there are enough noise scales, as shown in the Github repository provided.

4 Diffusion Models

Diffusion models [13] are a family of generative models which operate in two trajectories, the *forward process* and the *reverse process*. In general, diffusion processes are Markov processes capable of gradually transforming the distribution of an input to another distribution. The goal of a generative diffusion model is to convert a complex input distribution into a simple and tractable distribution and then learn a model which can successfully learn to revert the input back to its original representation.

A popular method for changing the distribution is by gradually adding small Gaussian noise to the input until it is almost unrecognizable. Continuing with previous notations, denote $\mathbf{x}_0 \sim p_{\text{data}}$ some point distributed based on the original data distribution. Sohl-Dickstein et al. [13] propose a process given *diffusion rates*, or variance schedules $\beta_1, \beta_2, \dots \beta_T$, which modify $\mathbf{x}_{1\dots T}$ as follows:

$$\mathbf{x}_t \mid \mathbf{x}_{t-1} \sim p_{\cdot|\mathbf{x}_{t-1}}, \qquad p_{\cdot|\mathbf{x}_{t-1}} := \mathcal{N}\left(\sqrt{1-\beta_t}\ \mathbf{x}_{t-1}; \beta_t \mathbf{I}\right) \tag{4.1}$$

such that the joint probability distribution of some trajectory $\mathbf{x}_{0\dots T}$ is:

$$p(\mathbf{x}_{0,\dots,T}) = p_{\text{data}}(\mathbf{x}_0) \prod_{t=1}^{T} p_{\cdot|\mathbf{x}_{t-1}}(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{4.2}$$

This process is assumed to be long enough such that $p_T(\mathbf{x}_T)$ (the marginal distribution of the final point in the diffusion trajectory) will be a standard Gaussian $\mathcal{N}(0, \mathbf{I})$. Afterwards, a network is trained to predict $q_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$, essentially denoising $\mathbf{x}_{t+1}$ to achieve an image closer to the input distribution, such that the prior distribution $q_\theta(\mathbf{x}_T) \sim \mathcal{N}(0, \mathbf{I})$. The objective of the function approximating q_θ is to minimize the log-likelihood of the marginal probability $q_\theta(\mathbf{x}_0)$ for $\mathbf{x}_0 \sim p_{\text{data}}$:

$$\theta^* = \arg\min_\theta L, \quad L := \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}}\left[-\log q_\theta(\mathbf{x}_0)\right] \tag{4.3}$$

It is possible to develop the expectation in L to achieve the following lower bound, known as the *variational lower bound* from the identities shown earlier:

$$L \geq \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}}\left[\underbrace{D_{\text{KL}}\left(p_{\text{data}}(\mathbf{x}_T|\mathbf{x}_0) \,||\, q_\theta(\mathbf{x}_T)\right)}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\text{KL}}\left(p_{\text{data}}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \,||\, q_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)}_{L_{t-1}} - \underbrace{\log q_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right] \tag{4.4}$$

4.1 Denoising Diffusion Probabilistic Models

A significant advancement in the field of diffusion models was Ho et al. [3]'s DDPMs – denoising diffusion probabilistic models; expanding the ideas shown above and providing a new training function. The variance schedule $\beta_1, \ldots, \beta_T$ is regarded as a set of static hyperparameters, making it nonlearnable and making the forward process remain the same throughout training. For this reason, L_T is also nonreliant on any learnable parameters, making it irrelevant to the loss function K. Another important observation is that the conditional distribution $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is a Gaussian distribution with mean and variance $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$, $\tilde{\Sigma}_t$ which are defined as:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\alpha_{t-1})}{1-\alpha_t}\mathbf{x}_t \tag{4.5}$$

$$\tilde{\Sigma}_t := \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t \mathbf{I} \tag{4.6}$$

where $\alpha_t := \prod_{i=1}^{t}(1-\beta_t)$.

The key difference between the general diffusion model method and the DDPM method is the difference between their loss functions and how the forward and backward distributions are parametrized. While no specific method was proposed by Sohl-Dickstein et al. [13], Ho et al. [3] propose modeling the reverse distribution as a Gaussian distribution $q_\theta(\mathbf{x}_t) \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}\right)$. In this model, all that must be learned to know $q_\theta(\mathbf{x}_t)$ is the mean $\mu_\theta(\mathbf{x}_t, t)$. Knowing that both reverse distributions $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $q_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are normal, there is a closed form to the expression in L_{t-1} defined in Equation 4.4, which can be modeled as:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}, \mathbf{x}_t \sim p(\cdot|\mathbf{x}_0)}\left[\frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2\right] + C \tag{4.7}$$

for some constant C which doesn't depend on θ, our only learnable parameters. Note that given $\mathbf{x}_0$, from properties of the Gaussian distribution, $\mathbf{x}_t|\mathbf{x}_0 \sim \mathcal{N}\left(\sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I}\right)$. Hence, using reparametrization one can replace $\mathbf{x}_t$ in the expectation with the equivalent expression: $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, meaning $\mathbf{x}_t$ can be viewed as a function of $\mathbf{x}_0$ and $\mathbf{z}$. They continue to develop the loss function and reach the following equality:

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(0,\mathbf{I})}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \mathbf{z}) - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{z}\right) - \mu_\theta(\mathbf{x}_t, t)\right\|^2\right] \tag{4.8}$$

meaning that μ_θ is attempting to predict the function $\frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{z}\right)$. Because the values $\mathbf{x}_t, \alpha_t, \beta_t$ are known to the model, the only thing that the model really needs to predict is $\mathbf{z}$. Hence, they define a new model $\mathbf{z}_\theta$ such that:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\right) \tag{4.9}$$

from here, it is possible to reach the following objective function:

$$L_{t-1} \equiv \mathbb{E}_{\mathbf{x}_0\sim p_{\text{data}}, \mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}\left\|\mathbf{z} - \mathbf{z}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{z}, t)\right\|^2\right] \tag{4.10}$$

A similar derivation can be made for L_1:

$$L_1 - D = \mathbb{E}_{\mathbf{x}_0\sim p_{\text{data}}, \mathbf{x}_1\sim p(\cdot|\mathbf{x}_0)}\left[\frac{1}{\sigma_1^2}\left\|\mathbf{x}_0 - \mu_\theta(\mathbf{x}_1, 1)\right\|^2\right] \tag{4.11}$$

They then decided to remove the weights in the expectations, changing the loss function. They train their model on the reweighted loss function:

$$L_{\text{simple}} = \mathbb{E}_{t\sim\text{U}(1,T)}\mathbb{E}_{\mathbf{x}_0\sim p_{\text{data}}, \mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\mathbf{z} - \mathbf{z}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{z}, t)\right\|^2\right] \tag{4.12}$$

In future sections, we will compare the two ideas and show how Denoising Diffusion Probabilistic Models can be modeled in the same form as a score-matching problem, and how we can use knowledge of SDEs to find better methods for generating new samples from a base distribution p_{data}.

5 A Unified Framework [15]

Given the theoretical basis as well as code provided in the submission, we now continue to the unified framework, showing how DDPMs and annealed Score Matching Langevin Dynamics (SMLD) are actually equivalent in training. Through the lens of stochastic differential equations (SDEs), we can provide a general framework to create generative models.

5.1 A Generalized Loss Function

As mentioned by Ho et al. [3], there is a similarity between the loss function L_{denoise} in Equation 3.11 of SMLD and L_{simple} in Equation 4.12 of DDPM. What this means is the models $\mathbf{s}_\theta$ and $\mathbf{z}_\theta$ are essentially approximating "equivalent" functions.

$$\begin{aligned}
L_{\text{simple}} &= \sum_{t=1}^{T}\frac{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}{\beta_t^2}L_{t-1} \\
&= \sum_{t=1}^{T}\frac{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}{\beta_t^2}\mathbb{E}_{\mathbf{x}_0\sim p_{\text{data}}, \mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}\left\|\mathbf{z} - \mathbf{z}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{z}, t)\right\|^2\right] \\
&= \sum_{t=1}^{T}\mathbb{E}_{\mathbf{x}_0\sim p_{\text{data}}, \mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\left\|\mathbf{z} - \mathbf{z}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{z}, t)\right\|^2\right].
\end{aligned}$$

For all t, define the reweighted $L_{t-1}, \tilde{L}_{t-1} := \frac{2\sigma_t^2(1-\beta_t)(1-\alpha_t)}{\beta_t^2} L_{t-1}$:

$$\begin{aligned}
\tilde{L}_t &= \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}, \mathbf{z} \sim \mathcal{N}(0,\mathbf{I})} \left[\left\| \mathbf{z} - \mathbf{z}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{z}, t) \right\|^2 \right] \\
&= \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}} \mathbb{E}_{\tilde{\mathbf{x}}_t \sim \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_0, \sqrt{1-\alpha_t}\mathbf{I})} \left[\left\| \frac{\tilde{\mathbf{x}}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1-\alpha_t}} - \mathbf{z}_\theta(\tilde{\mathbf{x}}_t, t) \right\|^2 \right] \\
&= \mathbb{E}_{\mathbf{x}_0} \mathbb{E}_{\tilde{\mathbf{x}}_t \sim \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_0, \sqrt{1-\alpha_t}\mathbf{I})} \left[\left\| -\sqrt{1-\alpha_t} \nabla_{\mathbf{x}_0} \log p(\tilde{\mathbf{x}}_t | \mathbf{x}_0) - \mathbf{z}_\theta(\tilde{\mathbf{x}}_t, t) \right\|^2 \right] \\
&= (1-\alpha_t) \mathbb{E}_{\mathbf{x}_0} \mathbb{E}_{\tilde{\mathbf{x}}_t \sim \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_0, \sqrt{1-\alpha_t}\mathbf{I})} \left[\left\| -\nabla_{\mathbf{x}_0} \log p(\tilde{\mathbf{x}}_t | \mathbf{x}_0) - \frac{1}{\sqrt{1-\alpha_t}} \mathbf{z}_\theta(\tilde{\mathbf{x}}_t, t) \right\|^2 \right],
\end{aligned}$$

If we define $\mathbf{s}_\theta(\tilde{\mathbf{x}}_t, t) = \frac{-1}{\sqrt{1-\alpha_t}} \mathbf{z}_\theta$, $\hat{\sigma}_t = \sqrt{1-\alpha_t}$, we get:

$$\tilde{L}_t(\theta, \beta) = \hat{\sigma}_t^2 \mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}, \tilde{\mathbf{x}}_t \sim q_{\hat{\sigma}_t}(\cdot|\mathbf{x}_0)} \left[\left\| \mathbf{s}_\theta(\tilde{x}_t, t) - \nabla_{\mathbf{x}} \log q_{\hat{\sigma}_t}(\tilde{\mathbf{x}}_t | \mathbf{x}_0) \right\|^2 \right] = \lambda(\hat{\sigma}_t) \ell_{\text{denoise}}(\theta; \hat{\sigma}_t), \tag{5.1}$$

for $\lambda(\sigma) = \sigma^2$, which happens to be the exact function proposed in [14]. This also implies that the network $\mathbf{z}_\theta$ minimizing $\tilde{L}_t$ can, after being scaled by an appropriate scalar, also minimize the function $\lambda(\hat{\sigma}_t)\ell_{\text{denoise}}(\theta, \hat{\sigma}_t)$. For this optimal network:

$$L_{\text{simple}}(\theta, \{\beta_i\}_{i=1}^T) = \sum_{t=1}^T \tilde{L}_t(\theta, \beta) = \sum_{t=1}^T \lambda(\hat{\sigma}_t) \ell_{\text{denoise}}(\theta, \hat{\sigma}_t) = L_{\text{denoise}}(\theta, \{\sigma_i\}_{i=1}^T). \tag{5.2}$$

The final loss function that will be used in the general framework will be as follows, after showing the equivalence of L_{simple} and L_{denoise}:

$$L_{\text{general}} = \mathbb{E}_{t \sim \text{Uni}[0,T]} \left\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0) \sim p_{\text{data}}} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \left[\left\| \mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t) | \mathbf{x}(0)) \right\|_2^2 \right] \right\} \tag{5.3}$$

5.2 Stochastic Differential Equations

Both SMLD and DDPM are models that begin with data that is complete noise $\mathbf{x}_T$ that denoise the data until it reaches $\mathbf{x}_0 \sim p_{\text{data}}$. In other words, we are given a prior distribution $\mathbf{x}_T \sim p_T$ and a true data distribution p_{data}, gradually transforming one distribution into the other in both directions. We can model this as a solution to the following SDE:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}, \tag{5.4}$$

which has a solution under Lipschitz conditions [8]. For input dimension d, $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is called the deterministic drift and $g : \mathbb{R} \to \mathbb{R}^{d \times d}$ is the stochastic diffusion based on Brownian motion. The noising scheme in Annealed Langevin Dynamics begins with data that follows the distribution and adds noise with increasing variance until the original image is almost completely gone. Using similar notation as Section 3.1 for $\{\sigma_t\}_{t=1}^T$, flipping the order such that $\sigma_1 < \cdots < \sigma_T$, we know that $q_{\sigma_t}(\mathbf{x}_t | \mathbf{x}_0)$ induces the same distribution for $\mathbf{x}_t$ given $\mathbf{x}_0$ as the following formula:

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \mathbf{z}_t,$$

where $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$. Similarly, we can develop that $\mathbf{x}_t | \mathbf{x}_{t-1}$ corresponds to the following formula, denoting $\sigma_0 = 0$ for simplicity:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}\mathbf{z}_t$$

For $T \to \infty$, by creating time steps $\Delta t = \frac{1}{T}$, the process $\mathbf{x}_{1,\ldots,T}$ becomes $\mathbf{x}(\frac{t}{T}) = \mathbf{x}_t$ and $\sigma(\frac{t}{T}) = \sigma_t$.

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \sqrt{\sigma^2(t + \Delta t) - \sigma^2(t)}\, \mathbf{z}(t) \approx \mathbf{x}(t) + \sqrt{\frac{d\sigma^2(t)}{dt} \Delta t}\, \mathbf{z}(t).$$

As $\Delta t \to 0$, the above equation shows that the noising kernel in SMLD behaves as the following SDE:

$$d\mathbf{x}(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}\, d\mathbf{w}$$

This outcome shows how given a noise function $\sigma(t)$, Annealed Langevin Dynamics are actually a discretization of the given model. DDPM uses a different scheme to add noise to the data, based on the parameters $\beta_1, \dots \beta_T$. It is possible using reparametrization, to rewrite the formula in Equation 4.1 as follows:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathbf{z}_t$$

Defining $\bar{\beta}_t = T\beta_t, \Delta t = \frac{1}{T}$:

$$\mathbf{x}_t = \sqrt{1-\frac{\bar{\beta}_t}{T}}\,\mathbf{x}_{t-1} + \sqrt{\frac{\bar{\beta}_t}{T}}\,\mathbf{z}_t = \sqrt{1-\bar{\beta}_t\Delta t}\,\mathbf{x}_t + \sqrt{\bar{\beta}_t\Delta t}\mathbf{z}_t$$

For large T, we can again view $\mathbf{x}_t$ as an approximation of a continuous Markov chain with $t \in [0,1], \beta(\frac{t}{T}) = \bar{\beta}_t, \mathbf{x}(\frac{t}{T}) = \mathbf{x}_t, \mathbf{z}(\frac{t}{T}) = \mathbf{z}_t$. Using the Taylor expansion of $\sqrt{1-x}$, converting β into a continuous function using linear interpolation:

$$\begin{aligned}
\mathbf{x}(t+\Delta t) &= \sqrt{1-\beta(t+\Delta t)\Delta t}\,\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\,\mathbf{z}(t) \\
&= \left(1 - \frac{1}{2}\beta(t+\Delta t)\Delta t + o(\Delta t)\right)\mathbf{x}(t) + \sqrt{\beta(t+\Delta t)\Delta t}\,\mathbf{z}(t) \\
&\approx \mathbf{x}(t) - \frac{1}{2}\beta(t)\Delta t\mathbf{x}(t) + \sqrt{\beta(t)\Delta t}\,\mathbf{z}(t)
\end{aligned}$$

As $\Delta t \to 0$, the above equation shows that the noising kernel in DDPM behaves as the following SDE:

$$d\mathbf{x}(t) = -\frac{1}{2}\beta(t)\mathbf{x}(t)dt + \sqrt{\beta(t)}d\mathbf{w}, \tag{5.5}$$

which is very similar to Equation 3.8 when the score function is of a standard normal distribution and β is constant. Using an expansion of the Fokker-Planck equation [12], we can compute the conditional distributions in the continuous variations of SMLD and DDPM:

$$\mathbf{x}(t) \mid \mathbf{x}(0) \sim \begin{cases} \mathcal{N}(\mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I} & \text{(SMLD)} \\ \mathcal{N}(e^{-\frac{1}{2}\int_0^t \beta(s)\,ds}\mathbf{x}(0), (1 - e^{-\int_0^t \beta(s)\,ds})\mathbf{I}) & \text{(DDPM)} \end{cases}$$

Which after many iterations, converges to the following prior distributions:

$$\mathbf{x}(T) \dot{\sim} p_T \equiv \begin{cases} \mathcal{N}(0, \sigma_T^2\mathbf{I}) & \text{(SMLD)} \\ \mathcal{N}(0, \mathbf{I}) & \text{(DDPM)} \end{cases}$$

5.3 Sampling

Modeling the processes in both models as a stochastic differential equation has a strong advantage. Anderson [1] showed that the reverse process of an SDE as shown in Equation 5.4 can also be modeled as an SDE:

$$d\mathbf{x} = \left[f(\mathbf{x}_t, t) - g(t)^2\nabla_{\mathbf{x}_t}\log\rho_t(\mathbf{x}_t)\right]dt + g(t)d\hat{\mathbf{w}}, \tag{5.6}$$

where the process is moving backwards in time, and $d\hat{\mathbf{w}}$ is reverse Brownian motion. ρ_t is the marginal distribution of the forward process, such that the distribution of $\mathbf{x}_t$ that develops according to the SDE will be equal to ρ_t. Note that the network $\mathbf{s}_\theta(\mathbf{x}, t)$ is exactly what approximates $\nabla_{\mathbf{x}}\log\rho_t(\mathbf{x})$ for all time points $t \in [0, T]$.

While it is possible to simply use a numerical sampler such as the Euler-Maruyama method [6], having a network that can predict the score at all time points is significantly beneficial and grants the opportunity to use additional information provided by the network.

Reverse Diffusion Using a very similar discretization method to the discretization DDPM provides to Equation 5.5, the following method was proposed for sampling from the reverse-time SDE:

$$\mathbf{x}_t = \mathbf{x}_{t+1} - f(\mathbf{x}_{t+1}, t+1) + g(t+1)^2 \mathbf{s}_\theta(\mathbf{x}_{t+1}, t+1) + g(t+1)\mathbf{z}_{t+1}, \tag{5.7}$$

where $\mathbf{z}_{t+1}$ is a standard Normal random variable. This method can be used as a possible *predictor* in the predictor-corrector sampling method.

Predictor-Corrector Sampling It is possible to combine the sampling method provided by Ho et al. [3] in DDPM and the sampling method provided by Song and Ermon [14] in SMLD for an iterative method that samples from the target distribution p_{data}. It consists of a *predictor* and a *corrector*, where the former is responsible for simulating the backwards progress of the SDE. After a single step of the predictor, it applies the latter, which essentially performs Langevin Dynamics to approximate ρ_t for all time steps t. The combined approach, as opposed to only using a predictor, is an attempt to make sure that a random error in any stage can't cause the sampler to sample something completely out of distribution and for there to be more stability in the generation.

Probability Flow Sampling A faster method than the method proposed by Song and Ermon [14] is to completely remove the intermediate sampling in the generation loop, only sampling $\mathbf{x}_T \sim p_T$. Using the Fokker-Planck equation, it is possible to prove that the following reverse ODE ($\mathbf{x}_T \sim p_T$) induces the same marginal distribution as the marginal distribution induced by Equation 5.6:

$$d\mathbf{x}(t) = \left(f(\mathbf{x}(t), t) - \frac{1}{2}\nabla \cdot (g(\mathbf{x}(t), t) g(\mathbf{x}(t), t)^T) - \frac{1}{2} g(\mathbf{x}(t), t) g(\mathbf{x}(t), t)^T \mathbf{s}_\theta(\mathbf{x}(t), t) \right) dt$$

The above ODE provides an invertible mapping from each point in the space to a new point that will ideally be distributed according to p_{data}. This is a large benefit as the generation process is less noisy and slightly more explainable. Additionally it makes interpolation possible – visualizing how different points on a line will be generated.

6 Results

6.1 Code Implementation

The implementation of the three methods proposed above (SMLD, DDPM and SDEs) is available at our Github repository[1]. All experiments were written in Python 3.10, using PyTorch and run on Kaggle's T4 GPU. All methods utilized U-Net architectures for prediction of the score (in the case of SMLD and SDE) / noise (in the case of DDPM). The networks were trained on MNIST due to limited computation resources, however the code can be easily adjusted for different datasets such as CIFAR-10 and more. The optimizer used to minimize the optimization function was Adam. Main hyperparameters used for each method are provided in Table 1.

For more specifics about the implementation, all can be found in the Github repository.

6.2 Visual Outputs

[1] https://github.com/ben-eliav/Diffusion-Models-Project

Value	SMLD	DDPM	SDE
T	10	1000	1000
σ_0 or β_0	0.01	0.0001	0.0001
σ_T or β_T	1	0.02	0.02
# training epochs	40	40	45
Learning rate	0.001	0.0001	0.0001
Sampling method	Langevin Dynamics	Ancestral Sampling	Ancestral Sampling
Time embedding method	Through normalization	Torch embedding + linear	Gaussian Fourier

Table 1: Hyperparameters used for each model.

Visualizing the diffusion process in the generation of images using each method:

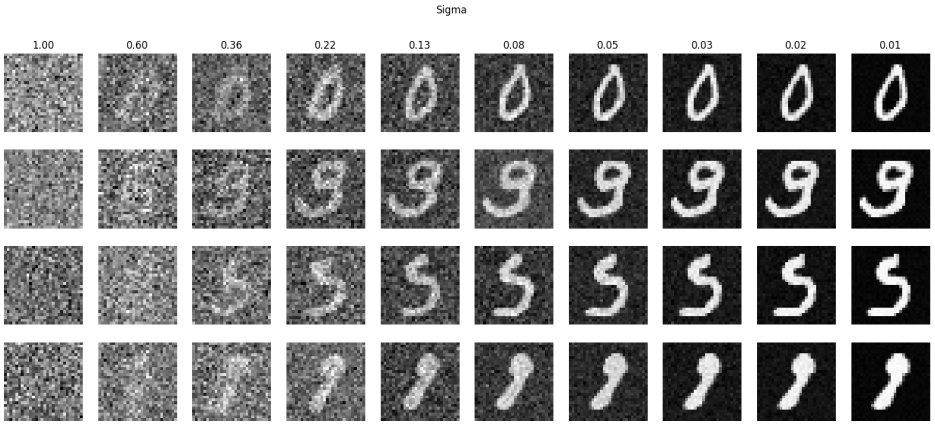

Figure 5: Diffusion Process in SMLD

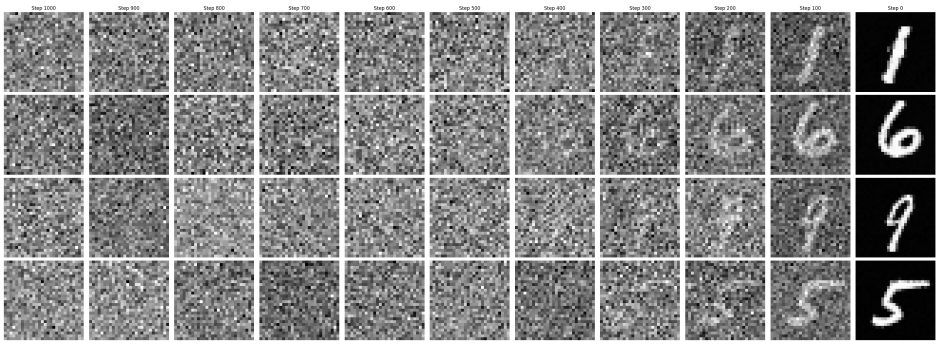

Figure 6: Diffusion Process in DDPM

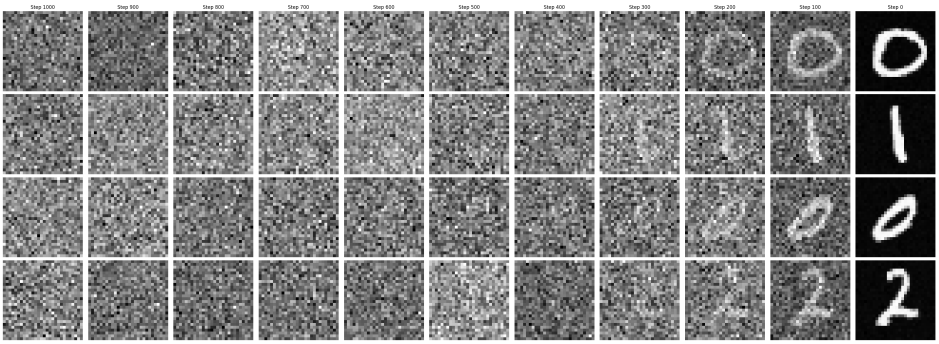

Figure 7: Diffusion Process in Unified Framework (SDE)

To see more examples of generated images, see Appendix A. While performance metrics such as Inception Score [11] and Frechét Inception Distance [2] are available for generative models, unfortunately we don't have the capability of generating a sufficient number of samples to measure these scores, due to computational limitations. Hence we resort to visually assessing the generated samples by all methods, which we will discuss in the next section.

7 Conclusions

In this paper, we surveyed two prominent methods for image generation: Score Based Langevin Dynamics [14] and Diffusion Models [13, 3]. Subsequently, we introduced a combined method [15], integrating both methods. From a qualitative visual assessment, it appears that DDPM generates the highest-fidelity images as demonstrated in Appendix A. An interesting observation in the generation process is while Figure 5 shows gradual denoising, Figures 6 and 7 show that the images appear to be complete noise until the process is about 70% finished, meaning the generation process could be potentially shortened. This is based solely on visual assessment and requires further testing to confirm its validity.

Our experiments were conducted on MNIST, as limited resources did not allow for training on more complex datasets, for instance RGB images. However, our implementation and results are a basis, showing that the theory works in practice and providing a general proof of concept for these types of models. Additionally, given more time and computational resources, we would have experimented with different forms of sampling in the third method (Reverse Diffusion, Probability Flow) as well as conditional sampling.

While the work was mainly theoretical, providing proofs for several theorems which were not sufficiently proven in their respective papers, we also provided a novel codebase for each of the methods based solely on the pseudocode and algorithms provided in the papers, using U-Net architectures available in different open-source projects credited in the Github repository. Some of the papers already have open-source implementations but our version is much more minimalistic and simple to run than those provided by the authors of the papers, being flexible enough to change datasets and methods with ease. The only python packages required are PyTorch and NumPy, as opposed to other implementations using different Python modules.

Our research consists of several limitations. The prominent weakness of our approach is our inability to compare the outputs of our models to other generative models such as GANs and VAE. Another limitation is not being able to provide an accurate log-likelihood of the sampled images as provided in methods such as Continuous Normalized Flows [5]. These are issues that could likely be addressed in future work with more available resources as well as new methods combining diffusion models and generative flows such as those proposed by Lipman et al. [5].

References

[1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[4] Paul Langevin et al. Sur la théorie du mouvement brownien. *CR Acad. Sci. Paris*, 146(530-533):530, 1908.

[5] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[6] Gisiro Maruyama. Continuous markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4:48–90, 1955.

[7] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[8] Brent Øksendal. *Stochastic differential equations*. Springer, 2003.

[9] VM Planck. Über einen satz der statistischen dynamik und seine erweiterung in der quantentheorie. *Sitzungberichte der*, 1917.

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[11] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[12] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[13] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[14] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[15] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[16] Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. *Advances in neural information processing systems*, 32, 2019.

[17] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[18] Li Wenliang, Danica J Sutherland, Heiko Strathmann, and Arthur Gretton. Learning deep kernels for exponential family densities. In *International Conference on Machine Learning*, pages 6737–6746. PMLR, 2019.

[19] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

A More Outputs

Figure 8: 64 randomly sampled SMLD images

Figure 9: 64 randomly sampled DDPM images

Figure 10: 64 randomly sampled SDE images