From Denoising Score Matching to Denoising Diffusion Probabilistic Models

Philippe Gratias-Quiquandon
ENS Paris-Saclay
pgratias-22@telecom-paris.fr

Enguerrand Paquin
ENS Paris-Saclay
enguerrand.paquin@telecom-paris.fr

Noé Vernier
ENS Paris-Saclay
noe.vernier@telecom-paris.fr

Abstract

This report explores the connection between Denoising Score Matching (DSM) and Denoising Diffusion Probabilistic Models (DDPM). We begin by examining how Langevin Dynamics can generate samples from an estimated score, but highlight challenges arising from inaccessible true densities and data concentrating on low-dimensional manifolds. DSM addresses these issues by adding noise to data, making score estimation feasible throughout the input space. Noise Conditional Score Networks (NCSNs) further refine this approach by training a model on multiple noise levels, enabling annealed Langevin sampling that gradually removes noise. Finally, we show that DDPMs, which reverse a forward diffusion process, can be viewed as a special case of DSM on a noise schedule, providing a unified perspective on recent advancements in score-based generative modeling.

The first part from 1 to 2.2 was written by Philippe Gratias-Quiquandon. The second part from 2.2 to 3 was written by Noé Vernier and the last part from 3 to the end was written by Enguerrand Paquin.

1 Sampling data with Langevin Dynamics

1.1 Intuition behind the equation

One is interested in a method that could generate data from a dataset. Starting from the score function $\nabla_x \log(p(\mathbf{x}))$, a well-known method to create new samples is the Langevin Dynamics Song and Ermon [2020] which is defined as below :

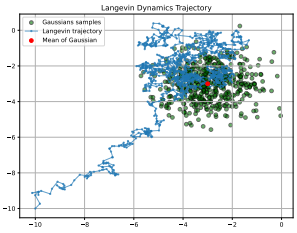
$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\varepsilon}{2}\nabla_{\mathbf{x}} \log(p(\tilde{\mathbf{x}}_{t-1})) + \sqrt{\varepsilon}\mathbf{z}_t$$

where $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$, $\varepsilon > 0$ is a fixed step size and $\tilde{\mathbf{x}}_0$ is given by a prior π. It can be shown that the density of $\tilde{\mathbf{x}}_T$ goes to $p(\mathbf{x})$ as $T \to \infty$ and $\varepsilon \to 0$. Let us understand why does it work and what are the conditions for using it. First of all, what is happening when $T \to \infty$ and $\varepsilon \to 0$? When $\varepsilon \to 0$, we can see ε as an infinitesimal time, therefore rewrite the equation as a differential stochastic equation :

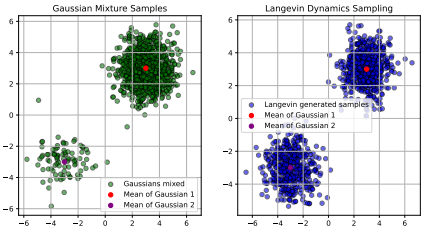
$$\tilde{x}_t - \tilde{x}_{t-1} = d\tilde{x}_{t-1} = \frac{1}{2}\nabla_{\mathbf{x}} \log(p(\tilde{\mathbf{x}}_{t-1}))dt + dN_{t-1}$$

where N_t is a Brownian motion. The utility of N_t and the noise we introduced is to assure the convergence w.r.t T : when $T \to \infty$, it permits the $\tilde{x}_t$ to explore the whole support of $p(x)$ and avoid the "fixation" of the equation on an equilibrium point. Indeed, if we had only the $\nabla_{\mathbf{x}} \log(p(\tilde{\mathbf{x}}_t))$ term, we would attract the $\tilde{x}_t$ to a minimum. As a reminder, we want the $\tilde{x}_t$ to be a generation of new data points following $p(x)$.

1.2 Limitations



(a) Trajectory of the Langevin dynamics, in blue is represented the values of $\tilde{x}_t$



(b) Comparison between data directly sampled from the density (left) and sampling with Langevin dynamics using ground truth score (right).

We implemented a simple experiment $1a$ to show what is the trajectory of the Langevin dynamics and more specifically, what are the challenges with this equation. In this example, we see that we need something like 150-250 steps to have the last $\tilde{x}_T$ to be considered as Gaussian sample. The number of steps we need depends on two parameters : the prior for $\tilde{x}_0$ and the step size ε we choose. Another limitation of classical Langevin dynamics is that if we have mixed Gaussians, for example, $p(x) \sim \mathcal{N}((-3,-3), I)$ with a probability α and $p(x) \sim \mathcal{N}((3,3), I)$ with a probability $1-\alpha$, we would have an imbalance between samples (α for a Gaussian and $1-\alpha$ for the other in proportion). However, if the two Gaussians have disjointed supports, the score function does not depend on α. Therefore, we observe that the results produced by the Langevin sampling is not consistent with the real density 1b. We can avoid this by using 1 that is written below.

2 Denoising Score Matching

2.1 Principal of Score Matching

In the previous section, we used the Langevin dynamics to generate new data but we assumed that we have access to $\nabla_x \log(p_{\text{data}}(x))$. In this section, we will explain the different methods to estimate it. The first idea was to write the density as a function of an energy, i.e, $p(\mathbf{x}, \theta) = \frac{\exp(-E(\mathbf{x},\theta))}{Z(\theta)}$. Here, $Z(\theta)$ is intractable as the data often lies in high-dimensional spaces and it is numerically impossible to integrate over all of them Vincent [2011]. We are looking for a way to estimate the score without having to compute $Z(\theta)$: as $\nabla_x \log(p(\mathbf{x}, \theta))$ does not depend on $Z(\theta)$, it is more convenient to estimate this value than directly the density. We will denote $s_\theta(\mathbf{x})$ as the estimated score from now on. Therefore we search the solution of the problem below :

$$\min_\theta \mathbb{E}_{p_{\text{data}}(x)} \left(\frac{1}{2} \| s_\theta(\mathbf{x}) - \nabla_x \log(p_{\text{data}}(\mathbf{x})) \|^2 \right) \text{ with } p_{\text{data}}(x) \text{ being the true density}$$

This approach is known as Explicit Score Matching (ESM). However, it still relies on knowing the true density $p_{\text{data}}(\mathbf{x})$, which makes defining the loss directly infeasible. To overcome this, we can reformulate the objective to remove the explicit dependence on the true density, leading to a method called Implicit Score Matching (ISM). The proof of this reformulation is provided in Section A.1.

$$\min_\theta \mathbb{E}_{p_{\text{data}}(x)} \left(\frac{1}{2} \| s_\theta(\mathbf{x}) \|^2 + \text{tr}(\nabla_x s_\theta(\mathbf{x})) \right)$$

In the proof though, we use assumptions that poses the limits of this matching. First of all, the term $\text{tr}(\nabla_x s_\theta(\mathbf{x}))$ can be really heavy to compute. But the main issue here is that we got this result by assuming that we have a $\mathbf{x}$ lying in the support of p_{data} where the density is not zero. In practice, it is rarely the case.

However, there is still another method to approximate $\text{tr}(\nabla_x s_\theta(\mathbf{x}))$ to avoid the computation of the matrix which can be high-dimensional. One needs to remark that $v^\top \nabla_x s_\theta(\mathbf{x}) v$ can be efficiently computed where v are random vectors. Indeed, the idea is to randomly project the matrix $\nabla_x s_\theta(\mathbf{x})$ with a multivariate standard normal. It is called the Sliced Score Matching (**SSM**) and it is formulated as below :

$$\min_\theta \mathbb{E}_{p_v} \mathbb{E}_{p_{\text{data}}} \left(v^\top \nabla_x s_\theta(\mathbf{x}) v + \frac{1}{2} \| s_\theta(\mathbf{x}) \|_2^2 \right)$$

With this matching, we circumvented the $\text{tr}(\nabla_x s_\theta(\mathbf{x}))$ computation but not the problem of the support of p_{data}. We did an experiment with MNIST by estimating the score of the true data using **SSM** and the issue of the low-manifold hypothesis (explained later in our report) appears in 2.

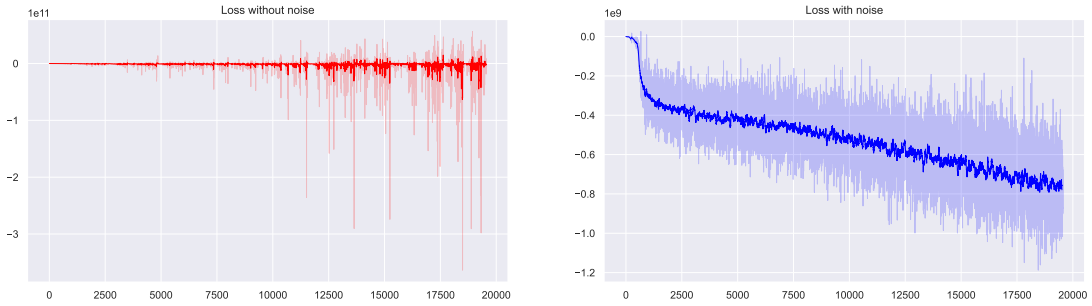


Figure 2: On the left, SSM loss during the iterations for the true data. On the right, the same loss but with a noise added to the data.

This phenomenon happens because, as the support of p_{data} is small compared to the whole space, the density is equal to zero in most of the space. Therefore, the score $\nabla_x \log(p(\mathbf{x}, \theta))$ is defined in a small manifold and outside of it, it can not be approximated by **SSM**. The solution to this is adding little Gaussian noise to the data in order to get a score function that is defined on the space. This is called the Denoising Score Matching (**DSM**):

$$\min_\theta \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})}\left(\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2\right)$$

where $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ is the pre-chosen noise that we add to the original data x.

2.2 Sampling, Manifold Hypothesis, and Low-Density Regions

Although Denoising Score Matching (DSM) provides a framework for training a model to approximate the score of a data distribution, the quality of samples generated via Langevin Dynamics remains unsatisfactory in practice. This phenomenon can be observed when sampling data such as MNIST, where generated samples often fail to resemble the original data distribution accurately. This can be attributed to regions of low data density, which themselves arise as a consequence of the *manifold hypothesis*.

Figure 3: Sampling MNIST with Langevin Dynamics

The *manifold hypothesis* posits that real data lie near a low-dimensional manifold $\mathcal{M}$ embedded in a high-dimensional space. As a result, the data density $p_{\text{data}}(\mathbf{x})$ is significant only near $\mathcal{M}$, while regions far from it have near-zero density. During DSM training, the model $s_\theta(\tilde{\mathbf{x}})$ is optimized to approximate $\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}})$ only in high-density regions. In low-density areas, where $p_{\text{data}}(\tilde{\mathbf{x}}) \approx 0$, the model is effectively unconstrained, leading to arbitrarily inaccurate score estimates that do not affect the loss. We show that the variance of the quantity estimated via Monte Carlo is very large when σ is small. As a result, in low-density regions with few samples, the estimator becomes highly inaccurate.

$$\sum_{i=1}^{d} \mathbb{V}_{\tilde{\mathbf{x}}|\mathbf{x}}\left(\nabla_{\tilde{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})_i\right) = d\sigma^{-2}, \quad \text{as } \sigma \to 0.$$

This issue becomes particularly problematic for Langevin dynamics sampling. Starting from $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$, the initial point will lie in low-density regions with high probability due to the curse of dimensionality. In these regions, where $s_\theta(\tilde{\mathbf{x}})$ is unreliable, the gradient updates are erratic, and the process struggles to move toward $\mathcal{M}$. Moreover, the probability of reaching $\mathcal{M}$ during sampling is exponentially small, given that high-density regions occupy a vanishingly small fraction of the space. Consequently, the combination of unconstrained training in low-density regions and poor initialization leads to poor sampling performance.

2.3 Noise Conditional Score Networks

To address the limitations imposed by the *manifold hypothesis* and improve the efficiency of Langevin sampling, the concept of Noise Conditional Score Networks (NCSNs) was introduced by Song and Ermon [2020]. The key idea is to perturb the data with various levels of Gaussian noise, making the data distribution more amenable to score-based generative modeling.

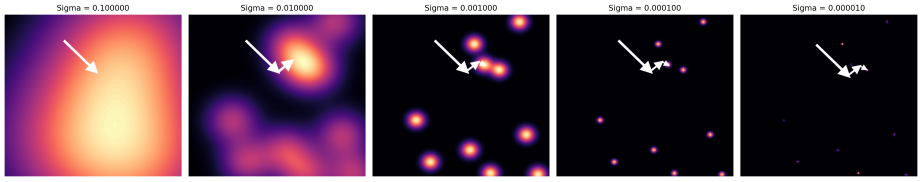


Figure 4: Noise Conditional Score Networks Sampling

Gaussian noise ensures that the perturbed data is no longer confined to a low-dimensional manifold, but when σ is too small, low-density regions are still problematic and when σ is too large, the data distribution is lost. To address this issue, the authors propose a schedule for σ that decreases over time, ensuring that the noise level is high initially to explore the space and gradually decreases to capture the data distribution accurately like illustrated in Figure 4.

2.3.1 Training : Noise Conditional Score Networks

Let $\{\sigma_i\}_{i=1}^L$ be a positive geometric sequence of noise levels such that $\sigma_1 > \sigma_2 > \cdots > \sigma_L$. For a given σ, the perturbed data distribution $q_\sigma(\mathbf{x})$ is defined as:

$$q_\sigma(\mathbf{x}) \triangleq \int p_{\text{data}}(\mathbf{t}) \mathcal{N}(\mathbf{x} \mid \mathbf{t}, \sigma^2 I)\, \mathrm{d}\mathbf{t}.$$

The goal is to train a conditional score network $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ to approximate the score of all perturbed distributions:

$$\mathbf{s}_\theta(\mathbf{x}, \sigma) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}), \quad \forall \sigma \in \{\sigma_i\}_{i=1}^L.$$

NCSNs are trained using a variant of Denoising Score Matching (DSM). For a given noise level σ, the DSM loss is:

$$\ell(\theta; \sigma) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 I)} \left[\left\| \mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} \right\|_2^2 \right].$$

The unified training objective aggregates this loss across all noise levels:

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^L) = \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \ell(\theta; \sigma_i),$$

where $\lambda(\sigma_i)$ is a weighting factor for each noise level.

2.3.2 Sampling : Noise Conditional Score Networks

The sampling process in Noise Conditional Score Networks (NCSNs) proposed by Song and Ermon [2020] uses an annealed version of Langevin dynamics to progressively generate high-quality samples from a noisy distribution. When sampling a point $\tilde{\mathbf{x}}_t$ for a given noise level σ_i, it ensures that the point avoids regions of low density, making it more likely to land in areas with higher density for the subsequent step with a lower noise level σ_{i+1}. By gradually reducing the noise during the process, the model guides the sample towards the true distribution, ensuring that the point transitions smoothly into regions of higher probability as the noise is annealed.

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L, \varepsilon, T$.
1: Initialize $\tilde{\mathbf{x}}_0$
2: **for** $i \leftarrow 1$ to L **do**
3: $\quad \alpha_i \leftarrow \varepsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ α_i is the step size.
4: $\quad$ **for** $t \leftarrow 1$ to T **do**
5: $\quad\quad$ Draw $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$
6: $\quad\quad \tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
7: $\quad$ **end for**
8: $\quad \tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
9: **end for**
return $\tilde{\mathbf{x}}_T$

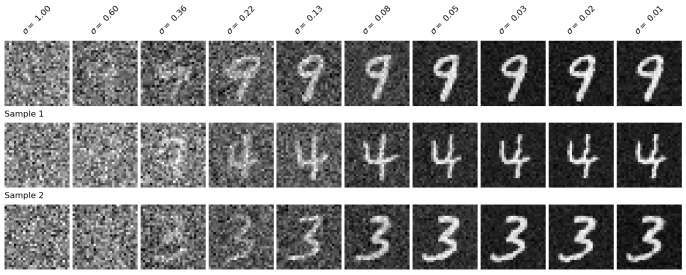


Figure 5: Samples generated by Noise Conditional Score Networks

2.3.3 Experiments : Noise Conditional Score Networks

For our experiments, we implemented a Noise Conditional Score Network (NCSN) using a UNet architecture with attention mechanisms to represent $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma)$. To simplify the training process, we conditioned the UNet directly on σ, avoiding the need to train separate models for each noise level. The UNet consists of 64 initial channels, with three levels of resolution where the number of channels doubles at each level. Attention mechanisms are applied at the highest resolution, and each level contains two residual blocks.

Our dataset is MNIST, consisting of grayscale images of size 28×28. We used $L = 10$ noise levels, with $\sigma_1 = 1$ and $\sigma_L = 0.01$, following a geometric sequence. The model was trained for 10,000 iterations using the Adam optimizer with a learning rate of 2×10^{-5}.

For sampling, we employed $T = 100$ steps per noise level and set $\varepsilon = 2 \times 10^{-5}$. Figure 5 will illustrate the progression of sampled images for three examples across the 10 noise levels, showing the transition from noisy data to high-quality samples resembling the original MNIST images.

3 Denoising Diffusion Probabilistic Models

In 2020, a new approach Ho et al. [2020] on Diffusion Probabilistic Models (DPMs) exhibits strong similarities between DPMs and DSM. We will first give a general introduction on DPMs and we will show in a second time that a specific parametrization of DPMs is equivalent to DSM.

3.1 Introduction to Diffusion Probabilistic Models

DPMs are generative models parametrized by a Markov chain and composed of a forward and a backward process. The forward process aims at adding Gaussian noise step by step to a data sample $\mathbf{x}_0$ while the backward process involves teaching a generative model to recover the original data sample from the noised version step by step.

Formally, the forward process can be described as follows :

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

where T is the number of steps of the process, $\mathbf{x}_0$ is the sample data and $\mathbf{x}_1 \dots \mathbf{x}_T$ are latent variables.

The addition of Gaussian noise at each step is described by :

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

where β_t is a constant that can be chosen to adjust the amount of noise added at each step and $\mathbf{I}$ is the identity matrix.

By exploiting reparametrization trick A.2, it is possible to express $\mathbf{x}_t$ simply :

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon \quad \text{where} \quad \bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i), \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

The DPM aims at learning a model $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T})d\mathbf{x}_{1:T}$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is known as the backward diffusion process and we define it as a Markov chain with learned Gaussian transitions :

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

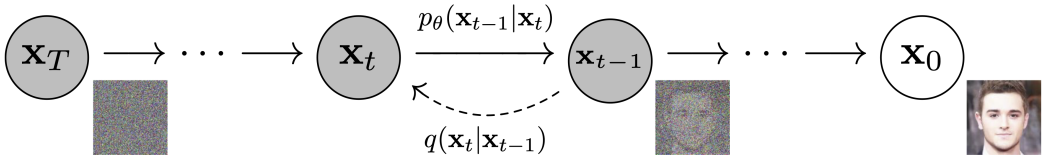


Figure 6: The markov chain of the forward and backward diffusion process Ho et al. [2020]

The goal is to learn how to get back to the real data distribution $q(\mathbf{x}_0)$ by taking Gaussian step starting from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. Each step can be described by :

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$

It can be shown A.3 that the true backward diffusion process is tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I})$$

where

$$\tilde{\mu}(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_t\Big) \quad \text{and} \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

We can see that only $\boldsymbol{\varepsilon}_t$ is unknown in the expression of $\tilde{\mu}(\mathbf{x}_t, t)$ so we can choose $\mu_\theta(\mathbf{x}_t, t)$ as follows:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\Big)$$

Where $\boldsymbol{\varepsilon}_\theta$ is a function approximator intended to predict $\boldsymbol{\varepsilon}$ from $\mathbf{x}_t$.

Usually, the β_t are fixed so $\tilde{\beta}_t$ is completely known and we can define:

$$\mathbf{\Sigma}_\theta(\mathbf{x}_t, t) = \tilde{\beta}_t \mathbf{I}$$

The mean $\mu_\theta(\mathbf{x}_t, t)$ can be found by solving the following optimization problem :

$$\min_\theta \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\varepsilon}_t}\Big[||\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)||^2\Big]$$

3.2 A link between Diffusion Probabilistic Models training objective and Denoising Score Matching

We have seen that in DSM, a score network is trained to estimate the gradient of the density probability of the noised training data.

$$s_\theta(\tilde{\mathbf{x}}, \sigma) \approx \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$$

with $\tilde{\mathbf{x}} = \mathbf{x} + \sigma\boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We can deduce that

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} = -\frac{\boldsymbol{\varepsilon}}{\sigma}$$

If we take diffusion annotation we have that

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$$

We also demonstrates in A.2 that

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

This leads to

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = \mathbb{E}_{q(\mathbf{x}_0)}\left[\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0)\right] = \mathbb{E}_{q(\mathbf{x}_0)}\left[-\frac{\boldsymbol{\varepsilon}_\theta}{\sqrt{1-\bar{\alpha}_t}}\right] = -\frac{\boldsymbol{\varepsilon}_\theta}{\sqrt{1-\bar{\alpha}_t}}$$

This shows that the NCSN model in DSM is also an approximation of a scaled version of the noise estimator in DPM.

3.3 A link between Diffusion Probabilistic Models sampling and Langevin dynamics

Once $\boldsymbol{\varepsilon}_\theta$ is trained, the sampling method for DPM is easy and consists of following the backward diffusion process. We start by $\mathbf{x}_T = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ and we progressively remove noise with :

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right) + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\boldsymbol{\varepsilon}$$

with $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Given that s_θ and $\boldsymbol{\varepsilon}_\theta$ are very similar, we can easily see that the sampling technique of DPM is really close to the Annealed Langevin dynamics algorithm. Only the scaling factors and the number of sampling step for each level of noise differ.

References

Brian Chao. Anime face dataset: a collection of high-quality anime faces., 2019. URL https://github.com/bchao1/Anime-Face-Dataset.

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

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33:12104–12114, 2020.

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.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. URL https://arxiv.org/abs/1907.05600.

Mitchell van Zuijlen, Hubert Lin, Kavita Bala, S.C. (Sylvia) Pont, and M.W.A. (Maarten) Wijntjes. Materials in paintings (mip): An interdisciplinary dataset for perception, art history, and computer vision, 2021. URL https://data.4tu.nl/articles/_/13679200/1.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23 (7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

A Mathematical demonstrations

A.1 Demonstration of the reformulation of the ESM to the ISM

$$\frac{1}{2}\|s_\theta(\mathbf{x}) - \nabla_x \log(p_{\text{data}}(\mathbf{x}))\|^2 = \frac{1}{2}\|s_\theta(\mathbf{x})\|^2 - \langle s_\theta(\mathbf{x}), \nabla_x \log(p_{\text{data}}(\mathbf{x}))\rangle + \frac{1}{2}\|\nabla_x \log(p_{\text{data}}(\mathbf{x}))\|^2$$

The last term does not depend on θ, thus we can ignore him from the problem.

$$-\mathbb{E}_{p_{\text{data}}}(\langle s_\theta(\mathbf{x}), \nabla_x \log(p_{\text{data}}(\mathbf{x}))\rangle) = -\int p_{\text{data}}(\mathbf{x})\langle s_\theta(\mathbf{x}), \nabla_x \log(p_{\text{data}}(\mathbf{x}))\rangle dx$$

Also, we have that $\nabla_x \log(p_{\text{data}}(\mathbf{x})) = \frac{\nabla_x p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})}$. We suppose that $\mathbf{x}$ is in the support of p_{data}. Therefore, we can simplify it in the integral

$$\begin{aligned}
-\mathbb{E}_{p_{\text{data}}}(\langle s_\theta(\mathbf{x}), \nabla_x \log(p_{\text{data}}(\mathbf{x}))\rangle) &= -\int \langle s_\theta(\mathbf{x}), \nabla_x p_{\text{data}}(\mathbf{x})\rangle dx \\
&= -[s_\theta(\mathbf{x})p_{\text{data}}(\mathbf{x})] + \int \langle \nabla_x, s_\theta(\mathbf{x})\rangle p_{\text{data}}(\mathbf{x}) dx \text{ by integration by parts} \\
&= \mathbb{E}_{p_{\text{data}}}(\langle \nabla_x, s_\theta(\mathbf{x})\rangle)
\end{aligned}$$

as we consider p_{data} to be zero on the boundaries of the support. It can be noticed that the inner product $\langle \nabla_x, s_\theta(\mathbf{x})\rangle$ corresponds to the sum of the second derivatives of the log-density, i.e, $\sum \frac{\partial^2 \log(p(\mathbf{x},\theta))}{\partial \mathbf{x}^2}$ which can be rewritten as the trace of the hessian of the log-density or the trace of the Jacobian of s_θ. Finally, we have rewritten the problem as :

$$\min_\theta \mathbb{E}_{p_{\text{data}}}\left(\frac{1}{2}\|s_\theta(\mathbf{x})\|^2 + \text{tr}(\nabla_x s_\theta(\mathbf{x}))\right)$$

A.2 Marginal conditional distribution of the diffusion process

By induction, we can show that the marginal conditional distribution of the diffusion process is given by:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \quad \text{where} \quad \bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)\,.$$

Let's prove this by induction. For $t = 1$, we have:

$$q(\mathbf{x}_1|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_1; \sqrt{1-\beta_1}\mathbf{x}_0, \beta_1\mathbf{I}) = \mathcal{N}(\mathbf{x}_1; \sqrt{\bar{\alpha}_1}\mathbf{x}_0, (1-\bar{\alpha}_1)\mathbf{I}).$$

Now, let's assume that the marginal conditional distribution of the diffusion process is given by $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ for a given t. Then we can write

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{w}_t$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Let $\mathbf{w}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be independent of $\mathbf{w}_t$. Then by the definition of the diffusion process, we have:

$$\begin{aligned}
\mathbf{x}_{t+1} &= \sqrt{1-\beta_{t+1}}\mathbf{x}_t + \sqrt{\beta_{t+1}}\mathbf{w}_{t+1} \\
&= \sqrt{1-\beta_{t+1}}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{w}_t\right) + \sqrt{\beta_{t+1}}\mathbf{w}_{t+1} \\
&= \sqrt{\bar{\alpha}_{t+1}}\mathbf{x}_0 + \sqrt{\alpha_t\,(1-\bar{\alpha}_t)}\mathbf{w}_t + \sqrt{\beta_{t+1}}\mathbf{w}_{t+1} \\
&= \sqrt{\bar{\alpha}_{t+1}}\mathbf{x}_0 + \sqrt{\alpha_t\,(1-\bar{\alpha}_t) + \beta_{t+1}}\boldsymbol{\varepsilon}_{t+1} \quad \text{where} \quad \boldsymbol{\varepsilon}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\bar{\alpha}_{t+1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t+1}}\boldsymbol{\varepsilon}_{t+1}
\end{aligned}$$

Therefore, we have $q(\mathbf{x}_{t+1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t+1}; \sqrt{\bar{\alpha}_{t+1}}\mathbf{x}_0, (1-\bar{\alpha}_{t+1})\mathbf{I})$. This concludes the proof by induction.

A.3 Backward diffusion process closed-form solution

By applying the Bayes rule, we can write the backward diffusion process as follows:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \propto q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)$$

We know that $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$ by the definition and markov property of the diffusion process. We also know that $q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})$ by the previous result. Therefore, we have:

$$\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &\propto \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})\mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I}) \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{1}{\beta_t}\left\|\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1}\right\|^2 + \frac{1}{1-\bar{\alpha}_{t-1}}\left\|\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\right\|^2\right)\right) \\
&\propto \exp\left(-\frac{1}{2}\left(-\frac{2\sqrt{\alpha_t}}{\beta_t}\left\langle\mathbf{x}_t, \mathbf{x}_{t-1}\right\rangle + \frac{\alpha_t}{\beta_t}\left\|\mathbf{x}_{t-1}\right\|^2\right.\right. \\
&\left.\left. -\frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\left\langle\mathbf{x}_{t-1}, \mathbf{x}_0\right\rangle + \frac{1}{1-\bar{\alpha}_{t-1}}\left\|\mathbf{x}_{t-1}\right\|^2\right)\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\left\|\mathbf{x}_{t-1}\right\|^2\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) - \left\langle\mathbf{x}_{t-1}, \frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right\rangle\right)\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\gamma\left\|\mathbf{x}_{t-1}\right\|^2 - \left\langle\mathbf{x}_{t-1}, \mathbf{v}\left(\mathbf{x_t}, \mathbf{x_0}\right)\right\rangle\right)\right)
\end{aligned}$$

Hence, by identifying the terms, we have:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{\mathbf{v}\left(\mathbf{x_t}, \mathbf{x_0}\right)}{\gamma}, \frac{1}{\gamma}\mathbf{I}\right)$$

We can now express the vector $\tilde{\mu}\left(\mathbf{x_t}, \mathbf{x_0}\right)$ and the scalar $\tilde{\beta}_t$ as follows:

$$\tilde{\mu}\left(\mathbf{x_t}, \mathbf{x_0}\right) = \frac{\mathbf{v}\left(\mathbf{x_t}, \mathbf{x_0}\right)}{\gamma} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}\mathbf{x_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_{t-1}}\mathbf{x_0}$$

$$\tilde{\beta}_t = \frac{1}{\gamma} = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Using the results from the marginal conditional distribution of the diffusion process, we can express the vector $\mathbf{x_0}$ as follows:

$$\mathbf{x_t} = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t \quad \text{where} \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Hence we have :

$$\mathbf{x_0} = \frac{\mathbf{x_t} - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t}{\sqrt{\bar{\alpha}_t}}$$

We can now express the vector μ as a function of $\mathbf{x_t}$ and t:

$$\tilde{\mu}(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_t\right)$$

A.4 Lower bound of the log-likelihood

The log-likelihood of the diffusion process is given by:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{x}_0)} \left[-\log p_\theta(\mathbf{x}_0) \right]$$

wich isn't quite computable in practice due to its dependence on $T-1$ more random variables. However, we can derive a lower bound of the log-likelihood by using Jensen's inequality. We have:

$$\begin{aligned}
\mathcal{L} &= -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \Big) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} \Big) \\
&= -\mathbb{E}_{q(\mathbf{x}_0)} \log \Big(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big) \\
&\leq -\mathbb{E}_{q(\mathbf{x}_{0:T})} \log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})} \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \Big] = \mathcal{L}_{\text{VLB}}
\end{aligned}$$

To be analytically tractable, the objective function can be rewritten as a sum of Kullback-Leibler divergences Weng [2021]:

$$\begin{aligned}
\mathcal{L}_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \Big] \\
&= \mathbb{E}_q \Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)} \Big] \\
&= \mathbb{E}_q \Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \Big] \\
&= \mathbb{E}_q [\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{\mathcal{L}_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_0}]
\end{aligned}$$

Notice that $\mathcal{L}_T$ do not depend on θ and can be ignored in the optimization process because $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Ho et al. [2020] models $\mathcal{L}_0$ using a separate discrete decoder derived from $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \boldsymbol{\Sigma}_\theta(\mathbf{x}_1, 1))$. For $t \in \{2, \dots, T\}$, we have:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I}) \quad \text{and} \quad q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I})$$
$$\tilde{\mu}(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_t\Big) \quad \text{and} \quad \mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\Big)$$

Hence, we can express the lower bound of the log-likelihood as follows:

$$\begin{aligned}
\mathcal{L}_{t-1}(\theta) &= \mathbb{E}_q\Big[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\Big] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[\log \frac{\mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I})}{\mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I})}\right] \\
&\propto \mathbb{E}_q\left[\frac{1}{2\tilde{\beta}_t^2}\left\|\tilde{\mu}(\mathbf{x}_t, t) - \mu_\theta(\mathbf{x}_t, t)\right\|^2\right] \\
&= \mathbb{E}_{q,\boldsymbol{\varepsilon}}\left[\frac{1}{2\tilde{\beta}_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\Big)\right\|^2\right] \\
&= \mathbb{E}_{q,\boldsymbol{\varepsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\tilde{\beta}_t^2}\left\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right\|^2\right] \\
&= \mathbb{E}_{\mathbf{x_0},\boldsymbol{\varepsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\tilde{\beta}_t^2}\left\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t, t)\right\|^2\right]
\end{aligned}$$

Empirically, Ho et al. [2020] uses a simplified version of this variational lower bound without coefficients:

$$\mathcal{L}^S(\theta) = \mathbb{E}_{t,\mathbf{x_0},\boldsymbol{\varepsilon}}\left[\left\|\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t, t)\right\|^2\right]$$

B Results

We used the same architecture than in 2.3.3 to train a noise prediction model for DDPM. We trained our model on a dataset of anime faces of size 64×64 (Chao [2019]). We use $L = 1000$ diffusion time steps with $\beta_L = 2 \times 10^{-2}$ and $\beta_1 = 1 \times 10^{-4}$ following a linear sequence. The model was trained for 100,000 iterations with a batch size of 64 using the Adam optimizer with a learning rate of 2×10^{-5}. We use this noise prediction model to sample 81 images in Fig 7.

We also trained a second model on a dataset of 14,000 faces from paintings of size 512×512. This time we used a pre-trained VAE to encode our images in latent and we trained our diffusion model in the latent space following Rombach et al. [2022]. The hyperparameters used are the same than for the anime faces. We created the dataset manually by extracting all the faces from the dataset from van Zuijlen et al. [2021] with the metface extracting script from Karras et al. [2020]. We sample 21 images in Fig 8.

Figure 7: Random samples of a DDPM trained on an anime faces dataset Chao [2019]

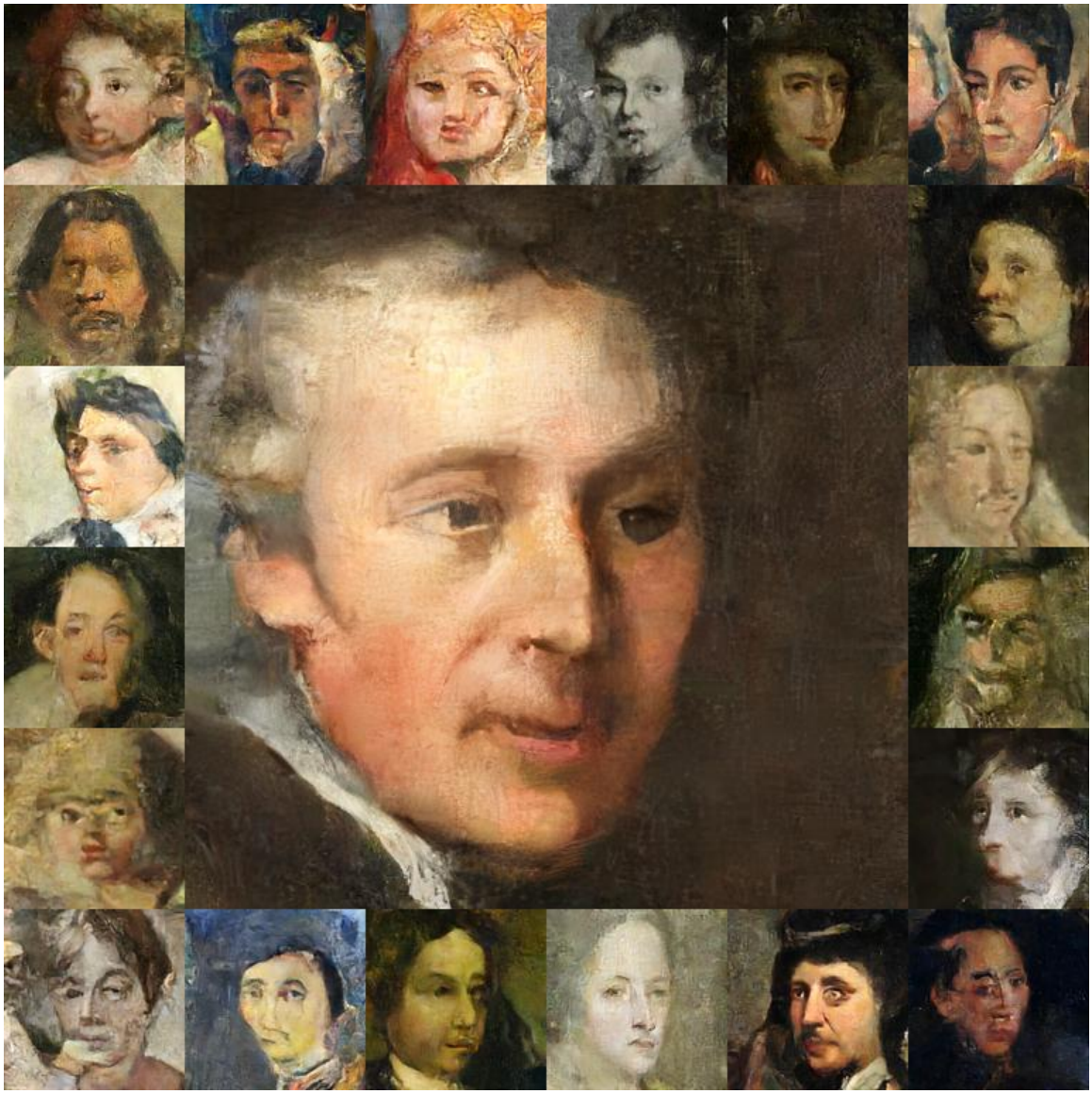


Figure 8: Samples of a DDPM trained on a dataset of faces from paintings