Denoising score matching for diffusion models

Alessio Spagnoletti*
École Normale Supérieure Paris-Saclay
aspagnol@ens-paris-saclay.fr

Antoine Ratouchniak*
École Normale Supérieure Paris-Saclay
antoine.ratouchniak@ens-paris-saclay.fr

1 Introduction

This project aims to present a global overview of Score-based generative models, starting from the origins of the idea behind **Score-matching**, as described in the seminal work of [Hyv05] that we will describe in Section 2. After this, in Section 3, we will cover the techniques used for sampling using the score function, i.e., **Langevin dynamics** as the authors in [SE19] do and explore some issues that require further analysis. To tackle these issues, in Section 4, we will explore the ideas behind **Denoising Score Matching** and how the work of [Vin11] makes feasible the use of the score for generating samples from manifold embedded in a high dimensional space. Since, at this point, all the tools have been deployed, in Section 5, we will understand how we will use them, as we will do in our experiments in Section 7. As a last (but very interesting) theoretical insight, we will see that following the work of [HJA20] on **Diffusion models**, we can see that, despite coming from entirely different approaches, the two models aim at optimizing a similar loss function.

2 Score-matching

In many applications, we are dealing with unnormalized probability distributions and are interested in computing the normalization constant. Unfortunately, this constant is often intractable, for instance, the space of all images. Formally, assume we have a dataset of i.i.d. samples $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathbb{R}^d = \mathcal{X}$ drawn from a probability distribution $p_{data}(x)$, we would like to model it with another probability distribution $\tilde{p}_q(x;\theta)$ where $\theta \in \mathbb{R}^p$ is a parameter. However, since this probability distribution may still be unnormalized, we would need to compute

$$p_q(x;\theta) = \frac{\tilde{p}_q(x;\theta)}{\int_{\mathcal{X}} \tilde{p}_q(x;\theta)dx}$$

that can be impractical due to the intractability of $\int_{\mathcal{X}} p(x,\theta)dx$. If the distribution was known, one could maximize the log-likelihood. Some previous works focused on estimating this constant ([Mac03]) In machine learning applications, we also frequently work with a finite set of observed data samples, leading to an approximation of this distribution. The choice of the model to represent probability distributions can also be implicit with GAN [GPAM$^+$14], for example, where we learn the sampling process. It can also be explicit with models such as Bayesian networks [Mac92], Markov random fields [Li09], or diffusion models [HJA20] where a part of the probability distribution is learned from the data.

In this study, we will focus on the **Stein score** of a probability distribution given by $\nabla_x \log p_{data}(x)$; we thus wish to estimate $\nabla_x \log p_q(x;\theta)$, which does not depend on the normalization constant. Finally, one would compute

$$\theta^* \in \arg\min_\theta \mathcal{L}(\theta) = \arg\min_\theta \mathbb{E}_{p_{\text{data}}}(\|\nabla_x \log p_q(x;\theta) - \nabla_x \log p_{data}(x)\|_2^2) \tag{1}$$

where $\mathcal{L}(\theta)$, under some regularity conditions (see [Hyv05], Theorem 1), can be shown to be:

$$\mathcal{L}(\theta) = \mathbb{E}_{p_{\text{data}}}\left(Tr(\nabla_x^2 p_q(x;\theta)) + \frac{1}{2}\|\nabla_x \log p_q(x;\theta)\|_2^2\right) + \overbrace{\mathbb{E}_{p_{\text{data}}}(\|\nabla_x \log p_{data}\|_2^2)}^{\text{constant}} \tag{2}$$

where the second term appears as a constant since it does not depend on θ.

Finally, according to Theorem 2 of [Hyv05], we remind the following proposition:

Proposition 1 *Let θ^* be such that $p_q(x;\theta^*) = p_{data}(x)$. Assume further that no other parameter value gives a pdf that is equal to $p_q(x;\theta^*)$, and that $\tilde{p}_q(x;\theta) > 0$ for all x, θ. We then have:*

$$\theta = \theta^* \iff \mathcal{L}(\theta) = 0 \tag{3}$$

*The authors contributed equally.

3 Langevin dynamics for Generative Modelling

Since the score is a vector field pointing in the direction where the log data density grows the most, the main idea is that following it can lead from a white noise image to one lying on the dataset manifold: the support of the probability density $p_{\text{data}}(x)$ related to our dataset. To find this path, we use **Langevin dynamics**:

$$\dot{X}_t = \frac{1}{2}\nabla \log p_{\text{data}}(X_t) + \dot{W}_t \tag{4}$$

where W is a Brownian motion. This particular Itô diffusion process can be proved to have as a solution a process X_t whose density $\rho_t(X)$ (obtained solving the related Fokker-Planck equation) converges to $p_{\text{data}}(X_t)$ as $t \to \infty$.

Informally, the Langevin dynamics drive the random walk towards high-probability regions in the manner of a gradient flow. This evolution is suited to our model since it can produce samples from a probability density $p_{\text{data}}(x)$ requiring only its score function. In practice, discretizing 4 with the standard Euler–Maruyama method [BPB18], given a fixed step size $\epsilon > 0$, and an initial value $\tilde{x}_0 \sim \pi(x)$ with π being a prior distribution, we have:

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p_{\text{data}}(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t \tag{5}$$

where $z_t \sim \mathcal{N}(0, I)$. So, it can be proved that for $\epsilon \to 0$ and $T \to \infty$, we have an actual sample form $p_{\text{data}}(x)$ (similar results prove that the error committed when $T < \infty$ is small, ensuring that an actual algorithm can be run, as can be seen in [RR98]).

Given a dataset consisting of i.i.d. samples $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$, what we aim to do is to train a neural network $s_\theta : \mathbb{R}^d \to \mathbb{R}^d$ parametrized by θ, called *Score Network*. It will learn to approximate the score of the dataset probability density $p_{\text{data}}(x)$ so that $s_\theta(x) \simeq \nabla_x \log p_{\text{data}}(x)$ (i.e. when $\theta \simeq \theta^*$ as seen in Proposition 1). Then, we will use this network to produce samples following the Langevin dynamics.

3.1 Challenges and solutions of the Score-based approach

The technique proposed above has to face various issues; they are **The manifold hypothesis** and the **Low data density regions**.

3.1.1 The manifold hypothesis

The manifold hypothesis says that data in the real world tend to concentrate on low-dimensional manifolds embedded in a high-dimensional space. Let us think, for example, of the well-known MNIST dataset; it is clear that the manifold on which all the possible hand-written digits can lie is much smaller than the 28×28 dimensional space in which they are embedded (for example, the pixels on the border will always be black, so these constraints reduce by a lot the dimensions).

Under this hypothesis, the two main problems are the following:

- Since the score $\nabla_x \log p_{\text{data}}(x)$ is a gradient taken in the ambient space, it is undefined when x is outside the manifold (i.e., the support).
- Proposition 1 in Section 2 can be applied only when the support of the data distribution is the whole space. Without it, we will not have any theoretical support for our network to approximate the score function.

3.1.2 Low data density regions

Recall from Section 2 that score matching minimizes the expected squared error of the score estimates,

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\|\nabla_\theta \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|_2^2\right].$$

where the expectation w.r.t. the data distribution is always estimated using i.i.d. samples $\mathcal{D} = \{x_1, \dots, x_n\} \sim p_{\text{data}}(x)$. Consider now any region $\mathcal{R} \subset \mathbb{R}^d$ such that $p_{\text{data}}(\mathcal{R}) \approx 0$. In most cases, $\{x_1, \dots, x_n\} \cap R = \emptyset$, so the main issue is that score matching will not have sufficient data samples to estimate $\nabla_x \log p_{\text{data}}(x)$ accurately for $x \in R$.

Furthermore, consider the case in which a low-density region separates two high-density ones, like in the case of a bi-modal distribution $p_{\text{data}}(x) = \pi p_1(x) + (1-\pi)p_2(x)$, where $p_1(x)$ and $p_2(x)$ are normalized distributions with disjoint supports, and $\pi \in (0, 1)$. In the support of $p_1(x)$, we have $\nabla_x \log p_{\text{data}}(x) = \nabla_x(\log \pi + \log p_1(x)) = \nabla_x \log p_1(x)$, and in the support of $p_2(x)$, we have $\nabla_x \log p_{\text{data}}(x) = \nabla_x(\log(1-\pi) + \log p_2(x)) = \nabla_x \log p_2(x)$. In either case, the score $\nabla_x \log p_{\text{data}}(x)$ does not depend on π. As Langevin dynamics use $\nabla_x \log p_{\text{data}}(x)$ to sample from $p_{\text{data}}(x)$, the samples obtained will not depend on π, i.e., they will not reflect correctly the true distribution.

3.1.3 Solution

The idea exploited in [SE19] is to perturb the data distribution with a Gaussian noise $q_\sigma(\tilde{x}|x) = \frac{1}{(2\pi)^{d/2}\sigma^d} e^{-\frac{1}{2\sigma^2}||\tilde{x}-x||^2}$ and then to employ score matching to estimate the score of the perturbed data distribution

$$q_\sigma(\tilde{x}) := \int q_\sigma(\tilde{x} \mid x)\, p_{\text{data}}(x)\, dx. \tag{6}$$

In the next section, we will explain how this idea can be used in practice and see its theoretical foundations. Now, let us see why this solves the problems stated above.

Since the Gaussian noise is isotropic, the support of the perturbed distribution will be the whole space, ensuring the applicability of Proposition 1 and thus solving both issues related to **the manifold hypothesis**. Furthermore, with big enough σ, we can produce samples in low-density regions of the original (unperturbed) data distribution, solving the first issue related to the **low data density regions**. We will see that the second issue is solved by simultaneously learning a score function for different noise levels. In this way, we can recover information by sampling in low-density regions using the Langevin dynamics with large σ values, and then, as t grows, use small σ to recover the precise data distribution.

4 Denoising Score Matching

Drawing from [Vin11] and [SE19], this section explores the intersection of Denoising Autoencoders (DAEs) and Score Matching (SM), using the notation that two loss functions J_1 and J_2 are equivalent ($J_1 \backsim J_2$) if they differ by scaling and a constant term.

4.1 Score Matching with Perturbed Density

We defined $q_\sigma(\tilde{x})$ in (6) as the perturbed, smooth version of our data distribution, also called the Parzen windows density estimator. Our objective is to align the parametrized score estimator $\psi(x;\theta)$ with the score of $q_\sigma(\tilde{x})$, formalized as:

$$J_{ESMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{x})}\left[\frac{1}{2}\left\|\psi(\tilde{x};\theta) - \frac{\partial \log q_\sigma(\tilde{x})}{\partial \tilde{x}}\right\|^2\right], \tag{7}$$

which aligns with J_{ISMq_σ} under the regularity conditions specified in Theorem 1 of [Hyv05] (which gives (2)). However, this equivalence breaks as $\sigma \to 0$.

4.2 Denoising Score Matching (DSM)

As explored in [Vin11], we consider the denoising version of a simple classical autoencoder that employs a single sigmoidal hidden layer. The model assumes that data points originate from a continuous real-valued distribution, and thus, a linear decoder with a squared reconstruction loss is used. Tied weights are employed, where the encoder and decoder share the same linear transformation parameters. The corruption is introduced as additive isotropic Gaussian noise (details in Appendix B). In the end, the loss can be written as:

$$J_{DAE\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\|\operatorname{decode}(\operatorname{encode}(\tilde{\mathbf{x}})) - \mathbf{x}\|^2\right] = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\|\mathbf{W}^T \operatorname{sigmoid}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) + \mathbf{c} - \mathbf{x}\|^2\right].$$

Merging concepts from DAEs and SM, DSM uses pairs of clean and corrupted samples $(x, \tilde{x})$. The DSM objective is to approximate the score of the conditional density $q_\sigma(\tilde{x} \mid x)$, leading to the objective:

$$J_{DSMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{x},x)}\left[\frac{1}{2}\left\|\psi(\tilde{x};\theta) - \frac{\partial \log q_\sigma(\tilde{x} \mid x)}{\partial \tilde{x}}\right\|^2\right]. \tag{8}$$

The core idea is to follow the gradient ψ towards the clean sample x, with $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}$ guiding this process. Most importantly, this alternate objective is equivalent to explicit score matching. Formally,

$$J_{ESMq_\sigma} \backsim J_{DSMq_\sigma}.$$

The proof made by [Vin11] only requires that $\log q_\sigma(\tilde{x} \mid x)$ is differentiable with respect to $\tilde{x}$, and the Gaussian kernel satisfies this condition.

Using the perturbed distribution q_σ as in J_{ESMq_σ} ensures convergence of the score to $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x - \tilde{x})$ that guarantees that we are moving towards the actual sample x. This is why the idea exposed in Paragraph 3.1.3, in addition to solving the problems seen, still guarantees that the learned score will tend to move points to the real data distribution (like if we are climbing the Gaussians hills towards the actual x). However, this is not the only benefit this method has; we will now see an implementation issue not considered before that DSM can solve, the so-called *scalability issue*.

It is important, indeed, to highlight the fact that computing $\sum_{i=1}^{d} \frac{\partial \psi_i(\tilde{x};\theta)}{\partial \tilde{x}_i}$ becomes pretty hard when the data dimension is high and the neural network very deep. This makes J_{ISMq_σ} no more suitable to our purposes. Luckily, J_{DSMq_σ} does not suffer from this issue; indeed, it is straightforward to compute thanks to the special form of $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}$. Thus, we have to optimize J_{DSMq_σ} in practical applications, exploiting the equivalence:

$$J_{DSMq_\sigma} \backsim J_{ISMq_\sigma} \backsim J_{ESMq_\sigma}. \tag{9}$$

5 NCSN model

To apply the ideas seen up to now, we will describe the NCSN (Noise Conditional Score Network) model introduced by [SE19]. The objective is to add noise to the data and be able to recover samples for the original distribution afterwards. In order to do so, we can use different noise levels, denoted as $\sigma_1, \ldots, \sigma_L$ with $L > 0$. We will discuss how to choose these values later. For each noise level, we aim to approximate $\nabla_x \log q_\sigma(x)$ with a neural network, where the authors chose a U-Net architecture [RFB15] with dilated convolutions [YK15], that we will call $s_\theta(x, \sigma)$. In later experiments, the authors discovered that it is enough to rescale by σ a single unconditional network $\frac{s_\theta(x)}{\sigma}$ [SE20], instead of training an architecture for each noise level.

In practice, once the noise levels are defined, we select uniformly at each iteration of the learning procedure a random number between 1 and L and run a step of stochastic gradient descent on $s_\theta(x, \sigma)$. This yields to compute the DSM loss seen in 8:

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

After training the NCSN, we can use it for sampling through the already-seen Langevin dynamics, with the only caveat that we must be able to use the various scores learned, going from the initial high noise to the last low one. We will use the following algorithm, called Annealed Langevin Dynamics (since it is inspired by the Annealed Importance Sampling technique). The heuristic behind this is that we are allowing the first steps to move the generated sample on the noise-expanded manifold, trying to push it toward the higher-density regions. In this way, independently from the starting point, it should be able to be moved to any portion of the manifold. Gradually, this noise decreases, allowing the sample to get closer to the original manifold. In the end, $\tilde{x}_T$ will be a sample that comes from the last distribution, which has almost zero noise, and thus, it should be a valid sample that belongs to the dataset manifold.

Algorithm 1 Annealed Langevin Dynamics

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$
Initialize $\tilde{x}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \frac{\sigma_i^2}{\sigma_L^2}$ ▷ α_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 for
  $\tilde{x}_0 \leftarrow \tilde{x}_T$
end for
return $\tilde{x}_T$

6 Connection between diffusion models and score-matching

Applying score-matching at a given noise will naturally not cover all the regions, resulting in improper Langevin dynamics in low-density areas. The solution that was proposed is the injection of noise into the data, which provides an additional training signal. Song et al. [SSDK$^+$20] introduce the major step which involves perturbing the data using a diffusion process which is a form of a stochastic differential equation (SDEs). The SDE is then reversed using annealed Langevin dynamics leading to a generative process, where the reverse process makes use of score matching.

Diffusion models: diffusion models operate on the principle of transforming data through a series of learned Gaussian transitions starting from $p(x_T) = \mathcal{N}(x_T; 0; I)$, forming a Markov chain known as the reverse process. This process is represented as:

$$p_\theta(x_{0:T}) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Forward process: the forward process in diffusion models, denoted as $q(x_{1:T}|x_0)$, is also a Markov chain. It is defined by adding Gaussian noise to the data over time, following a variance schedule $\beta_1, \ldots, \beta_T$:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_t|x_{t-1}) := \mathcal{N}(x_t; (1-\beta_t)x_{t-1}, \beta_t I)$$

This recursive formulation allows direct sampling of x_t, with t an arbitrary timestep

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and $\alpha_t = 1 - \beta_t$, which shows that we can sample any noisy version x_t within a single step, with a fixed variance β_t.

Sampling from $q(\mathbf{x}_t|\mathbf{x}_0)$ is performed via the reparametrization trick. Let $x \sim \mathcal{N}(\mu, \sigma^2\mathbf{I})$, we first z-score z by setting $\tilde{z} = \frac{x-\mu}{\sigma}$, where $\tilde{z} \sim \mathcal{N}(0, \mathbf{I})$. The inverse of this normalization yields x from $\tilde{z}$ by scaling by σ and shifting by μ. In this context, the process gives:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\tilde{z}_t,$$

where $\tilde{z}_t \sim \mathcal{N}(0, \mathbf{I})$.

Reverse process: to generate new instances from $p(x_0)$, we start from $x_T \sim \mathcal{N}(0, I)$ and reverse the process using $p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t))$. A neural network $p_\theta(x_{t-1}|x_t)$ is trained to emulate this, predicting mean $\mu_\theta(x_t, t)$ and covariance $\Sigma_\theta(x_t, t)$ from the perturbed image x_t at timestep t.

Directly maximizing $p_\theta(x_0)$ for each x_0 is infeasible due to the complexity of aggregating all reverse trajectories. Instead, we minimize the variational lower bound on the negative log-likelihood to effectively train the network.

The training optimizes the variational lower-bound on negative log-likelihood:

$$\mathcal{L}_{vlb} = -\log p_\theta(x_0|x_1) + \text{KL}\left(p(x_T|x_0) \parallel \pi(x_T)\right) + \sum_{t>1}\text{KL}\left(p(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)\right), \tag{10}$$

focusing on the closeness of $p_\theta(x_{t-1}|x_t)$ to the true posterior of the forward process.

Ho et al. [HJA20] propose a specific parameterization for $\mu_\theta(x_t, t)$, leading to:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}x_t - \sqrt{\frac{\beta_t}{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t). \tag{11}$$

The simplified objective for denoising score matching becomes:

$$\mathcal{L}_{simple} = \mathbb{E}_{t\sim\mathcal{U}([1,T])}\mathbb{E}_{x_0\sim p(x_0)}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}(\|\epsilon - \epsilon_\theta(x_t, t)\|^2), \tag{12}$$

where the network predicts noise, and the mean is defined as in the equation, while covariance is fixed.

7 Experiments

In this section, we conduct experiments exploring different combinations of hyperparameters and stressing the limits of [SE19] model. We used the same optimizer and learning rates as the original paper throughout all the experiments in this section. We also flip randomly the data when training.

Noise scale: As mentioned in section 5, the authors decided to perturb the data by employing different values of σ to mitigate the impact of the manifold hypothesis described in section 3.1. They chose to use a geometric sequence $\{\sigma_k = \sigma_0 r^k\}_{k=0}^{L-1}$ with $|r| < 1$. In a later article [SE20], they explained the theoretical motivations for this choice. Nevertheless, in most recent articles, such as the work by Nichol et al. [ND21] on diffusion models, they proposed to use a cosine scheduler instead. It is important to mention that the models are nonetheless different, we use a Noise Conditional Score Network (NCSN), and they use a diffusion model, described in section 6. We explore different sequences and explain our motivations subsequently. The schedulers are: a linear sequence, a cosine sequence, a squared cosine sequence, and a sigmoid sequence, shown in figure 1. We define them as follows:

- linear scheduler $\{\sigma_k = \sigma_0 - kr\}_{k=0}^{L-1}$

- cosine scheduler $\{\sigma_k = \sigma_0 \cos(\frac{k}{\alpha L} \cdot \frac{\pi}{2})\}_{k=0}^{L-1}$ where α, similarly to [ND21], is a parameter that prevents the noise to get too close to 0, we choose $\alpha = 1.0064$.

- squared cosine scheduler $\{\sigma_k = \sigma_0 \cos(\frac{k}{\alpha L} \cdot \frac{\pi}{2})^2\}_{k=0}^{L-1}$, where $\alpha = 1.068$

- sigmoid scheduler $\{\sigma_k = \frac{-1}{1+\exp(-k-\alpha)} + \beta\}_{k=0}^{L-1}$, where $\alpha = 2.5$ and $\beta = 1.009$

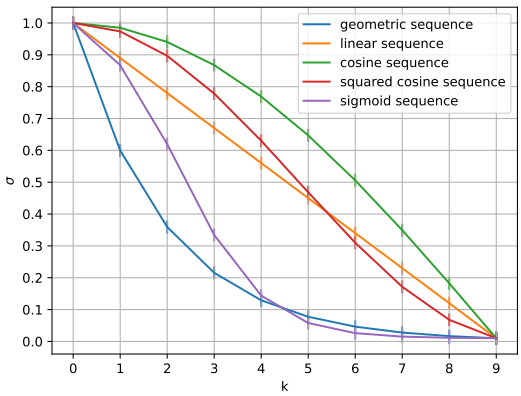


Figure 1: Evolution of σ with different schedulers

We train and test our results on the MNIST and CIFAR-10 [LLWT15] datasets. We let $L = 10$ with $\sigma_0 = 1$ and $\sigma_9 = 0.01$ like in the original article. We also use annealed Langevin dynamics with $T = 100$ and $\epsilon = 2 \times 10^{-5}$. To assess the quality of our generated samples, we employ the Fréchet Inception Distance (FID) introduced by Heusel et al. [HRU$^+$17]. Taking an Inception trained on ImageNet [SVI$^+$16], the generated samples are fed into the network, and subsequently, the features of the final layer, treated as multivariate Gaussians, are compared using the Wasserstein-2 distance against real samples. The Wasserstein-2 distance between two multivariate normal distributions $\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)$ is:
$W_2^2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)) =$

Scheduler	FID
Geometric	43.33
Linear	70.97
Cosine	130.70
Squared cosine	40.44
Sigmoid	32.46

Figure 2: MNIST

Scheduler	FID
Geometric	34.72
Linear	79.09
Cosine	130.63
Squared cosine	66.38
Sigmoid	47.99

Figure 3: CIFAR-10

Figure 4: FID scores on MNIST and CIFAR-10 with different schedulers

$$\|\mu_1 - \mu_2\|_2^2 + Tr(\Sigma_1) + Tr(\Sigma_1) - 2 \cdot Tr\left((\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2})^{1/2}\right) \tag{FID}$$

We follow the strategy used by Song et al., generating 1000 samples at each iteration and computing the corresponding FID. Given the lowest FID among all the iterations, we compute the FID again on 10000 samples and report the results in tables 2 and 3. The generation can be seen in figures 9 and 10.

One can notice that the FID score we obtained for the geometric sequence on CIFAR-10 is higher than the FID provided in the paper (25.32). This may be explained by the randomness of the generated samples and actual samples used. However, this still gives us a benchmark to compare the schedulers.

Theoretical guidance: following the improved techniques paper, the goal is to ensure that there is enough training data points spread in the high-density regions of the previous scales. We remind the second proposition of the paper.

Proposition 2 *Let* $\mathbf{x}_i \in \mathbb{R}^d \sim \mathcal{N}(\mathbf{0}, \sigma_i^2 \mathbf{I})$ *and* $r_i = \|\mathbf{x}_i\|_2$, *then* $r_i = X_i \approx \mathcal{N}(\sqrt{d}\sigma_i, \sigma_i^2/2)$.

With this proposition, the goal is to make samples from X_i to be in the range $\mathcal{I}_{i-1} = [\sqrt{d}\sigma_{i-1} - 3 \cdot \frac{\sigma_{i-1}}{\sqrt{2}}, \sqrt{d}\sigma_{i-1} + 3 \cdot \frac{\sigma_{i-1}}{\sqrt{2}}]$ to make sure they are covered in the previous density region. To do so, we want:

$$\begin{aligned}
\mathbb{P}(X_i \in \mathcal{I}_{i-1}) &= \mathbb{P}\left(\sqrt{d}\sigma_{i-1} - 3 \cdot \frac{\sigma_{i-1}}{\sqrt{2}} \leq X_i \leq \sqrt{d}\sigma_{i-1} + 3 \cdot \frac{\sigma_{i-1}}{\sqrt{2}}\right) \\
&= \mathbb{P}\left(\frac{\sqrt{d}(\sigma_{i-1} - \sigma_i) - 3 \cdot \frac{\sigma_{i-1}}{\sqrt{2}}}{\sigma_i/\sqrt{2}} \leq \frac{X_i - \sqrt{d}\sigma_i}{\sigma_i/\sqrt{2}} \leq \frac{\sqrt{d}(\sigma_{i-1} - \sigma_i) + 3 \cdot \frac{\sigma_{i-1}}{\sqrt{2}}}{\sigma_i\sqrt{2}}\right) \\
&= \mathbb{P}\left(\sqrt{2d}(\gamma_i - 1) - 3 \cdot \gamma_i \leq Z \leq \sqrt{2d}(\gamma_i - 1) + 3 \cdot \gamma_i\right) \qquad Z \sim \mathcal{N}(0, 1) \\
&= \phi(\sqrt{2d}(\gamma_i - 1) + 3 \cdot \gamma_i) - \phi(\sqrt{2d}(\gamma_i - 1) - 3 \cdot \gamma_i) \geq C \qquad \phi(\cdot) \text{ is the CDF of } Z
\end{aligned}$$

where $\gamma_i = \frac{\sigma_{i-1}}{\sigma_i}$ and $C > 0$ to be as maximum as possible. However, as the dimension d grows, the number of noise scales L should grow as well to keep a suitable C, which can be computationally expensive. From now on, we shift the indices of the sequences by one and denote σ_0 by σ_1 and σ_{L-1} by σ_L. Going back to an example, with images of size $32 \times 32 \times 3$, using a geometric sequence with $\sigma_1 = 1$, $\sigma_L = 0.01$, and $L = 10$, yields $P(X_i \in \mathcal{I}_{i-1}) \approx 0 \; \forall i \in \{2, \ldots, L\}$ since $\gamma_2 = \cdots = \gamma_L$. Driven by this theory, we tried different

sequences. Our first experiments were such that $\gamma_2 < \cdots < \gamma_L$ with the linear and cosine sequence to get a higher C in the early steps. Indeed, let $\gamma_k = \frac{\sigma_k}{\sigma_{k+1}}$, we get for a linear sequence of the form $\sigma_k = \sigma_1 - kr$, $\gamma_k = 1 + \frac{r}{\sigma_1 - kr - r}$, that is a strictly increasing sequence as the derivative w.r.t. k is $\gamma'_k = \frac{r^2}{(rk - \sigma_1 + r)^2} > 0$. Similarly, let $\sigma_k = \sigma_1 \cos(k/L \cdot \frac{\pi}{2})$, it is a strictly increasing sequence as the derivative w.r.t. k, $\gamma'_k = \frac{\sigma_1 \sin(\frac{\pi}{2L})}{2L \cos(\frac{\pi(k+1)}{2L})^2} > 0$. The squared cosine sequence is also a strictly increasing sequence as $\gamma_i > 0 \; \forall i \in \{2, \dots, L\}$. We expected distant regions from the original distribution to require more samples than the ones near high-density regions (i.e., low σ). However, due to the high dimensionality, a certain quantity of noise is still required near the training data points.

Besides that, we tried a sigmoid sequence of the form $\sigma_k = \frac{-1}{1+e^{-x-\alpha}+\beta}$ where $\gamma_k = \frac{1+e^{-x-1-\alpha}+\beta}{1+e^{-x-\alpha}+\beta}$ with its derivative $\gamma'_k \propto -(e^{2k} - \beta)$ that is first positive until $k > \log(\beta)/2$ and then negative. By establishing such a sequence, our expectations were such that in the early and last steps, we would need more data points. In the first steps, as previously discussed, we believe that an increased number of samples is beneficial, while in the last steps, this could lead to more accurate generations. In the intermediate steps, we would expect the presence of sufficiently robust gradients. Although we can achieve better performances with the sigmoid scheduler on MNIST, this is not the case when the dimension increases with CIFAR.

Initial noise: our experiments aim to understand the importance of the initial noise for obtaining good samples, especially concerning the distribution among the different classes. Following the issue raised in Section 3.1, some areas may not be well covered if the noise is not strong enough. For this reason, we tried to change the initial noise value σ_1 to 30 and increase the number of scales to $L = 200$. We show our results in figures 11 and 12.

Despite their visual similarities, we observe that the distributions of these two configurations vary (see histograms 13 and 14) [2].

Scheduler	FID	Scheduler	FID
Geometric	9.65	Geometric	40.05
Squared cosine	220.42	Squared cosine	305.74
Sigmoid	22.69	Sigmoid	47.59

Figure 5: MNIST Figure 6: CIFAR-10

Figure 7: FID scores on MNIST and CIFAR-10 with different schedulers with $\sigma_1 = 30$ and $\sigma_1 = 50$ respectively

It is clear that the initial noise value can help to make the distribution more uniform. Indeed, by setting $\sigma_1 = 1$, none of the classes reaches 10% in its confidence interval, while by setting $\sigma_1 = 30$, 8 of them managed to do so. Our intuition to explain the results from figure 13 is that some numbers get clustered together (i.e. 6 and 8) while some may be alone (i.e. 1) hence, the need to add more noise to cover these regions.

Using $L = 200$, we also experimented with generating samples from CIFAR-10 with an initial noise value of $\sigma_1 = 30$ in table 6. However, no discernible improvements were observed as can be seen in table 3. This is probably due to the model not being able to capture more information.

Additional dataset: finally, we tried to generate images with another dataset that has less variability. The Stanford cars dataset contains 8144 images of cars that we resized to 32×32. We only kept the geometric and sigmoid noise schedules as they were the most promising in the previous sections. The number of scales is still $L = 200$ with $\sigma_1 = 50$ and $\sigma_{200} = 0.01$. We report our results in table 8.

Scheduler	FID
Geometric	26.35
Sigmoid	51.31

Figure 8: FID scores on the Stanford cars dataset with $\sigma_1 = 50$

8 Conclusion

In this work, we have seen a powerful technique to generate images leveraging score matching and Langevin dynamics. We made a link with diffusion models that are considered state-of-the-art. Finally, we tried different combinations of hyperparameters. This work has been a precursor to many techniques used nowadays in generative AI.

[2] The label for each digit was obtained by a 3 layers-CNN with 99.5% accuracy on the MNIST test dataset.

Bibliography

[And82] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[BPB18] Mustafa Bayram, Tugcem Partal, and Gulsen Orucova Buyukoz. Numerical methods for simulation of stochastic differential equations. *Advances in Difference Equations*, 2018:1–10, 2018.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[HJA20] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

[HRU+17] 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.

[Hyv05] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709, 2005.

[Li09] Stan Z Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[Mac92] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[Mac03] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[RR98] Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60, 1998.

[SE19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Neural Information Processing Systems*, 2019.

[SE20] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

[SGSE20] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[SSDK+20] 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.

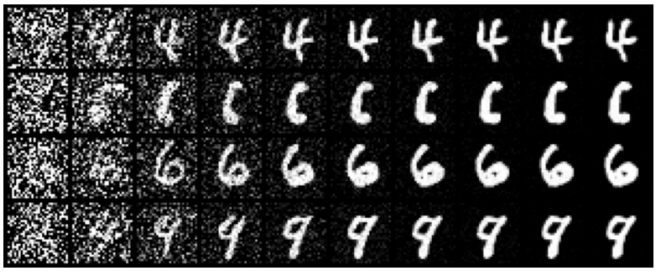
[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[Vin11] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23:1661–1674, 2011.

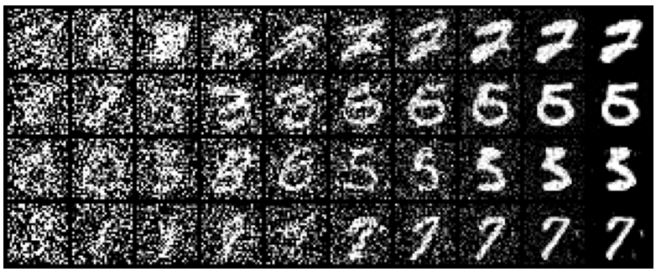
[YK15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

9 Appendix A

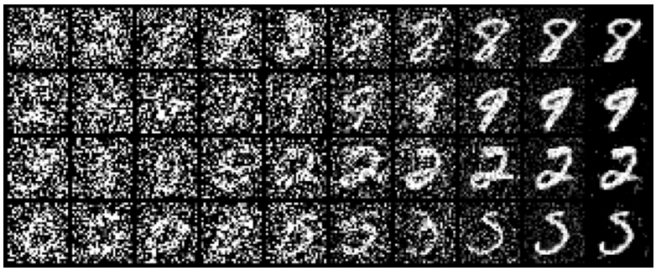
9.1 Image generation with different noise schedulers



Geometric sequence

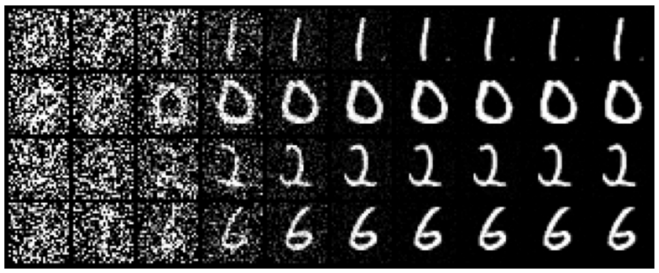


Linear sequence



Cosine sequence

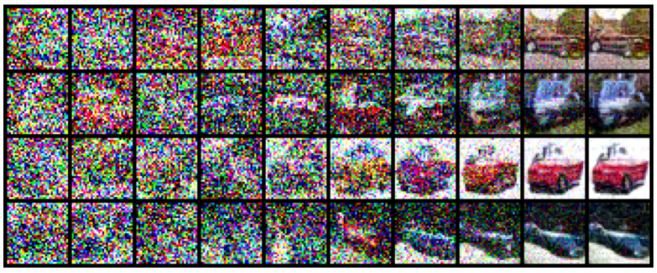
Squared cosine sequence



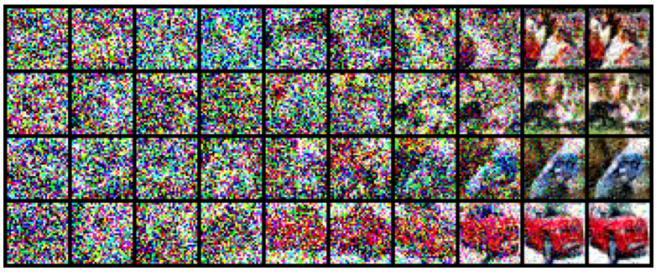
Sigmoid sequence

Figure 9: Intermediate samples of annealed Langevin dynamics with different schedulers on MNIST

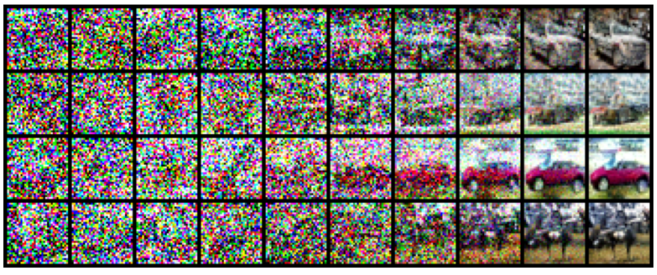
Geometric sequence



Linear sequence



Cosine sequence



Squared cosine sequence

Sigmoid sequence

Figure 10: Intermediate samples of annealed Langevin dynamics with different schedulers on CIFAR

9.2 Initial noise on the MNIST dataset

Figure 11: Samples generated with $\sigma_1 = 1$

Figure 12: Samples generated with $\sigma_1 = 30$

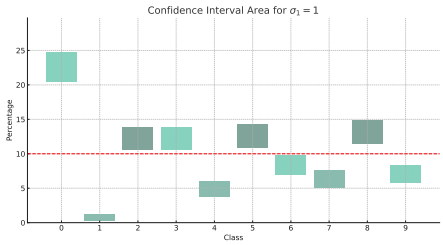


Figure 13: Distribution of the digits with $\sigma_1 = 1$

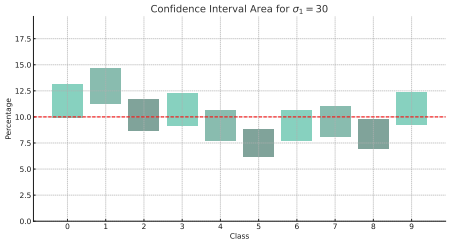


Figure 14: Distribution of the digits with $\sigma_1 = 30$

9.3 Image generation with the Stanford car dataset

Figure 15: Samples obtained after 100k steps with the geometric sequence

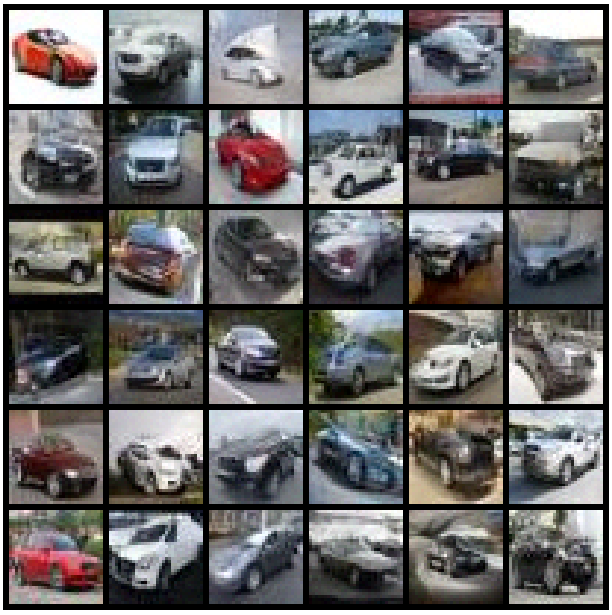


Figure 16: Samples obtained after 100k steps with the sigmoid sequence

10 Appendix B

10.1 DAE architecture

We explore here in detail the DAE architecture used by [Vin11] to establish the equivalence between DAEs and DSM.

The architecture is outlined as follows:

- **Corruption:** A training input x is corrupted by additive Gaussian noise with covariance $\sigma^2\mathbf{I}$, resulting in a corrupted input $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}\left(0, \sigma^2\mathbf{I}\right)$. The corresponding conditional density is $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$.

- **Encoding:** The corrupted version $\tilde{\mathbf{x}}$ is encoded into a hidden representation $\mathbf{h}$ using an affine mapping followed by a sigmoid nonlinearity: $\mathbf{h} = \text{sigmoid}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b})$, where $\mathbf{W}$ is a $d_h \times d$ matrix, $\mathbf{b} \in \mathbb{R}^{d_h}$.
- **Decoding:** The hidden representation $\mathbf{h}$ is decoded into the reconstruction $\mathrm{x}^{\mathbf{r}}$ through an affine mapping: $\mathbf{x}^{\mathbf{r}} = \mathbf{W}^T\mathbf{h} + \mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^d$.
- **Objective Function:** The parameters $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ are optimized to minimize the expected squared reconstruction error, given by the objective function $J_{DAE\sigma}(\theta)$:

$$\begin{aligned} J_{DAE\sigma}(\theta) &= \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\|\operatorname{decode}(\operatorname{encode}(\tilde{\mathbf{x}})) - \mathbf{x}\|^2\right] \\ &= \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\|\mathbf{W}^T \operatorname{sigmoid}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) + \mathbf{c} - \mathbf{x}\|^2\right]. \end{aligned} \tag{13}$$

10.2 Equivalence between DAEs and DSM

Denoising Score Matching can be efficiently used when Score Matching is applied to large datasets, solving the problems related to the manifold hypothesis. In this section, we will briefly investigate other interesting properties of DSM that make it equivalent to a Denoising Autoencoder when we choose a specific energy function, the Gaussian Parzen kernel, and we use the DAEs's shape as in 13.

10.2.1 The choice of the energy function

Let us start with the choice for the model p, which has the form of a Gibbs measure:

$$\begin{aligned} p(\mathbf{x};\theta) &= \frac{1}{Z(\theta)}\exp(-E(x;\theta)) \\ E(\mathbf{x};\underbrace{\mathbf{W},\mathbf{b},\mathbf{c}}_{\theta}) &= -\frac{\langle \mathbf{c},\mathbf{x}\rangle - \frac{1}{2}\|\mathbf{x}\|^2 + \sum_{j=1}^{d_h}\operatorname{softplus}\left(\langle \mathbf{W}_j,\mathbf{x}\rangle + \mathbf{b}_j\right)}{\sigma^2}. \end{aligned}$$

This specific energy function has been designed to be related to the reconstruction error in DAEs. We thus recover the score:

$$\begin{aligned} \psi_i(\mathbf{x};\theta) &= \frac{\partial \log p(\mathbf{x};\theta)}{\partial \mathbf{x}_i} \\ &= -\frac{\partial E}{\partial \mathbf{x}_i} \\ &= \frac{1}{\sigma^2}\left(\mathbf{c}_i - \mathbf{x}_i + \sum_{j=1}^{d_h}\operatorname{softplus}'\left(\langle \mathbf{W}_j,\mathbf{x}\rangle + \mathbf{b}_j\right)\frac{\partial\left(\langle \mathbf{W}_j,\mathbf{x}\rangle + \mathbf{b}_j\right)}{\partial \mathbf{x}_i}\right) \\ &= \frac{1}{\sigma^2}\left(\mathbf{c}_i - \mathbf{x}_i + \sum_{j=1}^{d_h}\operatorname{sigmoid}\left(\langle \mathbf{W}_j,\mathbf{x}\rangle + \mathbf{b}_j\right)\mathbf{W}_{ji}\right) \\ &= \frac{1}{\sigma^2}\left(\mathbf{c}_i - \mathbf{x}_i + \langle \mathbf{W}_{\cdot i}, \operatorname{sigmoid}(\mathbf{W}\mathbf{x} + \mathbf{b})\rangle\right) \end{aligned}$$

which we can write as a single equation

$$\psi(\mathbf{x};\theta) = \frac{1}{\sigma^2}\left(\mathbf{W}^T \operatorname{sigmoid}(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{c} - \mathbf{x}\right).$$

Now, recalling the equation for J_{DSMq_σ} seen in 8 and substituting the formulas for $\psi_i(\mathbf{x};\theta)$ and $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}$, we obtain:

$$\begin{aligned} J_{DSMq_\sigma}(\theta) &= \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta) - \frac{\partial \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\|^2\right] \\ &= \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\frac{1}{\sigma^2}\left(\mathbf{W}^T \operatorname{sigmoid}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) + \mathbf{c} - \tilde{\mathbf{x}}\right) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\right\|^2\right] \\ &= \frac{1}{2\sigma^4}\mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\left\|\mathbf{W}^T \operatorname{sigmoid}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) + \mathbf{c} - \mathbf{x}\right\|^2\right] \\ &= \frac{1}{2\sigma^4}J_{DAE\sigma}(\theta). \end{aligned}$$

From these relationships, we obtain that

$$J_{DSMq_\sigma} \backsim J_{DAE\sigma}.$$

10.2.2 Implications of the link between DAEs and DSM

The equivalence of DAEs and DSM in certain conditions implies that learning to denoise data in DAEs can be viewed as learning the score of the data distribution, as done in DSM. This creates a unified framework for understanding how these models learn and represent data distributions. Before this connection was established, DAEs were primarily understood as feature extraction and dimensionality reduction tools without a clear understanding of how they model the data distribution. The linkage with DSM provides a theoretical foundation explaining how DAEs capture and learn the underlying data distribution, enhancing our understanding of their functioning.

10.3 Stochastic Differential Equations

The concept of a finite number of noise scales can be expanded to an infinite, continuous range. The diffusion process, considered to be a continuous process, becomes a solution of a stochastic differential equation. It was shown in [And82] that the reverse process of this diffusion can be modeled as a reverse-time SDE, which requires the score function of the density at each time step. Hence, the SDE of the forward process $\{x_t\}_{t=0}^{T}$, is given by:

$$\frac{\partial x}{\partial t} = f(x,t) + \sigma(t) \cdot \omega_t \Longleftrightarrow \partial x = f(x,t) \cdot \partial t + \sigma(t) \cdot \partial \omega,$$

The drift coefficient, f, aims to gradually eliminate the data x_0, while the diffusion coefficient, $\sigma(t)$, characterizes the stochastic aspect of the SDE, defining the intensity of the noise infusion over time.

The corresponding reverse-time SDE is:

$$\partial x = \left(f(x,t) - \sigma(t)^2 \cdot \nabla_x \log p_t(x)\right) \cdot \partial t + \sigma(t) \cdot \partial \omega,$$

where ω represents the Brownian motion in reverse time, from T to 0. This SDE illustrates that starting with pure noise, data can be recovered by counteracting the drift responsible for the data destruction through the subtraction of $\sigma(t)^2 \cdot \nabla_x \log p_t(x)$.

The neural network $s_\theta(x,t) \approx \nabla_x \log p_t(x)$ is trained by optimizing the continuous case equivalent of the objective:

$$L^*_{\text{dsm}} = \mathbb{E}_t \left[\lambda(t) \mathbb{E}_{p(x_0)} \mathbb{E}_{p_t(x_t|x_0)} \left[\|s_\theta(x_t,t) - \nabla_{x_t} \log p_t(x_t|x_0)\|_2^2\right]\right],$$

where λ is a weighting function, and $t \sim \mathcal{U}([0,T])$. It is noted that when f is affine, $p_t(x_t|x_0)$ is Gaussian. For non-affine f, alternatives like sliced score matching [SGSE20] are used.