Denoising score matching for diffusion models

Rachida Saroui
ENS Paris-Saclay
rachidasaroui@gmail.com

Samar Rabeh
ENS Paris-Saclay
samar.rabeh@ens-paris-saclay.fr

Ilias Rami
ENS Paris-Saclay
ilias.rami@ens-paris-saclay.fr

1. Introduction

Generative modeling has become a cornerstone in artificial intelligence research, enabling applications in computer vision, natural language processing, and scientific simulations. From simple Gaussian distributions to advanced deep generative models, the field has evolved to address challenges like intractable normalizing constants and complex data distributions. A promising paradigm in this domain is Score-Based Generative Modeling (SBGM), a technique leveraging score functions to overcome traditional limitations. In this work, we explain **score matching** 4 and **Langevin dynamics** 6, and discuss their connection to **diffusion models** 7. Our main contributions are twofold: (1) applying sampling techniques to the MNIST dataset, and (2) analyzing the impact of hyperparameters, such as the noise scale (σ) and step size, on the sampling process.

2. Estimating the probability distribution of data

Generative models seek to estimate the probability distribution of data. Given a dataset $(x_1, \ldots, x_N)$, it is assumed that an unknown data distribution $p(x)$ represent these observations. By approximating $p(x)$ with a model distribution, it becomes possible to sample novel data points or compute probabilities for new observations. However, real-world data distributions, particularly for high-dimensional data such as images or audio, tend to exhibit significant complexity, necessitating correspondingly complex model distributions.

While simpler distributions, such as Gaussian distributions, are mathematically tractable, proved inadequate for modeling intricate data structures. In contrast, neural networks possess the capability to represent complex probability distributions. A neural network maps inputs x to outputs $f_\theta(x)$, which can be converted into a valid probability distribution using the exponential function to ensure positivity and a normalizing constant Z_θ for normalization.

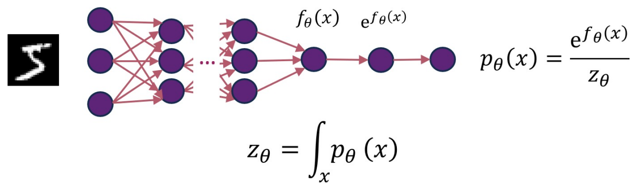

$f_\theta(x)$
$e^{f_\theta(x)}$
$p_\theta(x) = \frac{e^{f_\theta(x)}}{z_\theta}$
$z_\theta = \int_x p_\theta(x)$

Figure 1. Illustration of the annealed score matching process.

The normalizing constant Z_θ involves integrating the exponential of f_θ across all possible values of x. This integration is computationally feasible for straightforward cases, such as Gaussian distributions, yet becomes intractable for more complex neural networks. Even in the case of discrete x, where the integral simplifies to a summation, computing Z_θ is P-complete, making it at least as hard as NP-complete problems. This intractability represents a fundamental challenge in deep generative modeling.

3. Tackling The Intractable Normalizing Constant

The intractability of the normalizing constant in deep generative modeling constitutes a significant challenge, which is addressed through three primary approaches in the literature. Approximation methods, such as variational inference [3] and energy-based-models [4] , facilitate computational feasibility but compromise the accuracy of probability estimates due to their reliance on approximations. Alternatively, restricting neural network architectures [5] can ensure tractable normalizing constants, yet this approach inherently constrains model flexibility and expressive power. Generative adversarial networks (GANs)[1] circumvent this issue by directly modeling data generation; however, their non-probabilistic nature limits the ability to perform explicit probability evaluations.

These limitations underscore the necessity for a framework that achieves a balance between tractability, flexibility, and probabilistic accuracy. Accurate probability computations are essential for applications such as outlier detection, model comparison, and lossless compression, while controlled and high-quality data generation is critical for fields like medical imaging.

[6] proposes a novel framework that integrates flexible neural architectures, precise probabilistic modeling, and controllable data generation. This approach addresses existing limitations and enhances the capabilities of generative modeling for both theoretical and practical applications.

4. Working With Score Matching

The score function plays a critical role in understanding the data distribution. It is defined as:

$$\nabla_x \log p_{\text{data}}(x) = \nabla_x \log e^{f_\theta(x)}(x) - \nabla_x \log Z_\theta$$

where Z_θ represents the normalizing constant. Since this constant is independent of x, its gradient equals zero, permitting us to omit it from consideration. This simplification proves advantageous; once the score is computed, the probability density function can be estimated by integrating the score function with respect to x. Thus, the transition between the probability density function and the score function does not incur any loss of information.

In this context, we presuppose the existence of a set of independent and identically distributed (i.i.d.) data samples $(x_1, x_2, x_3, \ldots, x_N)$, drawn from the data distribution $p_{\text{data}}(x)$. Our goal is to estimate the score function of the true data distribution using a score model $s_\theta(x)$, which functions as a mapping from $\mathbb{R}^d$ input to $\mathbb{R}^d$ output.

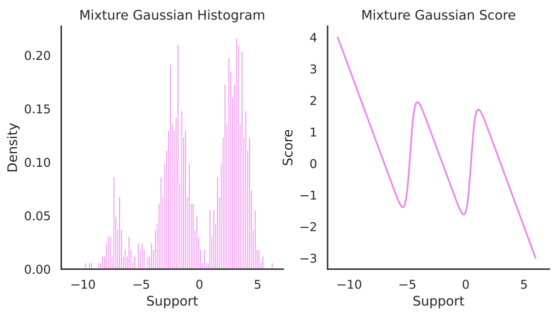


Figure 2. Score function of a mixture of Gaussians

To facilitate this process, we require an objective function that enables the comparison of two vector fields. This comparison is typically executed using the Fisher divergence, defined as follows:

$$\mathbb{E}_{p_{\text{data}}}\left[|\nabla_x \log p_{\text{data}}(x) - s_\theta(x)|_2^2\right]$$

This expression denotes the average Euclidean distance between the score model and the true score function. However, direct computation of the Fisher divergence is unfeasible as it necessitates knowledge of the true score function. To circumvent this challenge, [6] uses Gauss theorem to demonstrates that the Fisher divergence is equivalent to the subsequent objective:

$$\mathbb{E}_{p_{\text{data}}}\left[|s_\theta(x)|_2^2 + \text{trace}\left(\nabla_x s_\theta(x)\right)\right]$$

We can substitute the expected value with an empirical mean over the training data, yielding the following expression:

$$\frac{1}{N}\sum_{i=1}^{N}\left[|s_\theta(x_i)|_2^2 + \text{trace}\left(\nabla_x s_\theta(x_i)\right)\right]$$

Although this formulation is computationally manageable, it presents certain computational drawbacks. Specifically, the calculation of the trace of the Jacobian necessitates as many backpropagation steps as there are dimensions in the input, which may be inefficient for high-dimensional data.

To mitigate this issue, a more scalable variant of the Fisher divergence, referred to as the sliced Fisher divergence, has been proposed. The fundamental insight underlying this approach is that two vector fields that are close in the original space should also exhibit proximity when projected onto a random direction. This leads to the following expression:

$$\frac{1}{2}\mathbb{E}_{p_V}\mathbb{E}_{p_{\text{data}}(x)}\left[\left(v^T\nabla_x \log p_{\text{data}}(x) - v^T s_\theta(x)\right)^2\right]$$

where v represents a randomly selected projection direction, and p_V denotes the distribution over these projection directions. By utilizing Gauss's theorem, we can demonstrate that this expression is equivalent to minimizing:

$$\mathbb{E}_{p_V}\mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\left(v^T\nabla_x s_\theta(x)v\right)^2 + \frac{1}{2}\left(v^T s_\theta(x)\right)^2\right]$$

This formulation significantly enhances computational efficiency, as the computation of the Jacobian vector product necessitates only a single backpropagation step, irrespective of the input dimensionality.

5. Denoising Score Matching

Denoising score matching addresses the computational challenges of traditional score matching by introducing noise into data points, enabling easier computation of the Jacobian term. A perturbation kernel, $q_\sigma(\tilde{x})$, is used, where $\tilde{x}$ is the noisy data point and x is the original, noise-free data point. Typically modeled as a Gaussian distribution with mean x and standard deviation σ, this kernel generates a noisy data distribution.

The goal is to estimate the score function for the noisy distribution, rather than the original data distribution. As σ approaches zero, the noisy distribution converges to the original one. The denoising score matching objective is:

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x})}\left|\nabla_{\tilde{x}}\log q_\sigma(\tilde{x}) - s_\theta(\tilde{x})\right|^2$$

Alternatively, it can be reformulated as:

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q_\sigma(\tilde{x}|x)}\left|\nabla_{\tilde{x}}\log q_\sigma(\tilde{x}|x) - s_\theta(\tilde{x})\right|^2$$

This formulation is computationally manageable since the gradient of the perturbation kernel is tractable, and the score function $s_\theta(\tilde{x})$ can be computed through a forward pass.

However, denoising score matching has limitations. By introducing noise, it cannot directly estimate the scores of noise-free distributions. Additionally, reducing noise increases the variance of the objective, which can lead to instability.

6. Score-based generative modeling

Suppose our dataset consists of i.i.d. samples $\{x_i\}_{i=1}^N \subset \mathbb{R}^D$ drawn from an unknown data distribution $p_{\text{data}}(x)$. The score of a probability density $p(x)$ is defined as $\nabla_x \log p(x)$. A score network $s_\theta : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network parameterized by θ, which is trained to approximate the score of $p_{\text{data}}(x)$. The objective of generative modeling is to utilize the dataset to learn a model capable of generating new samples from $p_{\text{data}}(x)$. The framework of score-based generative modeling consists of two key components: score matching and Langevin dynamics.

6.1. Sampling with Langevin dynamics

6.1.1. MetropolisAdjusted LangevinAlgorithm

Langevin dynamics can produce samples from a probability density $p(x)$ using only the score function $\nabla_x \log p(x)$. Given a fixed step size $\varepsilon > 0$, and an initial value $\tilde{x}_0 \sim \pi(x)$ with π being a prior distribution, the Langevin method [6]recursively computes the following:

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\varepsilon}{2}\nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\varepsilon}z_t,$$

6.2. Challenges

6.2.1. Data lying on manifolds

The manifold hypothesis posits that real-world data often lie on low-dimensional manifolds embedded within high-dimensional spaces, known as the ambient space. While this assumption underpins manifold learning, it introduces significant challenges for score-based generative models. Specifically, the score $\nabla_x \log p_{\text{data}}(x)$, which represents the gradient of the data distribution, becomes undefined when data are restricted to a low-dimensional manifold. Additionally, the score matching objective is only valid when the data distribution has full support in the ambient space; otherwise, it becomes inconsistent.

6.2.2. Low density regions

The challenge of low data density regions leads to two significant issues: inaccurate score estimation in score matching and slow mixing in Langevin dynamics. In score matching, the scarcity of samples in low-density regions prevents accurate estimation of the score function since there is insufficient evidence to minimize the expected squared error. As a result, score estimates are reliable only near the high-density modes of the data distribution. In Langevin dynamics, when two modes are separated by low-density regions, the sampling process struggles to mix efficiently. Gradients of the log-density, which guide sampling, fail to encode the relative weights of the modes, causing the algorithm to converge slowly or inaccurately. Experimental results (Figure 4 & Figure 6) confirm this issue, showing that Langevin dynamics produces samples with incorrect relative densities between modes in a Gaussian mixture, consistent with theoretical predictions.

6.3. Solutions

To address these challenges, [6] proposes perturbing the data with random Gaussian noise of varying magnitudes. Adding random noise ensures that the resulting distribution does not collapse to a low-dimensional manifold. High noise levels produce samples in low-density regions of the original (unperturbed) data distribution, thereby improving the accuracy of score estimation. Importantly, a single score network is trained to be conditioned on the noise level, enabling the estimation of scores across all noise magnitudes. The authors further introduce an annealed version of Langevin dynamics, which begins by utilizing scores corresponding to the highest noise level and gradually reduces the noise level until it is small enough to be nearly indistinguishable from the original data distribution.

6.4. Noise Conditional Score Networks

Let $\{\sigma_i\}_{i=1}^L$ be a positive geometric sequence that satisfies $\sigma_1 > \sigma_2 > \cdots > \sigma_L > 1$. According to [6], the solution to the challenges above is to define :

$$q(x) = \int p_{\text{data}}(t)\mathcal{N}(x \mid t, 2I)dt$$

, which represents the perturbed data distribution. The noise levels $\{\sigma_i\}_{i=1}^L$ are chosen such that σ_1 is large enough to mitigate the difficulties discussed before, and σ_L is small enough to minimize the effect on data.

The objective is to train a conditional score network to jointly estimate the scores of all perturbed data distributions, i.e., $\{\sigma_i\}_{i=1}^L : s_\theta(x, \sigma_i) \approx \nabla_x \log q_\sigma(x)$. Note that $s_\theta(x, \sigma_i) \in \mathbb{R}^D$ when $x \in \mathbb{R}^D$. This network is referred to as a Noise Conditional Score Network (NCSN). The Denoising Score Matching objective is then adapted to optimize over all noise levels:

$$J_{\text{NCSN}}(\theta) \triangleq \frac{1}{L} \sum_{i=1}^{L} \mathbb{E}_{q_{\sigma_i}(\tilde{x}, x)} \left\| s_\theta(\tilde{x}, \sigma_i) + \frac{1}{\sigma_i^2}(\tilde{x} - x) \right\|_2^2$$

In practice, the authors have introduced a term correcting the magnitude orders, yielding the following objective:

$$J_{\text{NCSN}}(\theta) \triangleq \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \mathbb{E}_{q_{\sigma_i}(\tilde{x}, x)} \left\| s_\theta(\tilde{x}, \sigma_i) + \frac{\tilde{x} - x}{\sigma_i^2} \right\|_2^2$$

where $\lambda(\sigma_i) = \sigma_i^2$. The authors have empirically observed that this correction term allows for better results and improves the convergence of the algorithm.

6.5. Annealed Langevin dynamics

After training the NCSN (Noise Conditional Score Network), we can use it for sampling through the already-introduced Langevin dynamics. The only requirement is that we must utilize the scores learned across various noise levels, progressing from the initial high noise to the final low noise. To achieve this, we employ the following algorithm, referred to as *Annealed Langevin Dynamics* (ALD), inspired by the Annealed Importance Sampling (AIS) technique.

The heuristic behind this approach is as follows: in the initial steps, the noise allows the generated sample to move across the noise-expanded manifold, pushing it toward higher-density regions. This ensures that, regardless of the starting point, the sample can be transported to any relevant portion of the manifold. Gradually, as the noise decreases, the sample is guided closer to the true data manifold. Ultimately, the final sample $\tilde{x}_T$ will be drawn from the last distribution, where the noise is nearly zero. Consequently, $\tilde{x}_T$ is expected to be a valid sample lying on the dataset manifold.

7. Connection between Diffusion Models and Score Matching

The connection between diffusion models and score matching lies in their shared goal: learning the underlying structure of a data distribution. Although these approaches originate from different frameworks, they align through their reliance on gradient-based loss functions and noise-perturbed data. This section provides a detailed mathematical explanation.

Algorithm 1: Annealed Langevin Dynamics

```
Input($\{\sigma_i\}_{i=1}^L$, $\epsilon$, $T$) Initialize $\tilde{x}_0$;
for $i \leftarrow 1$ to $L$ do
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ;      // $\alpha_i$ is the step
      size
    for $t \leftarrow 1$ to $T$ do
        Draw $z_t \sim \mathcal{N}(0, I)$;
        $\tilde{x}_t \leftarrow \tilde{x}_{t-1} + \frac{\alpha_i}{2} s_\theta(\tilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} z_t$;
    end
    $\tilde{x}_0 \leftarrow \tilde{x}_T$;
end
Output($\tilde{x}_T$)
```

7.1. Diffusion Models: Adding and Removing Noise

Diffusion models [2] generate data by gradually adding noise (forward process) and learning to reverse this process (reverse process).

Forward Process. Starting with clean data $x_0 \sim p_{\text{data}}(x)$, Gaussian noise is added at each timestep t:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} \cdot x_{t-1}, \beta_t I),$$

where β_t is the variance schedule. This forms a Markov chain of increasingly noisy samples $x_1, x_2, \ldots, x_T$, eventually reducing the data to pure noise $x_T \sim \mathcal{N}(0, I)$.

Reverse Process. To generate samples, the reverse process starts with $x_T \sim \mathcal{N}(0, I)$ and iteratively denoises using a learned model $p_\theta(x_{t-1}|x_t)$:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)),$$

where μ_θ and Σ_θ are learned parameters. This process progressively reconstructs data from noise.

7.2. Connecting Diffusion and Score Matching

Noise Injection and Score Matching. Diffusion models resolve score matching challenges in low-density regions by injecting noise into the data, creating a perturbed distribution $q_\sigma(x_t|x_0)$. The perturbed data distribution is:

$$q_\sigma(x_t) = \int q_\sigma(x_t|x_0) p_{\text{data}}(x_0)\, dx_0.$$

Instead of directly estimating $\nabla_x \log p_{\text{data}}(x)$, diffusion models estimate $\nabla_x \log q_\sigma(x_t)$ at different noise levels σ, using a neural network $s_\theta(x, t)$. The loss function is:

$$L_{\text{DSM}}(\theta) = \mathbb{E}_{q_\sigma(x_t, x_0)} \left[\|s_\theta(x_t, t) - \nabla_x \log q_\sigma(x_t|x_0)\|^2 \right].$$

Reverse Process as Langevin Dynamics. The reverse process in diffusion models uses the estimated scores to guide denoising via Langevin dynamics:

$$x_{t-1} = x_t + \frac{\epsilon}{2}\nabla_x \log q_\sigma(x_t) + \sqrt{\epsilon} \cdot z_t,$$

where $z_t \sim \mathcal{N}(0, I)$ and ϵ is a small step size. Iterating this process across decreasing noise levels generates realistic samples.

7.3. Diffusion Models as a Generalization of Score Matching

Diffusion models generalize score matching by framing the forward process as a stochastic differential equation (SDE) [7]:

$$dx = f(x, t)\, dt + g(t)\, dW_t,$$

where W_t is Brownian motion, and the reverse process solves:

$$dx = \left[f(x, t) - g(t)^2 \nabla_x \log p_t(x)\right]\, dt + g(t)\, dW_t.$$

Here, $\nabla_x \log p_t(x)$ is learned using $s_\theta(x, t)$, the score of the noisy data.

Training. The training objective minimizes:

$$L^*_{\text{DSM}} = \mathbb{E}_t\Big[\lambda(t)\mathbb{E}_{p(x_0)}\mathbb{E}_{p_t(x_t|x_0)}\Big[\|s_\theta(x_t, t) - \nabla_x \log p_t(x_t|x_0)\|^2\Big]\Big]. \tag{1}$$

where $\lambda(t)$ is a weighting function, and $t \sim U([0, T])$. By framing the problem in terms of SDEs, diffusion models effectively combine the robustness of score matching with the flexibility of noise-perturbed generative modeling.

7.4. Denoising Diffusion Probabilistic Models (DDPMs)

DDPMs extend the concepts of score matching and diffusion by framing the forward and reverse processes in a variational framework:

Variational Lower Bound. The training objective optimizes the variational lower bound on the data likelihood:

$$\log p_\theta(x_0) \geq \mathbb{E}_q\left[\log \frac{p_\theta(x_0, z)}{q(z|x_0)}\right], \tag{2}$$

where $z = (x_1, \ldots, x_T)$ is the latent variable.

Simplifying this results in the noise prediction loss:

$$L(\theta) \propto \sum_{t>1} \frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\alpha_t)}\mathbb{E}_{x_0,\epsilon}\Big[\|\epsilon - \epsilon_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon, t)\|^2\Big]. \tag{3}$$

Here, ϵ_θ predicts the noise ϵ injected at each timestep.

Relation to Score Matching. The gradient of the noisy log density, $\nabla_{x_t} \log q(x_t|x_0)$, relates directly to ϵ via:

$$\nabla_{x_t} \log q(x_t|x_0) = -\epsilon.$$

Substituting this into the loss shows that DDPMs indirectly optimize a score-matching-like objective.

Key Differences. - DDPMs use timestep-dependent noise injection, while Noise Conditional Score Networks (NCSNs) rely on predefined noise levels. - DDPMs require a large number of timesteps to approximate the true posterior.

Score matching focuses on learning the gradient of the data distribution. Diffusion models extend this by learning scores for multiple noise levels and using them to generate data progressively. The connection lies in the shared loss function: both optimize the closeness of the estimated score to the true score of noisy data. This alignment makes diffusion models a natural generalization of score matching.
In summary, diffusion models turn score matching into a practical, scalable generative framework by leveraging noise and continuous dynamics, allowing them to generate high-quality samples in high-dimensional spaces.

8. EXPERIMENTS AND RESULTS

We implemented the training pipeline for noise conditional denoising networks based on their repository on GitHub and conducted a comprehensive analysis of their performance across various tasks, including image generation and inpainting. Additionally, we examined how the selection of different hyperparameters affects performance.

8.1. Model

The NoiseConditionalScoreNetwork utilized in [6] is specifically designed for tasks such as image generation, denoising, and interpolation. It employs a variation of RefineNet that integrates conditional instance normalization to accommodate the noise scale, denoted as σ, during score prediction. This architectural framework is predicated on the CondRefineNetDilated, which systematically processes input images alongside their corresponding noise levels to enhance the quality of the generated output.

8.2. Image Generation

In our image generation task using the score-matching approach with RefineNet on the MNIST dataset, we observed that the model did not fully converge, leading to poor generation. Although the loss converged, the model tended to collapse to a single mode of the data distribution, resulting in a lack of variability in the generated samples, as shown in Figure .

To address this, we increased the value of σ_1, a technique introduced in [6]. This adjustment facilitated smoother transitions between regions of the data distribution that are connected by low-density areas. As shown in Figures 2 and 9, increasing σ_1 led to more diverse samples. However, this also introduced more noise into the generated images, and the model still failed to fully converge, resulting in poor image generation.

8.2.1. Hyperparameter Tuning:

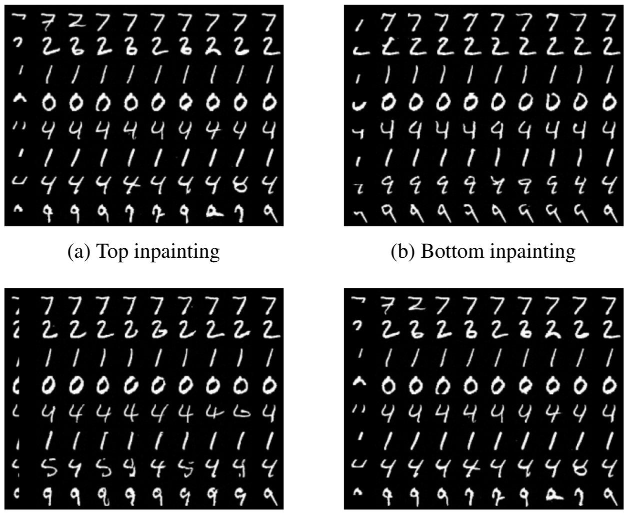
Effect of n_{step}: The Figure 8 demonstrates the effect of varying the n_{step} parameter on the quality of generated samples, with $\sigma = 1$ and $\epsilon = 0.00005$ held constant. As n_{step} increases, the generated images become clearer and more recognizable. In the case of 200 steps (a), the digits are well-defined, indicating the model has refined the samples effectively. With fewer steps, such as 50 (b), the images are noisier and less distinct, while 100 steps (c) strike a balance between clarity and noise. At 150 steps (d), the results are clearer than with 50 steps but slightly less refined than with 200, suggesting that increasing n_{step} improves image quality up to a certain point, beyond which the improvement is marginal.

Effect of ϵ: The Figure 9 compares results for varying ϵ values with fixed $\sigma = 1$ and $n_{\text{step}} = 200$, showing a progressive improvement in clarity as ϵ increases. At $\epsilon = 0.000001$ (a), the images are highly noisy with little discernible structure. Increasing to $\epsilon = 0.000005$ (b) reduces noise slightly, allowing faint structures to emerge. At $\epsilon = 0.00001$ (c), clearer shapes form with significantly less noise, and finally, at $\epsilon = 0.0001$ (d), the images are the sharpest, with well-defined digits and minimal noise. This trend indicates that larger ϵ values improve structure and clarity, with $\epsilon = 0.0001$ providing the best visual quality in this comparison.

Effect of σ: The Figure 10 compares results for $\sigma = 10$ and $\sigma = 1$ with fixed $\epsilon = 0.00005$ and $n_{\text{step}} = 200$, highlighting the impact of σ on the output quality. In panel (a) with $\sigma = 10$, the images exhibit significant noise, especially in the earlier iterations, with the digits becoming visible but less sharp. In contrast, panel (b) with $\sigma = 1$ produces clearer and more defined shapes, with significantly reduced noise and improved structure. This comparison demonstrates that a smaller σ value leads to better visual clarity and sharper results under the same ϵ and step conditions.

8.3. Inpainting Experiment

In this work, we explore the effectiveness of Langevin dynamics for image inpainting on the MNIST dataset. Inspired by [1], where a modified Langevin dynamics framework was used to inpaint one part of an image based on the other part. We test the approach to systematically inpaint in all different directions: top, bottom, left, and right. For each experiment, the algorithm begins by occluding a specific region of the image while preserving the remaining part. At each iteration, Langevin dynamics is applied to generate the missing region. The process continues iteratively until the final noise level. At this step the algorithm produces a coherent image combining the original and generated parts. We utilize the pretrained checkpoints from [1] for our experiments. Our results demonstrate that the inpainting quality is consistent and can produce diverse plausible completions. In an other hand, and as expected, we notice that as the occluded region always contains a part of the object and not the whole key object of interest, the reconstruction is significantly well achieved , This is due to the generative nature of the approach. In some cases where the


(a) Top inpainting (b) Bottom inpainting

(c) Left inpainting (d) Right inpainting

Figure 3. Inpaintings from top, bottom, left, and right of 8 images taken from MNIST. Occluded image in the far left and original image in the far right.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and

Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014. 1

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 4

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014. 1

[4] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 1(0):191–246, 2006. 1

[5] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015. 1

[6] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. 2, 3, 5, 6

[7] 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. 5

A. Appendix

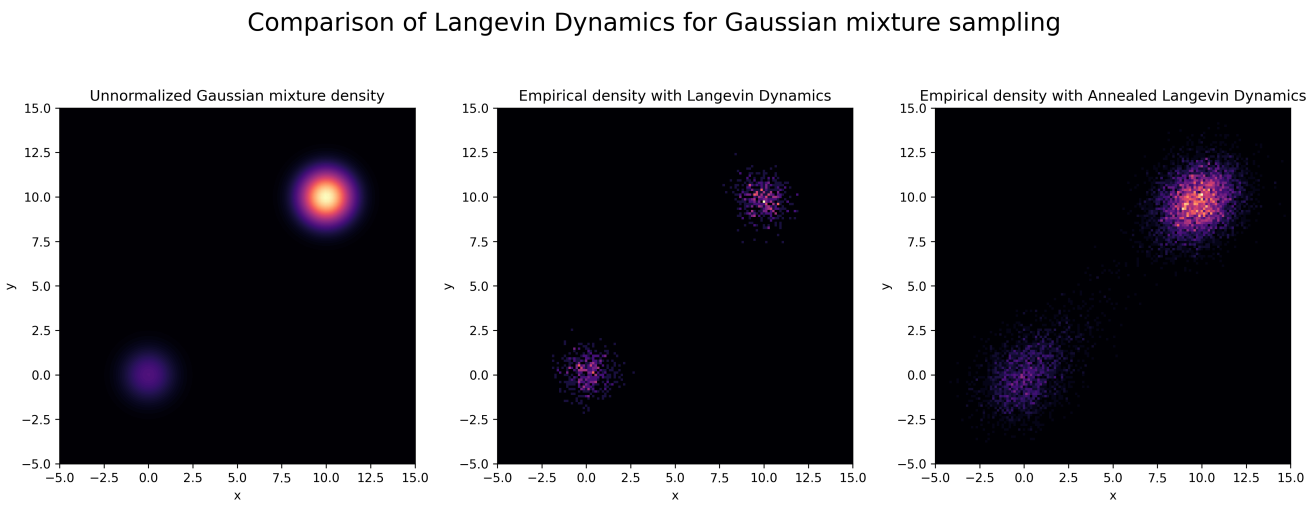
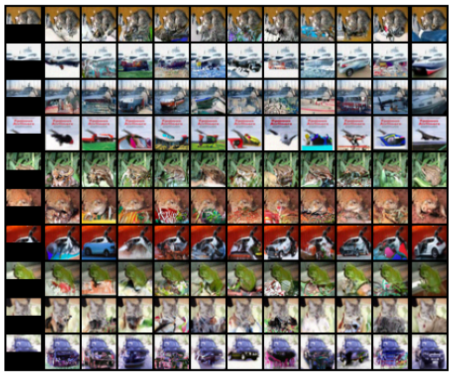
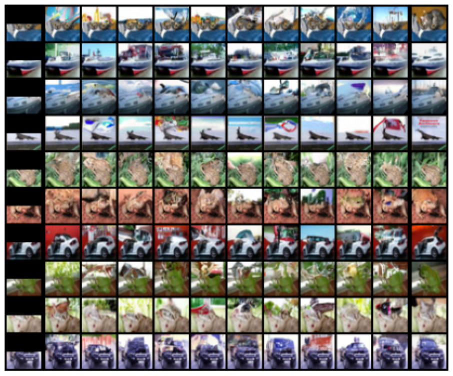


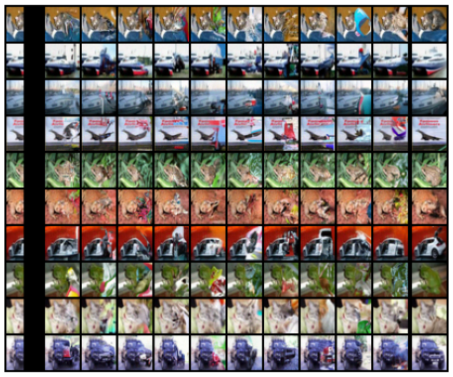
Figure 4. Comparaison between langevin and annealed langevin



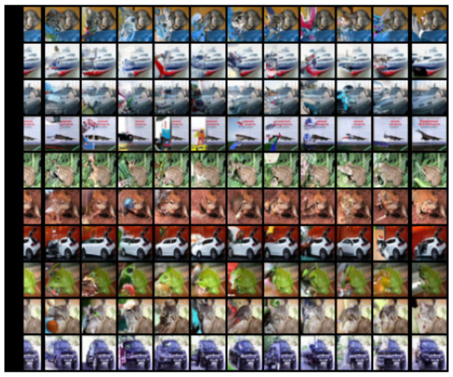
(a) Top inpainting



(b) Bottom inpainting



(c) Left inpainting



(d) Right inpainting

Figure 5. Inpaintings from top, bottom, left, and right of 8 images taken from Cifar10. Occluded image in the far left and original image in the far right.

This experiment demonstrates how adding increasing levels of Gaussian noise to a mixture of three Gaussians affects the

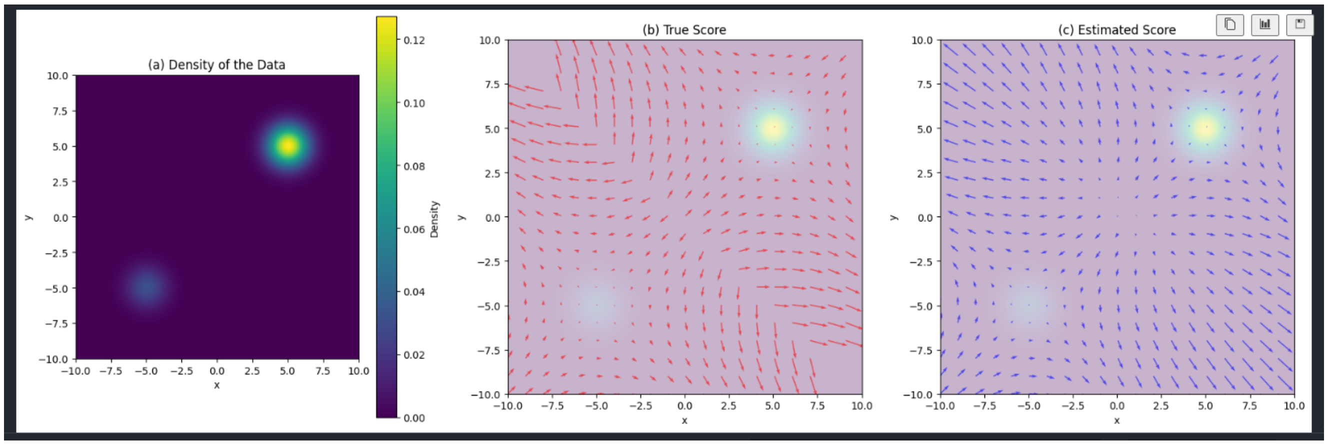


Figure 6. Illustration of the score matching process

data distribution, visualized through Empirical Cumulative Distribution Functions (ECDFs). The results show that as noise increases (σ from 1 to 8), the distinct steps in the ECDF (representing the original mixture components) gradually smooth out, illustrating how noise fills the low-density regions between mixture components but also corrupts the original data structure.

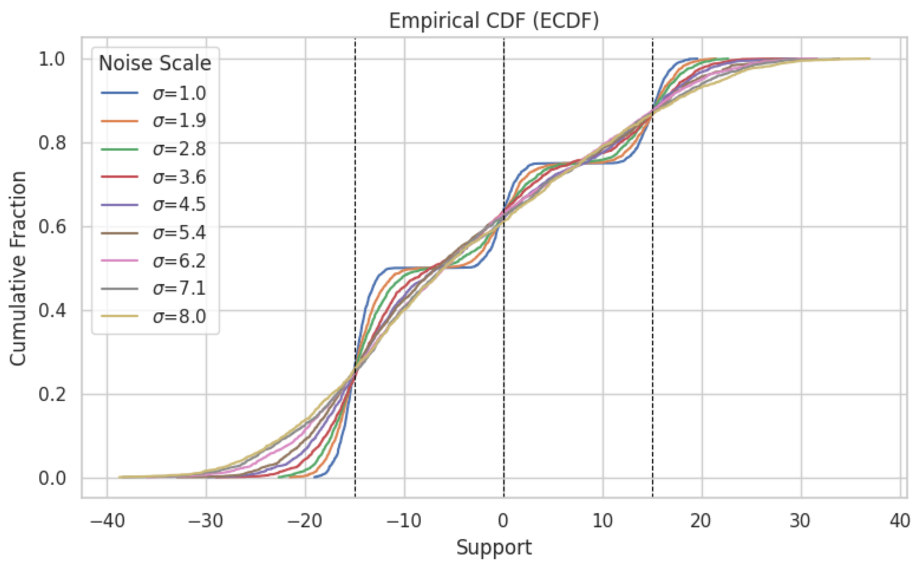
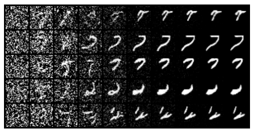
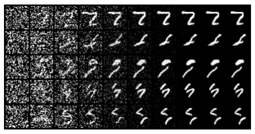
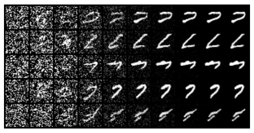


Figure 7. Noise Sclaes


(a) $\sigma = 1, \epsilon = 0.00005, n_{\text{step}} = 200$


(b) $\sigma = 1, \epsilon = 0.00005, n_{\text{step}} = 50$


(c) $\sigma = 1, \epsilon = 0.00005, n_{\text{step}} = 100$

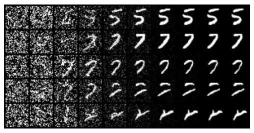
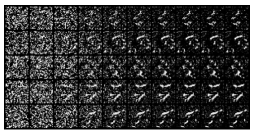
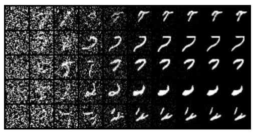
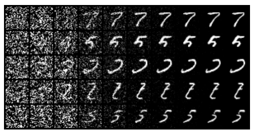

(d) $\sigma = 1, \epsilon = 0.00005, n_{\text{step}} = 150$

Figure 8. Comparison of results for different parameters $\sigma = 1$, $\epsilon = 0.00005$ and varying n_{step}.


(a) $\sigma = 1, \epsilon = 0.000001, n_{\text{step}} = 200$


(b) $\sigma = 1, \epsilon = 0.00005, n_{\text{step}} = 200$


(c) $\sigma = 1, \epsilon = 0.00001, n_{\text{step}} = 200$

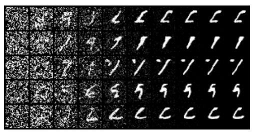
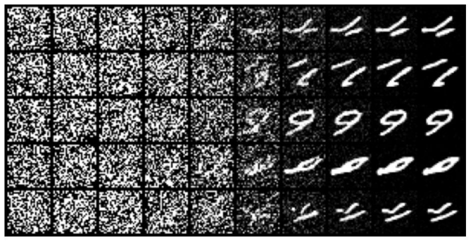

(d) $\sigma = 1, \epsilon = 0.0001, n_{\text{step}} = 200$

Figure 9. Comparison of results for varying ϵ with $\sigma = 1$ and $n_{\text{step}} = 200$.


(a) $\sigma = 10, \epsilon = 0.00005, n_{\text{step}} = 200$

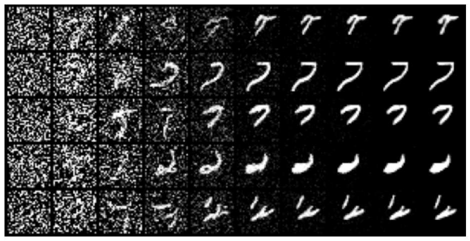

(b) $\sigma = 1, \epsilon = 0.00005, n_{\text{step}} = 200$

Figure 10. Comparison of results for $\sigma = 10$ and $\sigma = 1$ with $\epsilon = 0.00005$ and $n_{\text{step}} = 200$.