Score-Based Diffusion Models via Stochastic Differential Equations

Arsène CLAUSTRE

September 17, 2025

Abstract

This research report investigates score-based generative models from the perspective of stochastic differential equations (SDEs), closely following the theoretical framework introduced by Tang and Zhao [22]. We present a detailed study of the key concepts, such as time-reversal of SDEs, score estimation, score matching techniques, and convergence guarantees in total variation and Wasserstein distances, including explicit derivations and simplified explanations of the underlying mathematics. To complement the theoretical analysis, we also provide numerical experiments on generated Swiss Roll data and the MNIST dataset, which illustrate the impact of SDE design and network architecture on generation quality. This report was written as part of a research internship under the supervision of Yating Liu.

Keywords: Score-based generative models, diffusion models, stochastic differential equations, reverse-time SDEs, denoising, sampling algorithms, total variation, Wasserstein distance.

Contents

1 Introduction

Diffusion models have recently become one of the most popular methods in generative modeling, achieving excellent results in tasks like high-quality image generation [4, 19] and audio synthesis [15]. The main idea is to gradually add noise to data, then learn how to reverse this process to produce new, realistic samples. This provides a stable and effective way to model complex data distributions.

More broadly, diffusion models are part of a large family of generative methods that includes Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Energy-Based Models (EBMs). GANs are known for creating sharp images but can be unstable, often suffering from training difficulties [7]. VVAEs are generally more stable but often generate blurrier samples, a limitation linked to the variational approximation [13, 16]. EBMs provide a flexible framework based on energy functions but are generally harder to train effectively [17]. Diffusion models stand out by offering a good balance between sample quality, diversity, and training stability.

Diffusion models were first introduced using discrete steps [20, 10], where data is corrupted step by step and then recovered. More recently, Song et al. [21] introduced a continuous version based on stochastic differential equations (SDEs), which provides a more flexible and unified view of the process. This continuous version allows the use of tools from stochastic calculus and helps unify different generative models into a single framework.

An essential step is to learn the score function, the gradient of the log-density of noisy data. This is typically done with score matching methods [11, 23], which have been shown effective in practice [21]. Diffusion models have gained popularity not only for their empirical performance but also for their strong theoretical foundations, which make them attractive for both practical applications and mathematical analysis.

In this report, we study diffusion models from the SDE-based perspective, mainly following the framework introduced by Tang and Zhao [22]. We begin in Section 2 by introducing the general formulation of diffusion models, focusing on forward and backward processes, and the time-reversal formula. Section 3 presents several examples of stochastic differential equations (SDEs) used in diffusion models, including the Ornstein-Uhlenbeck process, Variance Exploding (VE), Variance Preserving (VP), sub-Variance Preserving (subVP), and CDPM processes. In Section 4, we cover score matching techniques, detailing standard score matching, implicit score matching, sliced score matching, and denoising score matching. Section 5 presents theoretical results on the convergence of sampling processes, including bounds in total variation and Wasserstein distance. It also briefly discusses practical methods, with a focus on the predictor-corrector scheme. In Section 6, we move to numerical experiments, describing our methodology, initial trials, model improvements, and final tests. Lastly, Section 7 concludes the report with a summary of the main insights and a discussion of future research directions. Additional mathematical background and full proofs of some results are provided in Appendices A and B for completeness.

Our main contributions. As mentioned above, this report follows the general framework proposed by Tang and Zhao [22]. It provides more detailed and pedagogical proofs for several theoretical results, with a particular focus on the convergence bounds in total variation and Wasserstein distance. When appropriate, these results and their derivations have been slightly refined or adjusted to improve clarity and precision.

Notations

We summarize below the main notations used throughout this work.

- For x, y two vectors, we use $x \cdot y$ or equivalently $\langle x, y \rangle$ to denote their inner product. The Euclidean (or L^2) norm of x is written $\|x\|$, and the infinity norm is $\|x\|_\infty := \max_i |x_i|$.
- For any matrix (or vector) A, we write $A^\top$ for its transpose, $\mathrm{Tr}(A)$ for its trace (if A is square), and $\det(A)$ for its determinant (if A is square).
- For a vector-valued function f, ∇f denotes the gradient of f, $\nabla \cdot f = \mathrm{div}(f)$ is the divergence, $\nabla^2 f$ the Hessian matrix, and Δf the Laplacian. For $f = (f_1, \ldots, f_n)$ a matrix-valued function, with f_i its i-th column, we have $(\nabla f)_i^\top = \nabla f_i$ (Jacobian matrix) and $(\nabla \cdot f)_i = \mathrm{div}(f_i)$ for $i = 1, \ldots, n$.
- We write $a = \mathcal{O}(b)$ if a/b stays bounded, and $a = o(b)$ if a/b goes to zero when a variable tends to 0 or ∞.

- For a random variable X, $\mathcal{L}(X)$ denotes its probability law. The notation $X \sim p(\cdot)$ means that X has density $p(\cdot)$.
- $\mathcal{N}(\mu, \Sigma)$ denotes the Gaussian distribution with mean μ and covariance matrix Σ.
- For any random variable X, $\mathbb{E}[X]$ and $\mathrm{Var}(X)$ respectively denote its expectation and variance. When considering a sample $(X_1, \ldots, X_n)$ of independent and identically distributed copies of X, we use $\widehat{\mathbb{E}}X = \frac{1}{n}\sum_{i=1}^{n} X_i$ to denote the empirical mean, and $\widehat{\mathrm{Var}}(X)$ the empirical variance.
- We denote by $\mathcal{L}_\theta$ a general loss functional depending on parameters θ, and by $\widehat{\mathcal{L}}_\theta$ its empirical counterpart estimated over a finite sample.
- For two probability measures P and Q:
  - $d_{\mathrm{TV}}(P, Q) = \sup_A |P(A) - Q(A)|$ is the total variation distance between P and Q;
  - $\mathrm{KL}(P\|Q) = \int \log\left(\frac{dP}{dQ}\right) dP$ is the Kullback-Leibler divergence;
  - $W_2(P, Q) = \left(\inf_{\pi \in \Pi(P,Q)} \mathbb{E}_{(X,Y)\sim\pi}\|X - Y\|^2\right)^{1/2}$ is the Wasserstein-2 distance, where $\Pi(P, Q)$ is the set of couplings between P and Q.

2 Diffusion Models: General Framework

2.1 Forward and Backward Processes

The core idea behind diffusion models is to gradually transform a complex data distribution $p_{\mathrm{data}}(\cdot)$ into a simpler one by adding noise over time. This is called the *forward process*. Then, the goal is to learn how to reverse this transformation and go back from noise to data. This defines the *backward process*, which is at the heart of the generative model.

In this section, we introduce both processes. We start by defining the forward process as a stochastic differential equation (SDE), and then describe how the time-reversed process can be used to generate new samples. This forms the foundation of the score-based generative modeling framework.

2.1.1 Forward Process

Let $(X_t)_{t\in[0,T]} \subset \mathbb{R}^d$ be a continuous-time stochastic process defined by the stochastic differential equation (SDE):

$$dX_t = f(t, X_t)\, dt + g(t, X_t)\, dB_t, \quad X_0 \sim p_{\mathrm{data}}(\cdot), \tag{1}$$

where:

- $(B_t)_{t\geq 0}$ is an n-dimensional standard Brownian motion;
- $f : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is the drift, describing the deterministic part of the dynamics;
- $g : [0, T] \times \mathbb{R}^d \to \mathbb{R}^{d\times n}$ is the diffusion coefficient, which controls how the noise is added.

This equation describes how a data point X_0 is gradually perturbed by noise over time. The functions f and g define how the process behaves, and their choice plays a key role in the model.

Under standard assumptions, such as global Lipschitz continuity in x uniformly in t and linear growth, the SDE (1) admits a unique solution. General reminders about stochastic differential

equations, including conditions under which they have well-defined solutions, are provided in Appendix A.1. We denote by $p(t, \cdot)$ the distribution of X_t at time t. It starts from the data distribution $p_{\text{data}}(\cdot)$ and gradually evolves over time as noise is added.

As time increases, the samples become increasingly noisy, and the distribution $p(t, \cdot)$ moves closer to a simple reference distribution, such as a standard Gaussian. This final distribution is easy to sample from and will be used as the starting point of the generative process.

2.1.2 Backward Process

In generative modeling with diffusion processes, the goal is to reverse the effect of the noise added by the forward process. We would like to start from a sample of a simple distribution, such as a standard Gaussian, and transform it into a sample that resembles the original data.

To formalize this idea, we define the time-reversed process $Y_t := X_{T-t}$. Under appropriate regularity assumptions, this process satisfies its own stochastic differential equation:

$$dY_t = \bar{f}(t, Y_t)\, dt + \bar{g}(t, Y_t)\, d\bar{B}_t, \quad Y_0 \sim p(T, \cdot), \tag{2}$$

where $\bar{f}$ and $\bar{g}$ are the drift and diffusion terms of the reversed process, and $\bar{B}_t$ is a Brownian motion adapted to the reverse-time filtration.

In theory, simulating this process starting from $p(T, \cdot)$ would let us recover samples from the original data distribution $p_{\text{data}}(\cdot)$. However, this approach is not practical for two main reasons:

- The distribution $p(T, \cdot)$ is generally unknown, as it depends on the unknown data distribution p_{data}.
- We aim to define a generative model that does not require simulating the forward process explicitly. Instead of using the unknown distribution $p(T, \cdot)$, we choose a fixed and simple distribution p_{noise} as a substitute. This distribution is selected to be close to $p(T, \cdot)$ given the dynamics of the chosen forward process.

With this approximation, the reverse process is then defined by:

$$dY_t = \bar{f}(t, Y_t)\, dt + \bar{g}(t, Y_t)\, d\bar{B}_t, \quad Y_0 \sim p_{\text{noise}}(\cdot). \tag{3}$$

This approximation raises several important questions: How should we define the drift term $\bar{f}$ in order to generate realistic samples? How is it related to the original functions f and g? In particular, while the diffusion term $\bar{g}$ often coincides with the forward diffusion g, the drift $\bar{f}$ requires a specific correction that involves the score function. Finally, can these terms be computed exactly, or only estimated from data?

2.2 Time Reversal Formula

To answer these questions, we rely on a fundamental result in the theory of stochastic processes, known as the *time-reversal formula*. Under suitable assumptions, the time-reversed process of a diffusion also satisfies a stochastic differential equation. The drift of this reverse SDE can be expressed in terms of the original drift and diffusion coefficient functions, together with the density of the process at each time. This result plays a central role in the mathematical formulation of score-based generative models.

Theorem 2.1 (Time reversal formula)**.** *Assume that the drift and diffusion coefficients $f : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $g : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times n}$ are globally Lipschitz and satisfy a linear growth condition. Then the stochastic differential equation*

$$dX_t = f(t, X_t)\, dt + g(t, X_t)\, dB_t, \quad X_0 \sim p_{data}(\cdot),$$

admits a unique strong solution $(X_t)_{t\in[0,T]}$. Suppose in addition that:

- *The diffusion matrix $a(t,x) := g(t,x)g(t,x)^\top$ is uniformly elliptic, i.e., there exists $\lambda > 0$ such that*
$$\xi^\top a(t,x)\xi \geq \lambda\|\xi\|^2 \quad \text{for all } \xi \in \mathbb{R}^d;$$
- *For each $t \in [0,T]$, the law of X_t admits a smooth density $p(t,x)$ with bounded derivatives, and*
$$p(t,\cdot) \in H^1(\mathbb{R}^d).$$

Then the time-reversed process $Y_t := X_{T-t}$ satisfies the SDE:

$$dY_t = (-f(T-t, Y_t) + \nabla \cdot a(T-t, Y_t) + a(T-t, Y_t)\nabla \log p(T-t, Y_t))\, dt + g(T-t, Y_t)\, d\bar{B}_t,$$

where $\bar{B}_t$ is a Brownian motion adapted to the reversed filtration.

Remark 2.2. *Here $H^1(\mathbb{R}^d)$ denotes the first-order Sobolev space, i.e., the set of square-integrable functions whose weak derivatives are also square-integrable.*

A detailed proof of this result can be found in the work of Haussmann and Pardoux [8].

According to Theorem 2.1, and under the stated conditions, the time-reversed process $Y_t := X_{T-t}$ satisfies the following stochastic differential equation:

$$\begin{aligned} dY_t = \Big(&- f(T-t, Y_t) + a(T-t, Y_t)\, \nabla \log p(T-t, Y_t) \\ &+ \nabla \cdot a(T-t, Y_t) \Big)\, dt + g(T-t, Y_t)\, d\bar{B}_t, \quad Y_0 \sim p_{\text{noise}}(\cdot). \end{aligned} \tag{4}$$

In this equation, all terms are known explicitly except for the quantity $\nabla \log p(t,x)$, called the *score function*. It represents the gradient of the log-density of the forward process at time t and must be estimated from data. Approximating this term is essential to designing effective generative model sand will be discussed in Section 4.

The functions f and g, which define the forward SDE, are chosen in advance. They determine how noise is added to the process over time, which directly influences the quality and stability of the generative model.

We now provide several examples of how f and g can be chosen in practice.

3 Standard SDE Formulations in Diffusion Models

We now present some common examples of forward SDEs used in diffusion models. Each SDE is defined by a specific choice of drift $f(t,x)$ and diffusion coefficient $g(t,x)$, which together control how the data is gradually transformed over time.

In most cases, the SDEs used in practice follow the structure:

$$n = d, \quad g(t,x) = g(t)I,$$

which means the Brownian motion has the same dimension as the data, and the noise added is the same in all directions, depending only on time.

This assumption simplifies both the mathematical analysis and the implementation. In particular, since the noise does not depend on the current value x, it becomes easier to define a fixed target distribution $p_{\text{noise}}(\cdot)$ at final time T.

We now review these standard SDEs in more detail, discussing both the motivations behind their design and their practical role in score-based generative modeling.

3.1 Ornstein–Uhlenbeck (OU) Process

The Ornstein–Uhlenbeck process is a classical example of a mean-reverting diffusion. It is defined by:

$$f(t, x) = \theta(\mu - x), \quad g(t) = \sigma,$$

with constants $\theta > 0$, $\sigma > 0$ and $\mu \in \mathbb{R}^d$.

This gives the forward SDE:

$$dX_t = \theta(\mu - X_t)\, dt + \sigma\, dB_t, \quad X_0 = x. \tag{5}$$

By applying Itô's formula to the process $Z_t := e^{\theta t} X_t$, one can derive:

$$X_t = \mu + (x - \mu)e^{-\theta t} + \sigma \int_0^t e^{-\theta(t-s)} dB_s,$$

which implies:

$$X_t \sim \mathcal{N}\left(\mu + (x - \mu)e^{-\theta t},\ \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t})I\right).$$

As $t \to \infty$, the process converges to the stationary distribution

$$\mathcal{N}\left(\mu,\ \frac{\sigma^2}{2\theta} I\right),$$

which provides a natural choice for $p_{\text{noise}}(\cdot)$.

The time-reversed process then follows:

$$dY_t = \big(\theta(Y_t - \mu) + \sigma^2 \nabla \log p(T - t, Y_t)\big)\, dt + \sigma\, d\bar{B}_t, \quad Y_0 \sim p_{\text{noise}}(\cdot).$$

This model is well-suited for score-based generative methods. It provides explicit Gaussian marginal distributions at every time step, and the parameter θ offers direct control over the convergence speed toward the stationary distribution. These features make both the theoretical analysis and the practical implementation easier.

3.2 Variance Exploding (VE) SDE

The Variance Exploding SDE appears as the continuous-time limit of certain discrete score-based methods, such as Score Matching with Langevin Dynamics (SMLD). It is defined by a zero drift and a time-dependent diffusion coefficient that increases over time:

$$dX_t = g(t)\, dB_t, \quad X_0 = x,$$

with $g(t) = \sqrt{\frac{d\sigma^2(t)}{dt}}$, where $\sigma(t)$ is a strictly increasing function. A common choice is

$$\sigma(t) = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{t/T},$$

leading to

$$g(t) = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{t/T} \sqrt{\frac{2}{T} \log \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)}.$$

Given $X_0 = x$, the process admits a closed-form Gaussian solution:

$$X_t \sim \mathcal{N} \left(x, \int_0^t g^2(s)\, ds \cdot I \right) = \mathcal{N} \left(x, \sigma_{\min}^2 \left(\left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^{\frac{2t}{T}} - 1 \right) I \right).$$

As time progresses, the variance increases rapidly, which motivates the name "variance exploding". Although the process does not converge to a stationary distribution, it reaches a Gaussian at time T, making it natural to define the noise distribution as

$$p_{\text{noise}}(\cdot) = \mathcal{N}(0, (\sigma_{\max}^2 - \sigma_{\min}^2) I).$$

The reverse process then satisfies the SDE

$$dY_t = g^2(T-t) \nabla \log p(T-t, Y_t)\, dt + g(T-t)\, d\bar{B}_t, \quad Y_0 \sim p_{\text{noise}}(\cdot).$$

The VE SDE is popular in practice due to its simplicity, its closed-form sampling, and its ability to strongly corrupt the data. As in other models, the only term that needs to be estimated is the score function $\nabla \log p(t, x)$.

3.3 Variance Preserving (VP) SDE

The variance preserving SDE corresponds to the continuous-time limit of denoising diffusion probabilistic models (DDPM). Unlike the VE process, where the variance increases unboundedly, this formulation ensures the variance remains controlled throughout the forward trajectory.

The SDE takes the form

$$dX_t = -\frac{1}{2} \beta(t) X_t\, dt + \sqrt{\beta(t)}\, dB_t, \quad X_0 = x,$$

where $\beta(t)$ is a positive and typically increasing function over time. A common choice is

$$\beta(t) = \beta_{\min} + \frac{t}{T}(\beta_{\max} - \beta_{\min}), \tag{6}$$

with $\beta_{\min} \ll \beta_{\max}$, allowing gradual and smooth corruption of the initial data.

Given $X_0 = x$, the marginal distribution of the process admits an explicit closed form:

$$X_t \sim \mathcal{N} \left(e^{-\frac{1}{2} \int_0^t \beta(s)\, ds}\, x,\ \left(1 - e^{-\int_0^t \beta(s)\, ds} \right) I \right). \tag{7}$$

Justification of the marginal law. To derive this result, consider the process

$$Z_t := e^{\frac{1}{2}\int_0^t \beta(s)\,ds} X_t.$$

Using Itô's formula A.2, we compute

$$dZ_t = e^{\frac{1}{2}\int_0^t \beta(s)\,ds}\left(\frac{1}{2}\beta(t)X_t\,dt + dX_t\right).$$

Substituting the SDE for dX_t, we obtain

$$dZ_t = e^{\frac{1}{2}\int_0^t \beta(s)\,ds}\left(\frac{1}{2}\beta(t)X_t\,dt - \frac{1}{2}\beta(t)X_t\,dt + \sqrt{\beta(t)}\,dB_t\right) = e^{\frac{1}{2}\int_0^t \beta(s)\,ds}\sqrt{\beta(t)}\,dB_t.$$

Integrating, we find

$$Z_t = x + \int_0^t e^{\frac{1}{2}\int_0^s \beta(u)\,du}\sqrt{\beta(s)}\,dB_s,$$

which is a Gaussian process. Its mean and covariance are given by

$$\mathbb{E}[Z_t] = x, \quad \mathrm{Cov}(Z_t) = \int_0^t e^{\int_0^s \beta(u)\,du}\beta(s)\,ds \cdot I.$$

Recovering $X_t = e^{-\frac{1}{2}\int_0^t \beta(s)\,ds} Z_t$, we deduce

$$\mathbb{E}[X_t] = e^{-\frac{1}{2}\int_0^t \beta(s)\,ds} x, \quad \mathrm{Cov}(X_t) = \left(1 - e^{-\int_0^t \beta(s)\,ds}\right) I.$$

Therefore,

$$X_t \sim \mathcal{N}\left(e^{-\frac{1}{2}\int_0^t \beta(s)\,ds} x,\ \left(1 - e^{-\int_0^t \beta(s)\,ds}\right) I\right).$$

Interpretation. As $t \to T$, the exponential terms become close to zero if T or $\beta_{\max}$ is large enough. In this case, the process X_T becomes approximately Gaussian with zero mean and identity covariance:

$$X_T \approx \mathcal{N}(0, I).$$

It is therefore natural to define the noise distribution as:

$$p_{\text{noise}} = \mathcal{N}(0, I),$$

which will be used to initialize the backward process.

Backward process. By applying the time reversal formula, the reverse process follows the SDE:

$$dY_t = \left(\frac{1}{2}\beta(T-t)Y_t + \beta(T-t)\nabla \log p(T-t, Y_t)\right)dt + \sqrt{\beta(T-t)}\,d\bar{B}_t, \quad Y_0 \sim \mathcal{N}(0, I). \tag{8}$$

Overall, the VP model defines a more controlled diffusion process, with smooth dynamics and bounded variance at every step. This makes training and sampling more stable, especially when the data is concentrated in a limited region of space.

3.4 Sub-Variance Preserving (subVP) SDE

The subVP SDE is a slight modification of the VP model, designed to inject slightly less noise over time. The forward equation is:

$$dX_t = -\frac{1}{2}\beta(t)\, X_t\, dt + \sqrt{\beta(t)(1-\gamma(t))}\, dB_t, \quad X_0 = x,$$

where $\beta(t)$ is the same as in the VP model, and $\gamma(t) = \exp\left(-2\int_0^t \beta(s)\, ds\right)$.

To derive the distribution of X_t, we follow the same steps as in the VP case. The process remains Gaussian with mean and variance:

$$X_t \sim \mathcal{N}\left(e^{-\frac{1}{2}\int_0^t \beta(s)\, ds}\, x,\ \left(1 - e^{-\int_0^t \beta(s)\, ds}\right)^2 I\right).$$

Using the expression of $\beta(t)$ from Equation (6), we obtain:

$$X_t \sim \mathcal{N}\left(e^{-\frac{t^2}{4T}(\beta_{\max}-\beta_{\min})-\frac{t}{2}\beta_{\min}}\, x,\ \left(1 - e^{-\frac{t^2}{2T}(\beta_{\max}-\beta_{\min})-t\beta_{\min}}\right)^2 I\right).$$

This version of the model injects slightly less noise at each step of the diffusion process. As a result, the variance increases more slowly, and the process stays closer to the initial data for longer. This makes the reverse process easier to control and can lead to better sample quality.

The reverse SDE becomes:

$$dY_t = \left(\frac{1}{2}\beta(T-t)\, Y_t + \beta(T-t)(1-\gamma(T-t))\nabla \log p(T-t, Y_t)\right) dt + \sqrt{\beta(T-t)(1-\gamma(T-t))}\, d\bar{B}_t,$$

with $Y_0 \sim \mathcal{N}(0, I)$.

Since the noise grows more slowly, this model tends to be more stable and can produce higher quality samples.

3.5 Contractive Diffusion Probabilistic Models (CDPM)

Contractive probabilistic diffusion models are designed to improve the stability of the reverse process, especially when the estimated score $\nabla \log p(t, x)$ is imperfect. Since sampling relies on this approximation, small errors in the score can cause the trajectory to deviate from the data distribution.

The idea is to make the reverse SDE more robust, so that nearby points at the final time remain close when propagated backward. This limits the effect of score errors and stabilizes the sampling.

To obtain this behavior, one requires that the forward drift $f(t, x)$ satisfies a **monotonicity condition**:

$$\langle x - x',\ f(t,x) - f(t,x')\rangle \geq r_f(t)\|x - x'\|^2,$$

for all $x, x' \in \mathbb{R}^d$ and all $t \in [0, T]$, with $\inf_{t\in[0,T]} r_f(t) > 0$. This ensures that the forward drift is regular enough for the reverse drift

$$\bar{f}(t, x) = -f(T-t, x) + \text{correction terms}$$

to be contractive, meaning it brings nearby points closer together.

This helps improve the robustness of the reverse dynamics, but can also make the forward process more constrained. In practice, one must carefully tune the time horizon T and the noise schedule to balance stability and flexibility.

Examples. Contractive OU (COU). This model is inspired by the Ornstein–Uhlenbeck process, but uses the opposite drift direction. The forward SDE is defined by:

$$f(t,x) = \theta(x-\mu), \quad g(t) = \sigma > 0.$$

Here $\theta > 0$, $\mu \in \mathbb{R}^d$, and $\sigma \in \mathbb{R}_+$. In this case, the forward process leads to the choice of the following Gaussian distribution for the noise:

$$p_{\text{noise}}(\cdot) = \mathcal{N}\left(0,\ \frac{\sigma^2}{2\theta}(e^{2\theta T}-1)I\right).$$

The reverse process then takes the form:

$$dY_t = \left(-\theta(Y_t-\mu) + \sigma^2 \nabla \log p(T-t, Y_t)\right) dt + \sigma d\bar{B}_t,$$

with initial condition $Y_0 \sim p_{\text{noise}}(\cdot)$.

Contractive VP (CVP). This variant is inspired by the variance preserving (VP) model and also uses the opposite drift direction. The forward SDE is:

$$f(t,x) = \frac{1}{2}\beta(t)\, x, \quad g(t) = \sqrt{\beta(t)},$$

with $\beta(t)$ defined as in the VP model. The resulting choice for the noise distribution at time T is:

$$p_{\text{noise}}(\cdot) = \mathcal{N}\left(0, \left(e^{\frac{T}{2}(\beta_{\max}+\beta_{\min})} - 1\right) I\right).$$

The reverse process is defined by:

$$dY_t = \left(-\frac{1}{2}\beta(T-t)Y_t + \beta(T-t)\nabla \log p(T-t, Y_t)\right) dt + \sqrt{\beta(T-t)}\, d\bar{B}_t,$$

with initial condition $Y_0 \sim p_{\text{noise}}(\cdot)$.

4 Score Matching Techniques

4.1 Overview

The central difficulty in implementing the reverse-time SDE of diffusion models lies in the estimation of the score function $\nabla \log p(t,x)$. This function governs the drift of the backward process and has a direct impact on the quality of the generated samples. However, the marginal density $p(t,x)$ is unknown in practice and only accessible through forward simulations. We must therefore rely on an approximation $s_\theta(t,x)$, typically implemented as a parameterized model such as a neural network. The general strategy for learning s_θ is called *score matching*.

Let us consider a forward process X_t defined by a SDE and its corresponding backward SDE:

$$dY_t = (-f(T-t,Y_t) + a(T-t,Y_t), \nabla \log p(T-t,Y_t) + \nabla \cdot a(T-t,Y_t))\, dt + g(T-t,Y_t), d\bar{B}t, \quad (9)$$

we replace the unknown score $\nabla \log p$ by its approximation s_θ. We obtain the following approximate dynamics:

$$dY_t = (-f(T-t,Y_t) + a(T-t,Y_t)\, s_\theta(T-t,Y_t) + \nabla \cdot a(T-t,Y_t))\, dt + g(T-t,Y_t)\, d\bar{B}_t. \quad (10)$$

In the particular case where $g(t,x) = g(t)I$, the equation simplifies to:

$$dY_t = \big(-f(T-t, Y_t) + g^2(T-t)\, s_\theta(T-t, Y_t)\big)\, dt + g(T-t)\, d\bar{B}_t. \tag{11}$$

To generate realistic samples, the approximation $s_\theta(t,x)$ should be as close as possible to the true score $\nabla \log p(t,x)$. The natural idea would be to minimize the following loss:

$$\mathcal{L}_{\mathrm{ESM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T),\ X\sim p(t,\cdot)} \left[\lambda(t)\, \|s_\theta(t,X) - \nabla \log p(t,X)\|^2\right], \tag{12}$$

where $\lambda(t)$ is a positive weight function that can emphasize certain time intervals during training. For instance, it may give more importance to time steps where the score is harder to estimate or where generation quality is more sensitive to errors.

However, this objective cannot be used directly, since the true score $\nabla \log p(t,x)$ is not available. The key idea is to rewrite the loss in an equivalent form that does not require access to the true score. These reformulations are known as *implicit score matching objectives*, and they form the basis of practical training strategies for score-based models.

Notation. In the rest of this section, we use the following names for different score matching losses:

- $\mathcal{L}_{\mathrm{ESM}}(\theta)$: explicit score matching loss,
- $\mathcal{L}_{\mathrm{ISM}}(\theta)$: implicit score matching loss,
- $\mathcal{L}_{\mathrm{SSM}}(\theta)$: sliced score matching loss,
- $\mathcal{L}_{\mathrm{DSM}}(\theta)$: denoising score matching loss.

We first show how to rewrite the loss in a way that no longer depends on the true score. Then, we introduce two practical versions that are easier to use in high dimensions: Sliced Score Matching (SSM) and Denoising Score Matching (DSM). These methods make it possible to train score-based models efficiently, even when the data lives in a high-dimensional space.

4.2 Implicit Score Matching

As mentioned earlier, we cannot directly minimize the explicit loss

$$\mathcal{L}_{\mathrm{ESM}}(\theta) = \mathbb{E}_{t,X} \left[\lambda(t)\, \|s_\theta(t,X) - \nabla \log p(t,X)\|^2\right]$$

because the score $\nabla \log p(t,x)$ is not available. However, this loss can be rewritten in an equivalent form that only involves the model s_θ. This gives the *implicit score matching loss*, which forms the basis of most training methods in practice.

We now state this result more precisely.

Theorem 4.1 (Implicit Score Matching)**.** *Let $p(t,x) \in C^1([0,T]\times\mathbb{R}^d)$ be a strictly positive probability density such that, for all $t \in [0,T]$,*

$$\int_{\mathbb{R}^d} \|\nabla \log p(t,x)\|^2\, p(t,x)\, dx < \infty.$$

Let $s_\theta \in C^1([0,T]\times\mathbb{R}^d;\mathbb{R}^d)$ such that :

(i) $\displaystyle\int_{\mathbb{R}^d} \|s_\theta(t,x)\|^2\, p(t,x)\, dx < \infty$ *for all $t \in [0,T]$,*

(ii) $\int_{\mathbb{R}^d} |\nabla \cdot s_\theta(t,x)|\, p(t,x)\, dx < \infty$ *for all* $t \in [0,T]$,

(iii) $s_\theta(t,x)\, p(t,x) \to 0$ *as* $\|x\| \to \infty$, *for all* $t \in [0,T]$.

Then the explicit score matching loss

$$\mathcal{L}_{ESM}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T),\, X\sim p(t,\cdot)} \left[\lambda(t)\ \|s_\theta(t,X) - \nabla \log p(t,X)\|^2\right]$$

is equivalent, up to a constant independent of θ, *to the implicit score matching loss*

$$\mathcal{L}_{ISM}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T),\, X\sim p(t,\cdot)} \left[\lambda(t)\ \left(\|s_\theta(t,X)\|^2 + 2\,\nabla \cdot s_\theta(t,X)\right)\right].$$

In particular, minimizing $\mathcal{L}_{ESM}$ *is equivalent to minimizing* $\mathcal{L}_{ISM}$.

Proof. To make the calculations easier to read, we may use the following shorthand :

$$\mathbb{E}_{t,X}[\cdot] := \mathbb{E}_{t\sim\mathcal{U}(0,T),\, X\sim p(t,\cdot)}[\cdot]$$

We develop the squared norm:

$$\|s_\theta(t,X) - \nabla \log p(t,X)\|^2 = \|s_\theta(t,X)\|^2 + \|\nabla \log p(t,X)\|^2 - 2\, s_\theta(t,X)^\top \nabla \log p(t,X).$$

We take the expectation over $t \sim \mathcal{U}(0,T)$, $X \sim p(t,\cdot)$, and multiply by $\lambda(t)$. All terms are well-defined thanks to the integrability conditions in the statement and this gives :

$$\begin{aligned}\mathbb{E}_{t,X}\left[\lambda(t)\, \|s_\theta(t,X) - \nabla \log p(t,X)\|^2\right] &= \mathbb{E}_{t,X}\left[\lambda(t)\|s_\theta(t,X)\|^2\right] \\ &\quad - 2\,\mathbb{E}_{t,X}\left[\lambda(t)\, s_\theta(t,X)^\top \nabla \log p(t,X)\right] + C\end{aligned}$$

where

$$C = \mathbb{E}_{t,X}\left[\lambda(t)\|\nabla \log p(t,X)\|^2\right]$$

is a constant that does not depend on θ.

We now simplify the second term. Using the identity $\nabla \log p = \frac{\nabla p}{p}$, we write:

$$\mathbb{E}_{X\sim p(t,\cdot)}\left[s_\theta(t,X)^\top \nabla \log p(t,X)\right] = \int s_\theta(t,x)^\top \nabla p(t,x)\, dx.$$

We also have that:

$$\int s_\theta(t,x)^\top \nabla p(t,x)\, dx = \sum_{i=1}^{d} \int s_{\theta,i}(t,x)\, \partial_i p(t,x)\, dx.$$

Using integration by parts gives:

$$\int s_{\theta,i}(t,x)\, \partial_i p(t,x)\, dx = -\int p(t,x)\, \partial_i s_{\theta,i}(t,x)\, dx,$$

since the boundary term vanishes by assumption.

Summing over i, we get:

$$\int s_\theta(t,x)^\top \nabla p(t,x)\, dx = -\int p(t,x)\, \nabla \cdot s_\theta(t,x)\, dx.$$

So:

$$\mathbb{E}_{t,X}\left[\lambda(t)\, s_\theta(t,X)^\top \nabla \log p(t,X)\right] = -\mathbb{E}_{t,X}\left[\lambda(t)\, \nabla \cdot s_\theta(t,X)\right].$$

Putting it all together:

$$\mathbb{E}_{t,X}\left[\lambda(t)\, \|s_\theta(t,X) - \nabla \log p(t,X)\|^2\right] = \mathbb{E}_{t,X}\left[\lambda(t)\left(\|s_\theta(t,X)\|^2 + 2\,\nabla \cdot s_\theta(t,X)\right)\right] + C,$$

which proves the result. □

This result shows that we do not need the true score $\nabla \log p(t,x)$ to train the model. It is enough to minimize a loss that depends only on s_θ and its divergence. The weighting function $\lambda(t)$ can be used to give more or less importance to certain time steps. For example, one may increase it where the data is more concentrated, or where learning the score is more difficult.

Remark 4.2. *One practical issue with the identity above is the computation of the divergence term $\nabla \cdot s_\theta(t,x)$. This requires summing the partial derivatives of each output component with respect to its input, which means performing d backward passes if s_θ is a neural network. This can be very costly when the dimension is high. The next subsections present two alternatives that are easier to scale: Sliced Score Matching and Denoising Score Matching.*

4.3 Sliced Score Matching

To reduce the cost of computing divergences in high dimensions, *Sliced Score Matching* (SSM) replaces the divergence term in $\mathcal{L}_{\text{ISM}}$ with an expectation over random projections. The resulting objective is:

$$\mathcal{L}_{\text{SSM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T)}\, \mathbb{E}_{X\sim p(t,\cdot)}\, \mathbb{E}_{v\sim\mathcal{N}(0,I)}\left[\lambda(t)\left(\|s_\theta(t,X)\|^2 + 2\, v^\top \nabla (v^\top s_\theta(t,X))\right)\right].$$

Theorem 4.3 (Sliced Score Matching Consistency)**.** *Assume that s_θ is sufficiently regular and satisfies the conditions of Theorem 4.1. Then:*

$$\mathcal{L}_{SSM}(\theta) = \mathcal{L}_{ESM}(\theta) + C,$$

for some constant C independent of θ. In particular, minimizing $\mathcal{L}_{SSM}$ is equivalent to minimizing the original explicit score matching loss.

Remark 4.4. *This method avoids computing all partial derivatives of the score. With neural networks, it only needs one backward pass per sample, instead of d. This makes training much faster in high dimensions.*

Proof. We aim to show that the sliced score matching loss

$$\mathcal{L}_{\text{SSM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T)}\, \mathbb{E}_{X\sim p(t,\cdot)}\, \mathbb{E}_{v\sim\mathcal{N}(0,I)}\left[\lambda(t)\left((v^\top s_\theta(t,X))^2 + 2\, v^\top \nabla (v^\top s_\theta(t,X))\right)\right]$$

is equal to the implicit score matching loss:

$$\mathcal{L}_{\text{ISM}}(\theta) = \mathbb{E}_{t,X}\left[\lambda(t)\left(\|s_\theta(t,X)\|^2 + 2\,\nabla \cdot s_\theta(t,X)\right)\right].$$

We proceed by computing the expectation over $v \sim \mathcal{N}(0,I)$.
We have :

$$v^\top \nabla (v^\top s_\theta(t,x)) = \sum_{i,j=1}^{d} v_i v_j \frac{\partial s_{\theta,j}}{\partial x_j}(t,x).$$

Taking expectation over $v \sim \mathcal{N}(0, I)$ and using the identity:

$$\mathbb{E}[v_i v_j] = \delta_{ij},$$

where δ_{ij} is the Kronecker delta gives :

$$\mathbb{E}_v \left[v^\top \nabla (v^\top s_\theta(t,x)) \right] = \sum_{i=1}^{d} \frac{\partial s_{\theta,i}}{\partial x_i}(t,x) = \nabla \cdot s_\theta(t,x).$$

Substituting this into the original expression:

$$\mathbb{E}_v \left[\|s_\theta(t,x)\|^2 + 2\, v^\top \nabla (v^\top s_\theta(t,x)) \right] = \|s_\theta(t,x)\|^2 + 2\, \nabla \cdot s_\theta(t,x),$$

we conclude that:

$$\mathcal{L}_{\text{SSM}}(\theta) = \mathcal{L}_{\text{ISM}}(\theta).$$

We conclude, by using Theorem 4.1 and the equality:

$$\mathcal{L}_{\text{ESM}}(\theta) = \text{const} + \mathcal{L}_{\text{ISM}}(\theta). \qquad \square$$

Remark 4.5. *In practice, the expectation over v is approximated by sampling m i.i.d. vectors $v_1, \ldots, v_m \sim \mathcal{N}(0, I)$, leading to the empirical loss:*

$$\widehat{\mathcal{L}}_{SSM}(\theta) = \mathbb{E}_{t,X} \left[\frac{1}{m} \sum_{i=1}^{m} \lambda(t) \left(\|s_\theta(t,X)\|^2 + 2\, v_i^\top \nabla (v_i^\top s_\theta(t,X)) \right) \right].$$

This version only requires m backward passes per sample, instead of one per input dimension. In practice, using a small number of projections, sometimes even $m = 1$, is often reported to work well.

4.4 Denoising Score Matching

A different approach to avoid using the unknown score $\nabla \log p(t,x)$ is to rewrite the loss using a conditional distribution that is sometimes easier to work with. This leads to *Denoising Score Matching* (DSM), which is based on the conditional score $\nabla \log p(t, x \mid x_0)$, where $x_0 \sim p_{\text{data}}(\cdot)$ is a clean data point, and $x_t \sim p(t, \cdot \mid x_0)$ is a noisy sample generated from it using the forward SDE.

Starting from the explicit loss,

$$\mathcal{L}_{\text{ESM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}(0,T),\ X \sim p(t,\cdot)} \left[\lambda(t) \, \|s_\theta(t,X) - \nabla \log p(t,X)\|^2 \right],$$

we can show that this is equal, up to a constant, to the denoising loss:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}(0,T)}\ \mathbb{E}_{X_0 \sim p_{\text{data}}(\cdot)}\ \mathbb{E}_{X_t \sim p(t,\cdot \mid X_0)} \left[\lambda(t) \, \|s_\theta(t,X_t) - \nabla \log p(t, X_t \mid X_0)\|^2 \right].$$

The conditional score $\nabla \log p(t, x \mid x_0)$ describes how noise spreads around each clean point. By minimizing this loss, the model learns to denoise: it predicts a vector field that points from each noisy sample x_t back toward its clean version x_0.

Theorem 4.6 (Denoising Score Matching Consistency)**.** *Assume that s_θ is sufficiently regular and satisfies the assumptions of Theorem 4.1. Then:*

$$\mathcal{L}_{DSM}(\theta) = \mathcal{L}_{ESM}(\theta) + C,$$

for some constant C independent of θ. As a result, both losses have the same minimizers.

Proof. We define the denoising loss as:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) := \mathbb{E}_{t\sim\mathcal{U}(0,T)}\ \mathbb{E}_{X_0\sim p(0,\cdot)}\ \mathbb{E}_{X_t\sim p(t,\cdot|X_0)}\left[\lambda(t)\left\|s_\theta(t,X_t)-\nabla\log p(t,X_t\mid X_0)\right\|^2\right].$$

For simplicity, we write:

$$\mathbb{E}_{t,X_0,X_t}[\,\cdot\,] := \mathbb{E}_{t\sim\mathcal{U}(0,T)}\ \mathbb{E}_{X_0\sim p(0,\cdot)}\ \mathbb{E}_{X_t\sim p(t,\cdot|X_0)}[\,\cdot\,].$$

We now compare this to the explicit loss:

$$\mathcal{L}_{\mathrm{ESM}}(\theta) = \mathbb{E}_{t,X}\left[\lambda(t)\left\|s_\theta(t,X)-\nabla\log p(t,X)\right\|^2\right].$$

We expand this:

$$\mathcal{L}_{\mathrm{ESM}}(\theta) = \mathbb{E}_{t,X}\left[\lambda(t)\|s_\theta(t,X)\|^2\right] - 2\,\mathbb{E}_{t,X}\left[\lambda(t)\,s_\theta(t,X)^\top\nabla\log p(t,X)\right] + \mathbb{E}_{t,X}\left[\lambda(t)\|\nabla\log p(t,X)\|^2\right].$$

On each term:
For the first term using $p(t,x)=\int p(0,x_0)\,p(t,x\mid x_0)\,dx_0$, we get:

$$\mathbb{E}_{t,X}\left[\lambda(t)\|s_\theta(t,X)\|^2\right] = \mathbb{E}_{t,X_0,X_t}\left[\lambda(t)\|s_\theta(t,X_t)\|^2\right].$$

We treat the second term in the same way:

$$\mathbb{E}_{t,X}\left[\lambda(t)\,s_\theta(t,X)^\top\nabla\log p(t,X)\right] = \mathbb{E}_{t,X_0,X_t}\left[\lambda(t)\,s_\theta(t,X_t)^\top\nabla\log p(t,X_t\mid X_0)\right].$$

The third term

$$\mathbb{E}_{t,X}\left[\lambda(t)\|\nabla\log p(t,X)\|^2\right]$$

does not depend on θ, so we write it as a constant C.
We now write:

$$\mathcal{L}_{\mathrm{ESM}}(\theta) = \mathbb{E}_{t,X_0,X_t}\left[\lambda(t)\left(\|s_\theta(t,X_t)\|^2 - 2\,s_\theta(t,X_t)^\top\nabla\log p(t,X_t\mid X_0)\right)\right] + C.$$

We expand the DSM loss:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t,X_0,X_t}\left[\lambda(t)\left(\|s_\theta(t,X_t)\|^2 - 2\,s_\theta(t,X_t)^\top\nabla\log p(t,X_t\mid X_0) + \|\nabla\log p(t,X_t\mid X_0)\|^2\right)\right].$$

The last term is also independent of θ, so we conclude:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathcal{L}_{\mathrm{ESM}}(\theta) + C',$$

for some constant C', which proves the result. □

Analytical expression in the Gaussian case. In many diffusion models like VP or VE, the forward process is Gaussian. For any $t\in(0,T)$, the conditional distribution of X_t given X_0 is:

$$X_t\mid X_0 \sim \mathcal{N}(\mu_t(X_0),\sigma_t^2 I),$$

with known functions μ_t and σ_t. In the VE case, $\mu_t(x)=x$ and $\sigma_t=\sigma_{\min}\left(\left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{2t/T}-1\right)^{1/2}$. In the VP case, $\mu_t(x)=e^{-\frac{1}{2}\int_0^t\beta(s)ds}x$ and $\sigma_t^2=1-e^{-\int_0^t\beta(s)ds}$.

In these cases, the conditional score has a closed form:

$$\nabla \log p(t, x \mid x_0) = \frac{\mu_t(x_0) - x}{\sigma_t^2}.$$

Plugging this into the DSM loss gives:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t,X_0,X_t}\left[\lambda(t)\left\|s_\theta(t, X_t) + \frac{X_t - \mu_t(X_0)}{\sigma_t^2}\right\|^2\right].$$

A common and useful choice is $\lambda(t) = \sigma_t^2$, which simplifies the loss to:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t,X_0,X_t}\left[\sigma_t^2\left\|s_\theta(t, X_t) + \frac{X_t - \mu_t(X_0)}{\sigma_t^2}\right\|^2\right].$$

Using the reparameterization $X_t = \mu_t(X_0) + \sigma_t\varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$, we get:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t,X_0,\varepsilon}\left[\left\|\sigma_t\, s_\theta(t, \mu_t(X_0) + \sigma_t\varepsilon) + \varepsilon\right\|^2\right].$$

This is the final objective used in practice to train diffusion models with DSM.

5 Stochastic generative processes: convergence results

In this section, we study the convergence of stochastic generative processes, that is, the backward dynamics used in diffusion models. To make this analysis possible, we assume that the estimated score function $s_\theta(t, x)$ is sufficiently close to the true score $\nabla \log p(t, x)$. We formalize this with the following assumptions:

Hypothesis 5.1 (L^2 score matching)**.** *There exists $\varepsilon > 0$ such that for all $t \in [0, T]$,*

$$\mathbb{E}_{p(t,\cdot)}\left[\left\|s_\theta(t, X) - \nabla \log p(t, X)\right\|^2\right] \leq \varepsilon^2.$$

This corresponds to an L^2-bound on score matching. A stronger but more restrictive requirement is the following:

Hypothesis 5.2 (L^∞ score matching)**.** *There exists $\varepsilon > 0$ such that for all $t \in [0, T]$,*

$$\left\|s_\theta(t, \cdot) - \nabla \log p(t, \cdot)\right\|_\infty \leq \varepsilon.$$

We will base our convergence results on these two hypotheses.

The performance of a diffusion model depends on both the accuracy of the score estimate and the structure of the forward and backward processes. Since the generative process is continuous in time, we can use tools from stochastic analysis to study how well it recovers the data distribution.

We focus on how the backward process Y_T approximates the target distribution p_{data}. For this, we use two standard distances:

- the total variation distance,
- the Wasserstein-2 distance.

While total variation (or KL divergence) is often used in theory, the Wasserstein distance is especially relevant in practice, as it aligns better with perceptual quality in image generation. For instance, metrics like the Fréchet Inception Distance (FID) are based on it, where the Wasserstein-2 distance is computed between Gaussian approximations of the feature embeddings of real and generated images [9].

In the rest of this section, we assume that $g(t,x) = g(t)I$ and that the dimension $n = d$. We begin by studying the total variation bound, followed by the Wasserstein bound.

5.1 Total variation bound

Our goal in this subsection is to study how close the final distribution of the generative process Y_T is to the target data distribution $p_{\text{data}}(\cdot)$. A natural way to quantify this is through the total variation distance $d_{\text{TV}}\left(\mathcal{L}(Y_T), p_{\text{data}}(\cdot)\right)$.

Theorem 5.3 (Total variation bound)**.** *Let Hypothesis 5.1 hold, and assume that $g(t)$ is bounded on $[0,T]$. Let Q_T'' denote the law on path space $\mathcal{C}([0,T],\mathbb{R}^d)$ of the process $(Y_t)_{t\in[0,T]}$ satisfying*

$$\mathrm{d}Y_t = -\left(f(T-t,Y_t) + g^2(T-t)\nabla \log p(T-t,Y_t)\right)\,\mathrm{d}t + g(T-t)\,\mathrm{d}B_t, \quad Y_0 \sim p(T,\cdot).$$

Assume moreover that the following Novikov condition holds:

$$\mathbb{E}_{Q_T''}\left[\exp\left(\frac{1}{2}\int_0^T \left\|g(T-t)\left(s_\theta(T-t,Y_t) - \nabla \log p(T-t,Y_t)\right)\right\|^2 \,\mathrm{d}t\right)\right] < \infty.$$

Then the total variation distance between the final law of the generative process Y_T and the data distribution satisfies

$$d_{\text{TV}}\left(\mathcal{L}(Y_T), p_{data}(\cdot)\right) \le d_{\text{TV}}\left(p(T,\cdot), p_{noise}(\cdot)\right) + \varepsilon\sqrt{\frac{T}{2}}\cdot\|g\|_\infty.$$

The proof is provided in Appendix B.1.

Remark 5.4. *The proof relies on the boundedness of the function g, through the term $\|g\|_\infty$ in the final bound. This assumption is reasonable in practice, as g is typically bounded in most models discussed earlier. However, in variance exploding (VE) models where $g(t) = \sqrt{\beta(t)}$, a large value of $\beta_{\max}$ may lead to a looser bound.*

Theorem 5.3 shows that the total variation distance between the final sample Y_T and the data distribution $p_{\text{data}}(\cdot)$ can be controlled by two terms. In some settings, such as the Variance Preserving (VP) SDE, the first term $\mathrm{d}_{\text{TV}}(p(T,\cdot), p_{\text{noise}}(\cdot))$ admits an explicit bound. This leads to the following corollary.

Corollary 5.5 (Explicit total variation bound for VP SDE)**.** *Assume that the forward process is a Variance Preserving SDE as defined in Equation* (8)*, and that the conditions of Theorem 5.3 hold. Moreover, assume that $\beta(t)$ is bounded away from zero, and that $\mathbb{E}_{p_{data}(\cdot)}[\|X_0\|^2] < \infty$. Then, for T sufficiently large,*

$$\mathrm{d}_{\text{TV}}\left(\mathcal{L}(Y_T), p_{data}(\cdot)\right) \le \exp\left(-\frac{1}{2}\int_0^T \beta(s)\,\mathrm{d}s\right)\sqrt{\frac{1}{2}\mathbb{E}_{p_{data}(\cdot)}[\|X_0\|^2]} + \varepsilon\|g\|_\infty\sqrt{\frac{T}{2}}.$$

The proof of this corollary is provided in Appendix B.2.

5.2 Wasserstein bound

We now provide a convergence result in the Wasserstein distance between $\mathcal{L}(Y_T)$ and $p_{\text{data}}(\cdot)$. Compared to the total variation setting, we require stronger regularity assumptions on the drift and score functions

Hypothesis 5.6. *We assume:*

1. *There exists a function $r_f : [0,T] \to \mathbb{R}$ such that for all $x, x' \in \mathbb{R}^d$ and $t \in [0,T]$,*

$$\langle x - x', f(t,x) - f(t,x') \rangle \geq r_f(t)\|x - x'\|^2.$$

2. *There exists a constant $L > 0$ such that for all $x, x' \in \mathbb{R}^d$ and $t \in [0,T]$,*

$$\left\|\nabla \log p(t,x) - \nabla \log p(t,x')\right\| \leq L\|x - x'\|.$$

Under these assumptions, we obtain the following bound:

Theorem 5.7 (Wasserstein bound)**.** *Let Hypotheses 5.6 and 5.2 hold. define*

$$u(t) := \int_{T-t}^{T} \left(-2r_f(s) + (2L + 2h)g^2(s)\right) \mathrm{d}s,$$

We have

$$W_2(p_{data}(\cdot), \mathcal{L}(Y_T)) \leq \sqrt{W_2^2(p(T,\cdot), p_{noise}(\cdot))\, e^{u(T)} + \frac{\varepsilon^2}{2h} \int_0^T g^2(t) e^{u(T)-u(T-t)} \mathrm{d}t}.$$

The proof is given in Appendix B.3.

Remark 5.8. *Theorem 5.7 assumes a uniform (L^∞) bound on the score matching error, as stated in Hypothesis 5.2. A similar result can also be obtained if we instead assume an L^2-bound, together with the following Lipschitz condition on the score estimator:*

$$\|s_\theta(t,x) - s_\theta(t,x')\| \leq L\|x - x'\| \quad \textit{for all } x, x'.$$

This condition is weaker, however, it may be too strong in practice, especially for a neural network.

An alternative is to directly bound the expected score error:

$$\mathbb{E}\left[\|s_\theta(T-t, V_t) - \nabla \log p(T-t, V_t)\|^2\right] \leq \varepsilon^2 \quad \textit{for all } t.$$

which leads to the same conclusion.

Finally, Theorem 5.7 does not require any specific form for the drift $f(t,x)$. The bound may be less effective when $r_f(t)$ is negative, because the exponential term grows with T. In some cases, more precise estimates can be obtained, as we will see in the next examples.

We now present more precise estimates under the additional assumption that the data distribution is strongly log-concave.

Definition 5.9 (κ-strong log-concavity)**.** *A smooth function $h : \mathbb{R}^d \to \mathbb{R}$ is said to be κ-strongly log-concave if*

$$(\nabla \log h(x) - \nabla \log h(y)) \cdot (x - y) \leq -\kappa\|x - y\|^2.$$

This assumption is satisfied, for example, when $p_{\text{data}}(\cdot)$ is Gaussian or, more generally, when it has a density of the form $p(x) \propto \exp(-V(x))$ with V a strongly convex function. While this is restrictive and rarely true for real-world data, it provides a useful theoretical setting in which sharper convergence guarantees can be established.

Corollary 5.10 (Wasserstein bound for VP)**.** *Assume that $p_{data}(\cdot)$ is κ-strongly log-concave for some $\kappa > \frac{1}{2}$, and that $\mathbb{E}_{p_{data}(\cdot)}[\|X_0\|^2] < \infty$. Then, for any $h < \min(\kappa, 1) - \frac{1}{2}$, we have*

$$W_2^2(p_{data}(\cdot), \mathcal{L}(Y_T)) \le e^{-\{\beta_{\min} + \beta_{\max}(2\min(\kappa,1)-1-2h)\}T} \left(\mathbb{E}_{p_{data}(\cdot)}[\|X_0\|^2] + o(e^{-\beta_{\min}T})\right) + \frac{\varepsilon^2}{2h} \cdot \frac{1}{2\min(\kappa,1) - 1 - 2h}.$$

Corollary 5.11 (Wasserstein bound for CVP)**.** *Assume that $p_{data}(\cdot)$ is κ-strongly log-concave and that $\mathbb{E}_{p_{data}(\cdot)}[\|X_0\|^2] < \infty$. Then we have*

$$W_2^2(p_{data}(\cdot), \mathcal{L}(Y_T)) \le e^{-2\left(\frac{\kappa}{1+\kappa} - \beta_{\max} hT + \mathcal{O}(e^{-\beta_{\min}T})\right)} \mathbb{E}_{p_{data}(\cdot)}[\|X_0\|^2] + \frac{\varepsilon^2}{2h(1-2h)}.$$

The detailed proofs of Corollaries 5.10 and 5.11, which follow from Theorem 5.7 by using specific expressions for $r_f(t)$ and $g(t)$, are given in Appendix B.4 and Appendix B.5, respectively.

5.3 Discretization and Predictor–Corrector Sampling

To simulate the generative process defined by the backward SDE (4), we need to use a numerical method. A simple and common approach is the *Euler–Maruyama scheme*, designed for SDEs of the form

$$dX_t = a(t, X_t)\, dt + b(t, X_t)\, dB_t,$$

It approximates the process using the discrete-time update

$$X_{k+1} = X_k + a(t_k, X_k)\, \Delta t + b(t_k, X_k)\, (B_{t_{k+1}} - B_{t_k}),$$

where $\Delta t = t_{k+1} - t_k$, and $B_{t_{k+1}} - B_{t_k} \sim \mathcal{N}(0, \Delta t\, I)$.

Application to the backward SDE. Applied to our setting, the Euler–Maruyama step becomes

$$\widehat{Y}_{k+1} = \widehat{Y}_k + \left(-f(T-t_k, \widehat{Y}_k) + a(T-t_k, \widehat{Y}_k)\, s_\theta(T-t_k, \widehat{Y}_k) + \nabla \cdot a(T-t_k, \widehat{Y}_k)\right) \Delta t + g(T-t_k)\, \Delta B_k, \tag{13}$$

with $\Delta B_k \sim \mathcal{N}(0, \Delta t\, I)$. This step is called the *predictor*, since it gives a first approximation of the new sample at each time step.

Corrector Step. To make the sample more accurate, we can apply a few steps of Langevin dynamics after the predictor. This second phase, called the *corrector*, improves the sample using the score estimate:

$$\widehat{Y}_k^{(\ell+1)} = \widehat{Y}_k^{(\ell)} + \epsilon_k\, s_\theta(T-t_k, \widehat{Y}_k^{(\ell)}) + \sqrt{2\epsilon_k}\, \xi_k^{(\ell)}, \quad \xi_k^{(\ell)} \sim \mathcal{N}(0, I), \tag{14}$$

starting from $\widehat{Y}_k^{(0)} = \widehat{Y}_k$, and setting $\widehat{Y}_k = \widehat{Y}_k^{(M)}$ after M iterations.

Motivation. This corrector step is based on the Langevin equation:

$$dX_t = \nabla \log p(T - t_k, X_t)\, dt + \sqrt{2}\, dB_t,$$

which naturally moves samples toward areas of higher density in the distribution $p(T - t_k, \cdot)$. By repeating this update several times, we can improve the match between our sample and the target distribution.

Since we do not know the exact gradient $\nabla \log p$, we use the approximation s_θ given by the score network. The predictor gives a first approximation, while the corrector locally refines it.

This combination is called the *Predictor–Corrector sampler*, and it is commonly used in practice to improve the quality of generated samples without adding too much computational cost.

6 Numerical Experiments

This section presents the experiments carried out during the project, following a chronological structure that reflects the challenges and learnings encountered along the way. Rather than detailing the implementation, we focus here on the main choices, observations, and lessons.

For readers interested in the exact implementation, all relevant code has been made available in documented Jupyter notebooks, which include comments and explanations in Markdown format.

6.1 Objectives and Methodology

The goal of these experiments was to implement the models studied in the theoretical sections and to understand their behavior in practice. Beyond validating the framework, the goal was to see how different design choices affect performance.

The experiments focused on several objectives:

- Understand the different steps involved in training and sampling.
- Compare the behavior of different SDEs (e.g., VP, COU).
- Evaluate how the architecture of the score network affects performance.
- Identify and analyze the main difficulties encountered during training.

The exploration began with a simple setup: two-dimensional Swiss Roll data and a basic score network. This initial configuration was chosen to validate the implementation with minimal complexity. However, this setup showed clear limitations in terms of performance.t To improve the results, deeper neural networks were introduced. In a second phase, a more complex and diverse dataset (MNIST) was used to test the model in more realistic conditions, beyond the synthetic Swiss Roll example.

This progressive approach allowed to gradually identify what works and what does not, and to adjust the experiments accordingly.

6.2 Initial Setup and First Trials

All experiments were run using PyTorch, which offered the flexibility needed to implement custom networks, training procedures, and sampling algorithms. As a first step, we created a two-dimensional Swiss Roll dataset, which is often used to test generative models because of its simple yet nontrivial structure.

We started by implementing two standards SDEs: the Ornstein–Uhlenbeck (OU) SDE (5) and the Variance Preserving (VP) SDE (8). These two formulations provided a concrete starting point to test the entire training and sampling pipeline in practice.

For the score network, we initially used a basic convolutional network with several stacked convolution layers. This choice was motivated by the nature of the data: the input is a two-dimensional image-like array, and the score function has the same dimensions as the input. Convolutional layers helped capture local patterns and produce coherent outputs.

The model was trained using denoising score matching (DSM), as described in Section 4.4 with optimization based on the Adam algorithm (see Appendix A.3). However, the first results were not convincing : both training and sampling was unstbale.

Because of these issues, the focus shifted to understanding where the instability was coming from. Several attempts were made using the VP and OU SDEs, but the results remained unstable. Even with different values for the hyperparameters and normalization methods, the generated samples were still noisy and inconsistent. One possible reason is that, although the forward process in these SDEs is contractive, the backward process is not. This means that small errors in the score function can grow during sampling, which makes generation unstable. Despite many trials, these models did not produce satisfying results. They might have worked better with the improvements tested later, but the rest of the work focused on models that gave more stable outcomes.

6.3 Improving Results with Better Models and Architectures

Therefore, to improve the sampling stability observed with VP and OU SDEs, I decided to test a different class of SDEs known as contractive SDEs (see Section 3.5). I implemented two of them: the Contractive Ornstein–Uhlenbeck (COU) SDE and the Contractive VP (CVP) SDE. These models are designed to be contractive in the backward direction, which helps make the generation more stable when the score function is imperfect.

However, this comes with a trade-off. Unlike VP or OU SDEs, the forward process of COU and CVP does not completely destroy the input. As a result, the final state retains some information about the original sample x_0. This means the model learns to reconstruct from a partially degraded input rather than to generate from pure noise. While this introduces a bias in theory, it is still possible in practice to sample from pure noise by applying the backward process from a fully random initial condition.

This trade-off is illustrated in Figure 1, where we compare the forward trajectories of a OU SDE and a contractive SDE (e.g., COU). In the OU case, the input is progressively destroyed until it reaches nearly pure noise at $t = T$. In contrast, the contractive SDE preserves more information, resulting in a final state that still depends on the original input.

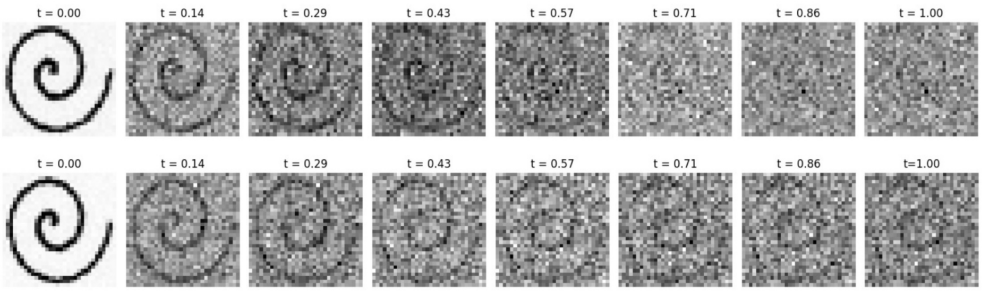

Figure 1: Comparison of forward processes for OU and COU SDEs with final time $T = 1$. The OU SDE fully destroys the structure of the input, while the COU SDE retains some information, resulting in a biased final state.

With contractive SDEs, notable improvements in sampling were observed. Although results remained imperfect, the generated samples began to show clearer patterns. Figure 2 shows an example using the COU SDE, where the general shape of the Swiss Roll starts to appear.

After reviewing recent diffusion models, it was noticed that the U-Net architecture, originally introduced for biomedical image segmentation, is frequently used in image generation tasks. A U-Net is a convolutional neural network with a symmetric structure, where the input is progressively downsampled and then upsampled, with skip connections linking corresponding layers. This design allows the model to capture both global and local features, making it particularly well-suited for score-based diffusion models.

Hence, the initial model was replaced with a U-Net. This upgrade led to more stable training and better visual quality, as shown in Figure 3, especially when working with more complex data.

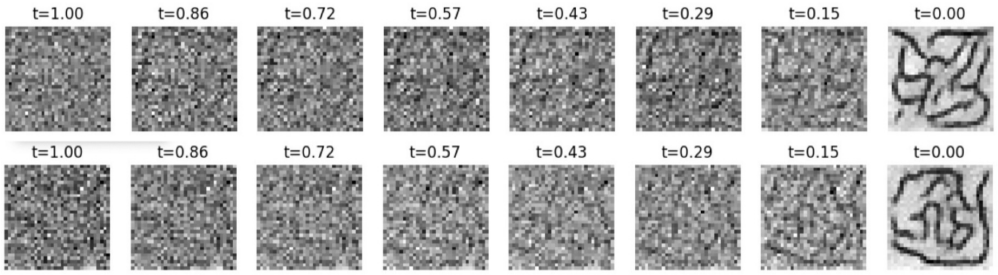

Figure 2: Early samples using the COU SDE. While still noisy, the overall shape of the Swiss Roll begins to emerge.

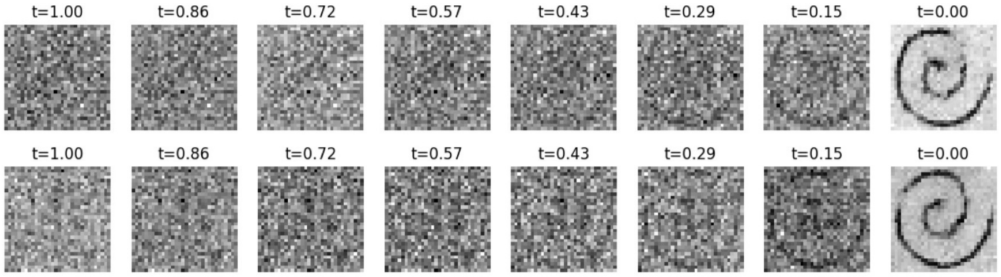

Figure 3: Improved samples after switching to a U-Net architecture. The generated shapes become sharper and better structured.

We also paid special attention to the choice of hyperparameters that control how the data is degraded during the forward process. For contractive SDEs, this includes the total time T, the shape of the noise schedule $\beta(t)$ (in particular $\beta_{\min}$ and $\beta_{\max}$ for the CVP SDE), and the values of θ and σ for the COU SDE.

These parameters need to be chosen with care. If the degradation is too weak, the model retains too much information from the original input, which introduces bias. On the other hand, if it is too strong or too abrupt, the process becomes unstable or hard to reverse. The goal was to find a good balance: a slow and smooth degradation that removes enough information while preserving the underlying structure. This helped reduce the bias and improved the stability of the generation.

Once the results on Swiss Roll became more convincing, we applied the same setup to the MNIST dataset. MNIST is a more complex and diverse dataset, with a variety of digit shapes. The transition to MNIST went relatively well, and the model produced coherent digit samples in many cases.

Figure 4 shows sampling trajectories with the COU SDE and U-Net. We can see that the model progressively refines its output, starting from a noisy initialization and converging toward structured digits. This confirmed that the training pipeline was functioning correctly in a more realistic setting.

However, some limitations remained. In a few cases, the model got stuck in poor local minima, producing blurry or ambiguous shapes that did not resemble any clear digit. In some cases, the model produced blurry or ambiguous digits, as if it were mixing several digit classes together. This indicates that the score function had difficulty clearly separating the different digit types. While such failures were not frequent, they highlight how challenging it remains to generate precise and well-defined samples, even with the improvement made in the architecture.

6.4 Final Improvements and Last Tests

Predictor–Corrector Sampling Until now, the sampling process used a basic Euler–Maruyama discretization of the reverse SDE. While simple to implement, this method sometimes produced noisy or unstable results, especially at the end of the trajectory.

These issues often come from inaccuracies in the learned score function. When the score is not precise, the sampling can drift and produce irregular or messy output.

To address this, a Predictor–Corrector (PC) sampler was implemented, as described in Section 5.3. It combines a predictor step (Euler–Maruyama) with a corrector step (Langevin dynamics). The corrector helps reduce noise in the score estimates and leads to more stable results.

The results are shown in Figures 5 and 6. Each figure compares two trajectories sampled independently: the top row uses standard Euler sampling, and the bottom row uses the PC sampler.

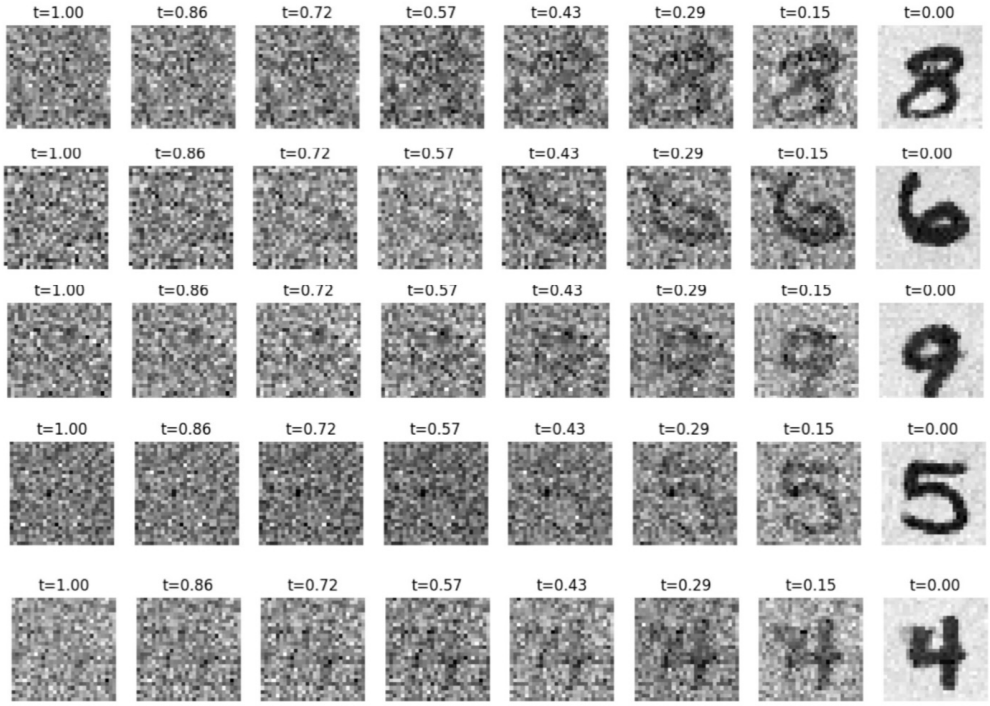

Figure 4: Sampling process on MNIST using contractive SDEs. From left to right: the model gradually transforms pure noise into a structured digit.

The difference is most visible at the end of the sampling process ($t = 0$), where the PC version often produces cleaner shapes with more stable contours.

Attempts with More Complex Images We also tried to apply the method to a more complex dataset: CIFAR-10, which contains small RGB images. These tests are still ongoing, and the current results are not convincing.

Several issues emerged. Training took significantly more time, which slowed down experimentation. Also, the generated images often collapsed into a single color or lacked clear structure.

These problems might come from multiple factors: a model too small for the task, poor hyperparameter choices, or the method needing further adaptation. Solving them would likely require a larger network, better tuning, and more computing power.

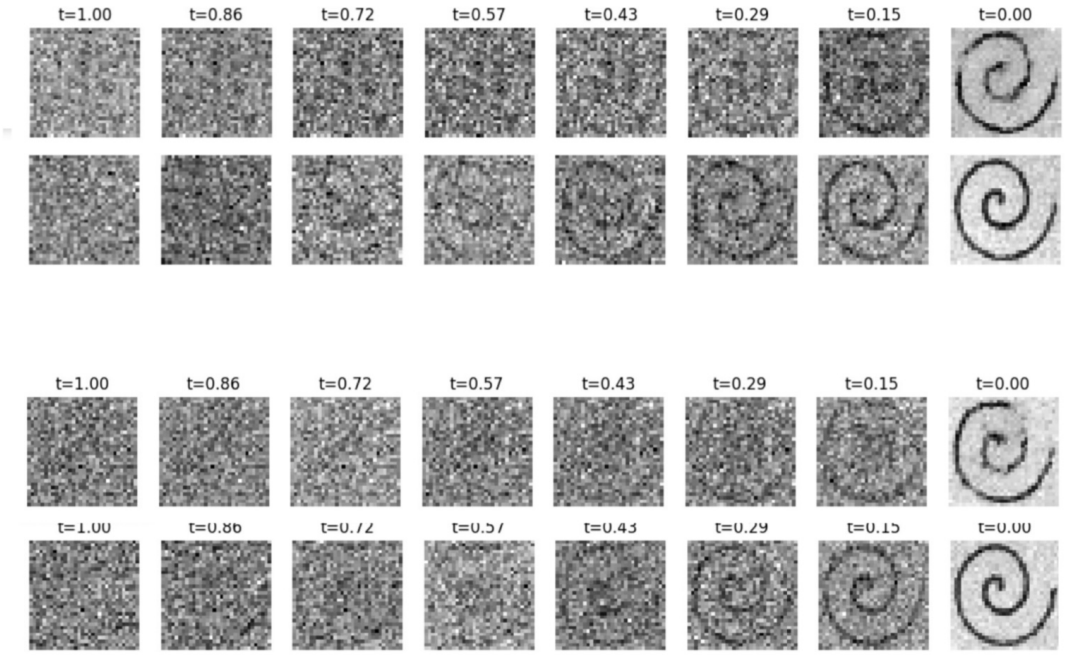

Figure 5: Comparison of samples generated with the basic Euler-Maruyama sampler (top row) and the Predictor–Corrector sampler (bottom row) on the Swiss Roll dataset. The PC version produces more structured and less noisy results.

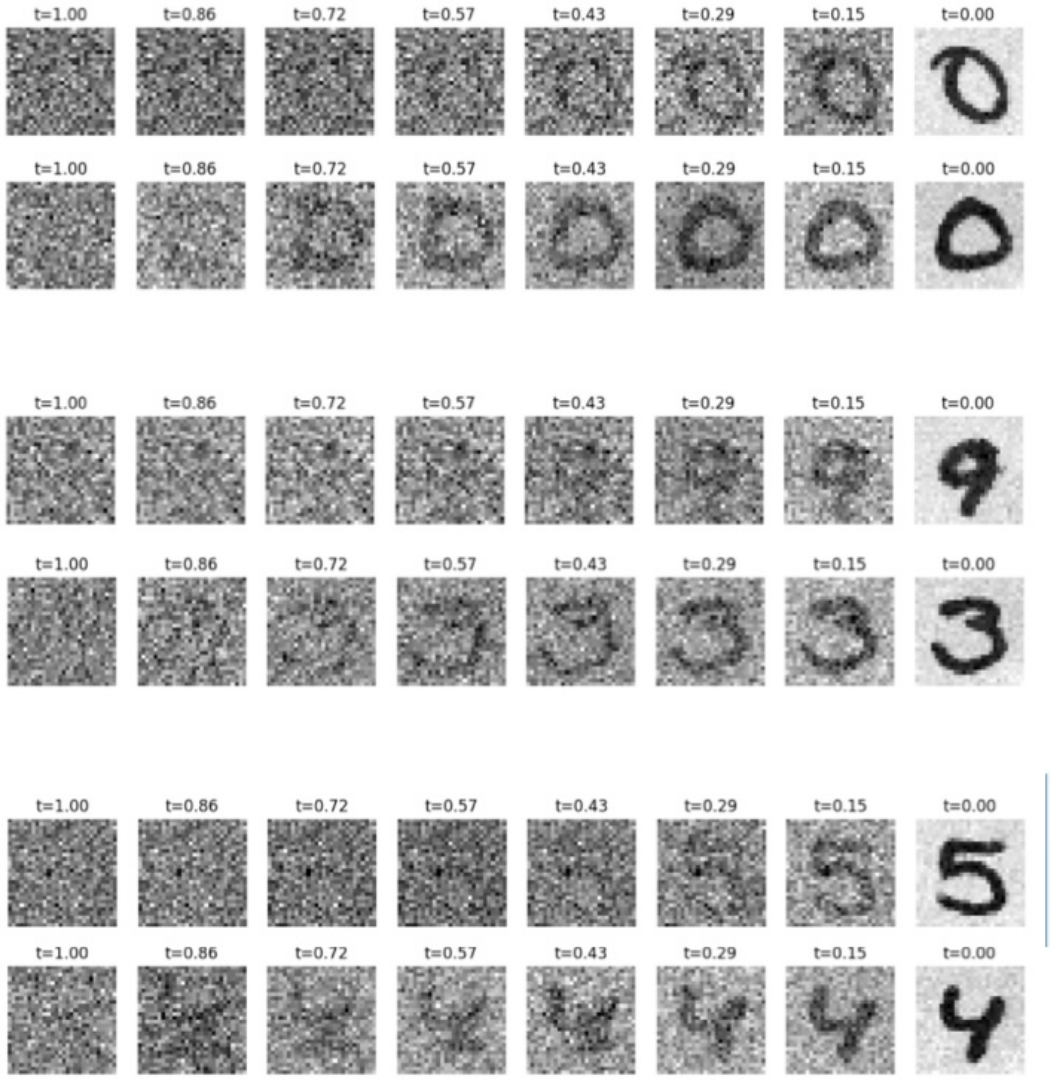

Figure 6: Comparison of MNIST digits generated with the Euler-Maruyama sampler (top row) and the Predictor–Corrector sampler (bottom row). The PC version improves shape regularity and reduces blur.

7 Conclusion

This report has examined both the theoretical and practical aspects of score-based generative modeling using stochastic differential equations (SDEs). Starting from the formulation of forward and backward processes and the time-reversal formula, we reviewed various score matching techniques and analyzed convergence guarantees in total variation and Wasserstein distances. In several cases, these results were clarified or refined with more detailed derivations and adjustments for increased precision.

On the experimental side, we implemented and evaluated different SDEs and neural network architectures to assess their impact on sample quality. On simple datasets like Swiss Roll and MNIST, our experiments showed that contractive SDEs, such as the Conservative Ornstein–Uhlenbeck (COU) process combined with U-Net architectures, often produced clearer and more stable outputs. Although specific to our setup, this observation highlights the influence of SDE design on empirical performance.

While the results on simple datasets were encouraging, several limitations emerged. On complex datasets like CIFAR-10, the model struggled to generate convincing samples. Training was slower and less stable, and the resulting images often lacked semantic structure. These issues likely stem from limited model capacity, suboptimal hyperparameters, and insufficient computational resources.

Future work could explore several directions. A first direction involves deterministic samplers using ODE solvers, such as the probability flow formulation, which allow generating samples in a single forward pass with significantly improved efficiency [21, 22]. Another active research area is the integration of reinforcement learning into diffusion models, with several recent works exploring reward-based training and alignment with user preferences [6, 1, 18]. These approaches enable models to better align with task-specific objectives or human feedback, making them more adaptable and useful in real-world applications.

References

[1] Kevin Black et al. *Training Diffusion Models with Reinforcement Learning*. 2023. arXiv: 2305.13301 [math.]. URL: https://arxiv.org/abs/2305.13301.

[2] Minshuo Chen et al. *An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and Optimization*. 2024. arXiv: 2403.17181 [cs.LG]. URL: https://arxiv.org/abs/2403.17181.

[3] Sitan Chen et al. *Sampling is as Easy as Learning the Score: Theory for Diffusion Models with Minimal Data Assumptions*. 2023. arXiv: 2209.11215 [cs.LG]. URL: https://arxiv.org/abs/2209.11215.

[4] Prafulla Dhariwal and Alex Nichol. "Diffusion models beat GANs on image synthesis". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[5] Xuefeng Gao, Hoang M. Nguyen, and Lingjiong Zhu. *Wasserstein Convergence Guarantees for a General Class of Score-Based Generative Models*. 2023. arXiv: 2311.11003 [cs.LG]. URL: https://arxiv.org/abs/2311.11003.

[6] Xuefeng Gao, Jiale Zha, and Xun Yu Zhou. *Reward-Directed Score-Based Diffusion Models via q-Learning*. 2024. arXiv: 2409.04832 [cs.LG]. URL: https://arxiv.org/abs/2409.04832.

[7] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[8] U. G. Haussmann and E. Pardoux. "Time reversal of diffusions". In: *The Annals of Probability* 14.4 (1986), pp. 1188–1205.

[9] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017, pp. 6626–6637. arXiv: 1706.08500 [cs.LG]. URL: https://arxiv.org/abs/1706.08500.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[11] Aapo Hyvärinen. "Estimation of non-normalized statistical models by score matching". In: *Journal of Machine Learning Research* 6.4 (2005), pp. 695–709.

[12] Ioannis Karatzas and Steven E Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 1998.

[13] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)* (2015). arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[15] Zhifeng Kong et al. *DiffWave: A Versatile Diffusion Model for Audio Synthesis*. 2021. arXiv: 2009.09761 [cs.SD]. URL: https://arxiv.org/abs/2009.09761.

[16] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *International Conference on Machine Learning (ICML)*. 2016, pp. 1558–1566. URL: https://arxiv.org/abs/1512.09300.

[17] Yann LeCun et al. "A tutorial on energy-based learning". In: *Predicting structured data* 1.0 (2006), pp. 1–59.

[18] Kimin Lee et al. *Aligning Text-to-Image Models using Human Feedback*. 2023. arXiv: 2302.12192 [cs.LG]. URL: https://arxiv.org/abs/2302.12192.

[19] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV]. URL: https://arxiv.org/abs/2204.06125.

[20] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning (ICML)* (2015).

[21] Yang Song et al. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations (ICLR)* (2021). arXiv: 2011.13456 [stat.ML]. URL: https://arxiv.org/abs/2011.13456.

[22] Wenpin Tang and Hanyang Zhao. *Score-based Diffusion Models via Stochastic Differential Equations – a Technical Tutorial*. 2024. arXiv: 2402.07487 [cs.LG]. URL: https://arxiv.org/abs/2402.07487.

[23] Pascal Vincent. "A connection between score matching and denoising autoencoders". In: *Neural Computation* 23.7 (2011), pp. 1661–1674.

A Prerequisites and Mathematical Background

A.1 Stochastic Differential Equations and Itô Calculus

Throughout this section, we work with a fixed probabilistic framework. All random variables and stochastic processes are defined on a filtered probability space

$$(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\in[0,T]}, \mathbb{P}),$$

which satisfies the usual conditions (completeness and right-continuity).

Unless otherwise specified, we will use this setup for all definitions and results involving stochastic differential equations and stochastic integrals.

A.1.1 Stochastic Differential Equations

We begin by introducing the general setting for stochastic differential equations (SDEs), which model the evolution of systems subject to both deterministic dynamics and stochastic perturbations.

Let $\{B_t\}_{t\in[0,T]}$ be a standard n-dimensional Brownian motion adapted to the filtration $\{\mathcal{F}_t\}_{t\in[0,T]}$.

A $\mathbb{R}^d$-valued stochastic process $X_{tt\in[0,T]}$ is said to solve a stochastic differential equation if it satisfies, almost surely, the integral equation:

$$X_t = Z + \int_0^t f(s, X_s)\, ds + \int_0^t g(s, X_s)\, dB_s, \quad t \in [0, T], \tag{15}$$

where:

- Z is the initial condition. It is a $\mathbb{R}^d$-valued random variable that is measurable with respect to $\mathcal{F}_0$;
- $f : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is a measurable function called the **drift coefficient**, representing the deterministic part of the dynamics;
- $g : [0, T] \times \mathbb{R}^d \to \mathbb{R}^{d\times n}$ is a measurable function called the **diffusion coefficient**. It controls how the noise from the Brownian motion influences the system.

The stochastic process $\{X_t\}$ thus evolves according to the differential notation

$$dX_t = f(t, X_t)\, dt + g(t, X_t)\, dB_t, \quad X_0 = Z, \tag{16}$$

where the term dB_t denotes the infinitesimal increment of Brownian motion, interpreted in the Itô sense.

The term $g(t, X_t)\, dB_t$ represents a random perturbation in $\mathbb{R}^d$. It comes from the multiplication of the diffusion matrix $g(t, X_t)$ with the Brownian increment dB_t. This part of the equation describes how the noise affects the system at each time, depending on its current state.

This formulation serves as the foundational model for the stochastic dynamics studied in this work. In the next section, we define the Itô integral, which gives rigorous meaning to the stochastic term.

Remark A.1. *In this work, we restrict ourselves to the classical framework where the noise is driven by a standard Brownian motion, and all integrals are interpreted in the Itô sense. We assume that the integrands are adapted and square-integrable processes.*

Although more general constructions exist (e.g., integration with respect to local martingales or semimartingales), they are beyond the scope of this study and will not be considered.

A.1.2 Itô Integral

The Itô integral is the fundamental tool for defining integration with respect to Brownian motion. Unlike Riemann or Lebesgue integrals, it is specifically designed for integrating against processes that have continuous but nowhere differentiable paths, such as Brownian motion.

Let $\{X_t\}_{t\in[0,T]}$ be a stochastic process. The Itô integral of X with respect to Brownian motion B over $[0,t]$ is denoted by:

$$\int_0^t X_s \, dB_s. \tag{17}$$

In this work, we consider the classical framework in which the Itô integral is defined whenever the process $\{X_t\}_{t\in[0,T]}$ satisfies the following assumptions:

- It is **adapted** to the filtration generated by the Brownian motion, meaning that X_t only depends on information available up to time t;
- It is **square-integrable**, that is, $\mathbb{E}\left[\int_0^t X_s^2 \, ds\right] < \infty$.

Under these conditions, the Itô integral satisfies the following properties:

- **Isometry:** $\mathbb{E}\left[\left(\int_0^t X_s \, dB_s\right)^2\right] = \mathbb{E}\left[\int_0^t X_s^2 \, ds\right]$;
- **Martingale property:** The process $\left\{\int_0^t X_s \, dB_s\right\}_{t\in[0,T]}$ is a martingale.

These properties make the Itô integral central to stochastic calculus and to the analysis of stochastic differential equations.

A.1.3 Itô's Formula

Itô's formula is an important result in stochastic calculus. It extends the chain rule from classical calculus to stochastic processes. Although we have not formally defined stochastic integration with respect to general semimartingales, we present here the classical version of Itô's formula.

Theorem A.2 (Itô's formula)**.** *Let $\{X_t\}_{t\in[0,T]}$ be a continuous semimartingale in $\mathbb{R}^d$, and let $\phi : [0,T]\times\mathbb{R}^d \to \mathbb{R}$ be a function of class $\mathcal{C}^{1,2}$. Then, for all $t \in [0,T]$,*

$$\begin{aligned}\phi(t, X_t) = \phi(0, X_0) &+ \int_0^t \frac{\partial \phi}{\partial s}(s, X_s)\, ds + \sum_{i=1}^d \int_0^t \frac{\partial \phi}{\partial x_i}(s, X_s)\, dX_s^i \\ &+ \frac{1}{2}\sum_{i,j=1}^d \int_0^t \frac{\partial^2 \phi}{\partial x_i \partial x_j}(s, X_s)\, d\langle X^i, X^j\rangle_s.\end{aligned}$$

Remark A.3. *The process $\langle X^i, X^j\rangle_t$ is called the quadratic covariation between the components X^i and X^j.*

When X_t solves a SDE with diffusion matrix $g(t, X_t) \in \mathbb{R}^{d\times n}$, this quantity is given by

$$d\langle X^i, X^j\rangle_t = \left(g(t, X_t) g(t, X_t)^\top\right)_{i,j} \, dt.$$

For a general definition and more properties of quadratic variation, see for example [12]

A.1.4 Solutions of a Stochastic Differential Equation

We consider stochastic differential equations of the form:

$$dX_t = f(t, X_t)\, dt + g(t, X_t)\, dB_t, \quad X_0 = Z, \tag{18}$$

where all terms are defined as previously.

Definition A.4. *A **strong solution** to this SDE is a stochastic process $\{X_t\}_{t\in[0,T]}$ that satisfies the following conditions:*

- *The process $\{X_t\}$ is adapted to the filtration $\{\mathcal{F}_t\}$;*
- *The filtration contains all the information from the initial condition Z and the Brownian motion $\{B_t\}$, that is,*
$$\mathcal{F}_t \supseteq \sigma\left(Z, B_s : 0 \le s \le t\right);$$
- *The process satisfies, almost surely, the integral form of the SDE:*
$$X_t = Z + \int_0^t f(s, X_s)\, ds + \int_0^t g(s, X_s)\, dB_s.$$

We now ask under which conditions such a solution exists and is unique.

If the coefficients f and g satisfy some regularity conditions, we can guarantee that the SDE has a unique strong solution. This is stated in the following theorem.

Theorem A.5 (Existence and Uniqueness)**.** *Let $f : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $g : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times n}$ be two measurable functions. Suppose that they satisfy:*

- ***Lipschitz condition:** There exists a constant $L > 0$ such that for all $t \in [0,T]$ and all $x, y \in \mathbb{R}^d$,*
$$|f(t,x) - f(t,y)| + |g(t,x) - g(t,y)| \le L|x - y|;$$
- ***Linear growth condition:** There exists a constant $K > 0$ such that for all $t \in [0,T]$ and all $x \in \mathbb{R}^d$,*
$$|f(t,x)|^2 + |g(t,x)|^2 \le K(1 + |x|^2).$$

Then there exists a unique strong solution $\{X_t\}_{t\in[0,T]}$ to the SDE.

Remark A.6. *In this work, we only consider strong solutions, as they are sufficient for the models studied here.*

A.2 Neural Networks and Function Approximation

Disclaimer: This section is largely inspired by the lecture notes of Yating Liu on Statistical Learning. The presentation of neural networks and the associated notations closely follow the structure and explanations introduced in her course materials, with minor adaptations for the context of this work.

In the following, we adopt the notation where $z^{(l)}$ refers to the pre-activation values at layer l, and $a^{(l)}$ denotes the corresponding post-activation values, i.e., the outputs of the neurons after applying the activation function. This convention aligns with the visual representation introduced in the next figure.

Definition A.7 (Neuron)**.** *A **neuron** is a function that maps an input vector $x \in \mathbb{R}^d$ to a real output using an affine transformation followed by a non-linear activation:*

$$a = \phi(w^\top x + b),$$

where $w \in \mathbb{R}^d$ is the weight vector, $b \in \mathbb{R}$ is the bias, and $\phi : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function.

Definition A.8 (Layer of Neurons)**.** *A **layer** with h neurons applies this transformation in parallel:*

$$z = Wx + b, \quad a = \phi(z),$$

where $W \in \mathbb{R}^{h \times d}$ is the weight matrix, $b \in \mathbb{R}^h$ is the bias vector, and the activation function ϕ is applied component-wise.

Definition A.9 (Feedforward Neural Network (MLP))**.** *A **feedforward neural network** (or multilayer perceptron) with L hidden layers computes a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^m$ via a sequence of transformations:*

$$\begin{aligned} a^{(0)} &= x, \\ z^{(l)} &= W^{(l)} a^{(l-1)} + b^{(l)}, \\ a^{(l)} &= \phi(z^{(l)}), \quad \text{for } l = 1, \dots, L, \end{aligned}$$

where each $W^{(l)} \in \mathbb{R}^{h_l \times h_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{h_l}$ are trainable parameters, and ϕ is applied component-wise.

The final output is usually computed as

$$f_\theta(x) = \psi(W^{(L+1)} z^{(L)} + b^{(L+1)}),$$

where ψ is the final activation and is chosen depending on the task: for example, the identity function for regression, or a softmax for classification. The full set of parameters is $\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^{L+1}$.

Figure 7 gives a simple visual overview of the network structure introduced above. This kind of architecture explains why neural networks are so powerful at approximating functions, as we will now see.

Theorem A.10 (Universal Approximation Theorem)**.** *Let $\phi : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, and continuous activation function. Then, for any continuous function $f : [0,1]^d \to \mathbb{R}$ and any $\varepsilon > 0$, there exists a neural network f_θ with a single hidden layer such that:*

$$\sup_{x \in [0,1]^d} |f(x) - f_\theta(x)| < \varepsilon.$$

This result tells us that neural networks can get as close as we want to any continuous function on a compact domain, as long as the network is wide enough and well chosen.

This means that neural networks can approximate a wide range of functions, which is why they are used in many applications.

A.3 Stochastic Optimization Methods

In machine learning, and especially deep learning, we want to adjust the model parameters $\theta \in \Theta \subseteq \mathbb{R}^p$ to minimize a **loss function** $\mathcal{L}(\theta)$. This function measures how far the model's predictions are from the true outputs, and plays a central role during training.

The parameters θ usually include all the weights and biases of the neural network. Since we do not know the full data distribution, we cannot compute $\mathcal{L}(\theta)$ exactly. Instead, we use a finite dataset and minimize an approximation of this loss using gradient-based methods.

This section explains how the loss is defined from data, and how we minimize it using stochastic optimization techniques like stochastic gradient descent (SGD).

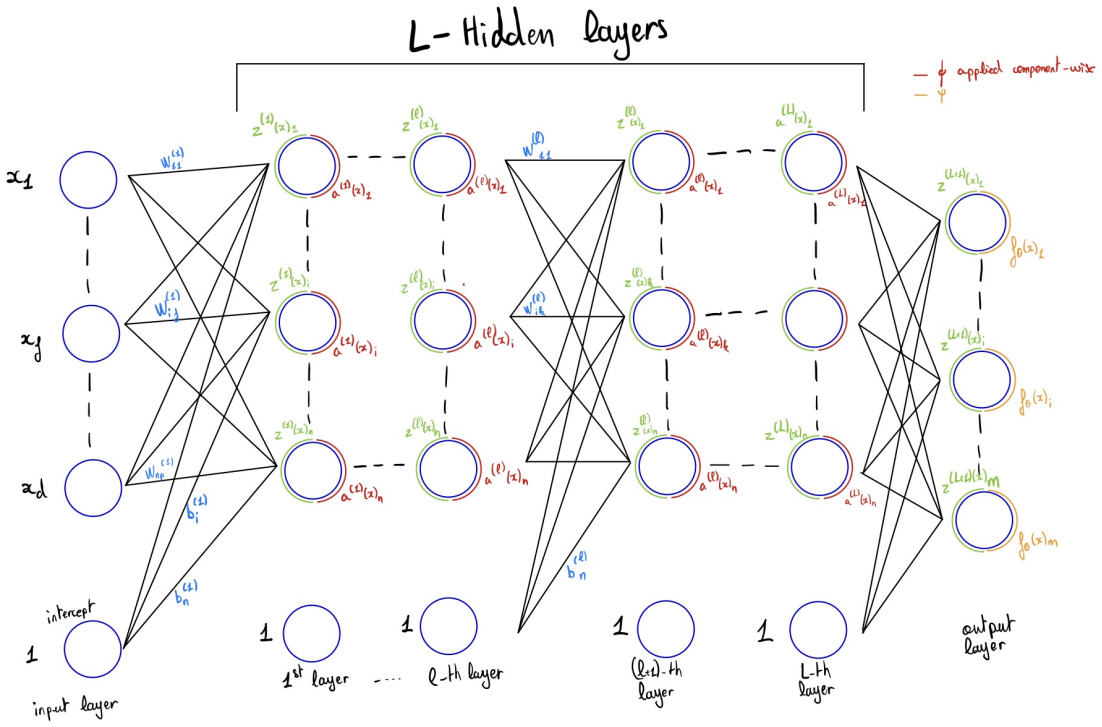

Figure 7: Architecture of a feedforward neural network with L hidden layers, each containing n neurons. Each neuron first computes a linear combination of its inputs (shown in green), then applies a non-linear activation function ϕ (in red). The output layer applies a final activation ψ, adapted to the specific task (in orange).

A.3.1 Training Objective from Data.

Let $(\mathcal{X}, \mathcal{Y}, \rho)$ be a probability space representing the data distribution, where ρ is a probability measure over input-output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. For a model $f_\theta : \mathcal{X} \to \mathcal{Y}$, the expected loss is defined as:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)\sim\rho}\left[\ell(f_\theta(x), y)\right],$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ measures the discrepancy between the prediction and the true output.

A common example is the **mean squared error** (MSE):

$$\ell(f_\theta(x), y) = \|f_\theta(x) - y\|^2,$$

which gives the expected loss :

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)\sim\rho}\left[\|f_\theta(x) - y\|^2\right].$$

Since the distribution ρ is unknown, we approximate the expected loss using a dataset $\{(x_i, y_i)\}_{i=1}^N$. This gives the **empirical loss**:

$$\widehat{\mathcal{L}}_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} \|f_\theta(x_i) - y_i\|^2.$$

A.3.2 Gradient-based optimization.

To minimize the empirical loss $\widehat{\mathcal{L}}_N(\theta)$, we use iterative methods based on gradients. The simplest approach is **gradient descent**, which updates the parameters at each step according to:

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta \widehat{\mathcal{L}}_N(\theta_k),$$

where $\eta > 0$ is called the *learning rate.*This quantity controls how large each update step is. In this update, we need to compute the full gradient of the empirical loss:

$$\nabla_\theta \widehat{\mathcal{L}}_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \ell(f_\theta(x_i), y_i).$$

If the data points (x_i, y_i) are drawn independently from the same distribution, this quantity gives an unbiased estimate of the true gradient $\nabla_\theta \mathcal{L}(\theta)$. However, computing this full sum can be very expensive when the dataset is large.

A.3.3 Stochastic Gradient Descent (SGD).

To reduce the cost of computing the full gradient, **stochastic gradient descent** (SGD) uses a random estimate based on a small part of the dataset. At each step, we randomly select a subset $\mathcal{B}_k \subset \{1, \ldots, N\}$ of size $m \ll N$ (called a mini-batch), and update the parameters as follows:

$$\theta_{k+1} = \theta_k - \eta \cdot \frac{1}{m} \sum_{i \in \mathcal{B}_k} \nabla_\theta \ell(f_\theta(x_i), y_i).$$

This produces an unbiased but noisy estimate of the gradient. Interestingly, this randomness can be beneficial: it may help escape sharp local minima and instead reach flatter regions, which often lead to better generalization.

Remark A.11. *Stochastic methods like SGD are widely used for training neural networks. They are simple, efficient on large datasets, and work well in practice under mild assumptions.*

A.3.4 Adaptive Methods: Adam

Stochastic gradient descent (SGD) is simple and effective, but it can be sensitive to the choice of learning rate. It also uses the same step size for all parameters, which may lead to slow progress in some directions.

To improve this, the **Adam** algorithm (short for *adaptive moment estimation*) adjusts the learning rate for each parameter by using information from past gradients. At each step t, Adam keeps track of two quantities:

- A moving average of past gradients (called the *momentum* term):

  $$m_t = \gamma_1 m_{t-1} + (1 - \gamma_1) g_t,$$

  This term averages the past gradients, which helps the algorithm move more steadily in useful directions and reduces sudden changes during training.

- A moving average of the squares of past gradients:

  $$r_t = \gamma_2 r_{t-1} + (1 - \gamma_2) g_t^2,$$

  This second term measures how big the gradients are on average. It helps the algorithm take smaller steps when the gradients change a lot, and bigger steps when they are more stable.

where g_t is the gradient at step t, and typical values are $\gamma_1 = 0.9$, $\gamma_2 = 0.999$.

At the beginning of training, these moving averages are close to zero. To correct this bias, Adam uses the following formulas:

$$\hat{m}_t = \frac{m_t}{1-\gamma_1^t}, \quad \hat{r}_t = \frac{r_t}{1-\gamma_2^t}.$$

The parameters are updated as:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{r}_t} + \varepsilon},$$

where η is the learning rate, and ε is a small constant (e.g. 10^{-8}) to avoid division by zero.

Remark A.12. *Adam is widely used because it combines the benefits of momentum (via m_t) and adaptive step sizes (via r_t). It generally leads to faster convergence and works well with little tuning. In our case, we use Adam to train the neural networks involved in approximating unknown functions.*

B Proofs of convergence results

B.1 Proof of Theorem 5.3

To prove Theorem 5.3, we will use three classical tools: the data processing inequality, Pinsker's inequality, and a consequence of Girsanov's theorem. We recall them here for completeness.

Lemma B.1 (Data processing inequality)**.** *Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability measures on a space Ω, and let $X : \Omega \to \mathcal{X}$ be a measurable function. Then*

$$\mathrm{KL}(\mathbb{P}_X \| \mathbb{Q}_X) \leq \mathrm{KL}(\mathbb{P} \| \mathbb{Q}),$$

where $\mathbb{P}_X$ and $\mathbb{Q}_X$ are the pushforward measures of $\mathbb{P}$ and $\mathbb{Q}$ by X.

Lemma B.2 (Pinsker's inequality)**.** *Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distributions on the same measurable space. Then*

$$\mathrm{d}_{TV}(\mathbb{P}, \mathbb{Q}) \leq \sqrt{\frac{1}{2} \mathrm{KL}(\mathbb{P} \| \mathbb{Q})}.$$

Lemma B.3 (Consequence of Girsanov's theorem)**.** *Let $B = (B_t)_{t\in[0,T]}$ be a Brownian motion under a probability measure $\mathbb{Q}$, and let $\mathcal{L}_t := \int_0^t b_s \, \mathrm{d}B_s$, where $(b_t)_{t\in[0,T]}$ is an adapted process.*

Assume that

$$\mathbb{E}_{\mathbb{Q}} \left[\exp\left(\frac{1}{2} \int_0^T \|b_t\|^2 \, \mathrm{d}t \right) \right] < \infty.$$

Then the process

$$\mathcal{E}(\mathcal{L})_t := \exp\left(\int_0^t b_s \cdot \mathrm{d}B_s - \frac{1}{2} \int_0^t \|b_s\|^2 \, \mathrm{d}s \right)$$

is a martingale, and defines a new probability measure $\mathbb{P}$ on $(\Omega, \mathcal{F}_T)$ by $\mathrm{d}\mathbb{P} = \mathcal{E}(\mathcal{L})_T \, \mathrm{d}\mathbb{Q}$.

Under $\mathbb{P}$, the process

$$W_t := B_t - \int_0^t b_s \, \mathrm{d}s$$

is a Brownian motion.

We now prove Theorem 5.3 using these results. We work on the path space $\mathcal{C}([0,T], \mathbb{R}^d)$ and define the following probability measures :

- Q_T: law of the process $(Y_t)_{t\in[0,T]}$ associated with

$$\mathrm{d}Y_t = -(f(T-t,Y_t) + g^2(T-t)s_\theta(T-t,Y_t))\,\mathrm{d}t + g(T-t)\,\mathrm{d}B_t, \quad Y_0 \sim p_{\text{noise}}(\cdot),$$

- $Q_T^{'}$: same dynamics as Q_T, but with initial distribution $Y_0 \sim p(T,\cdot)$,
- $Q_T^{''}$: same initial distribution as $Q_T^{'}$, but using the true score $\nabla \log p$.

We decompose the total variation distance as :

$$\begin{aligned}\mathrm{d_{TV}}\left(\mathcal{L}(Y_T), p_{\text{data}}(\cdot)\right) &\le \mathrm{d_{TV}}(Q_T, Q_T^{'}) + \mathrm{d_{TV}}(Q_T^{'}, Q_T^{''}) \\ &\le \mathrm{d_{TV}}(p(T,\cdot), p_{\text{noise}}(\cdot)) + \mathrm{d_{TV}}(Q_T^{'}, Q_T^{''}). \end{aligned} \tag{19}$$

using the triangle inequality and the data processing inequality.

We now try to bound the term $\mathrm{KL}(Q_T^{''}\|Q_T^{'})$ to control the total variation distance.

We apply Lemma B.3 with

$$\mathbb{Q} = Q_T^{''}, \quad b_t := g(T-t)\left(s_\theta(T-t,Y_t) - \nabla \log p(T-t,Y_t)\right).$$

Then under $\mathbb{P} := \mathcal{E}(\mathcal{L})_T Q_T^{''}$, there exists a Brownian motion $\tilde{B}$ such that

$$\mathrm{d}B_t = \mathrm{d}\tilde{B}_t + g(T-t)\left(s_\theta(T-t,Y_t) - \nabla \log p(T-t,Y_t)\right)\mathrm{d}t.$$

Under $Q_T^{''}$, we have almost surely:

$$\mathrm{d}Y_t = -(f(T-t,Y_t) + g^2(T-t)\nabla \log p(T-t,Y_t))\,\mathrm{d}t + g(T-t)\,\mathrm{d}B_t.$$

This equation remains valid $\mathbb{P}$-a.s. since $\mathbb{P} \ll Q_T^{''}$, even if B is not a Brownian motion under $\mathbb{P}$.

Plugging the previous expression of $\mathrm{d}B_t$ into this equation, we obtain:

$$\begin{aligned}\mathrm{d}Y_t &= -(f(T-t,Y_t) + g^2(T-t)\nabla \log p(T-t,Y_t))\,\mathrm{d}t \\ &\quad + g(T-t)\left[\mathrm{d}\tilde{B}_t + g(T-t)\left(s_\theta(T-t,Y_t) - \nabla \log p(T-t,Y_t)\right)\mathrm{d}t\right] \\ &= -(f(T-t,Y_t) + g^2(T-t)s_\theta(T-t,Y_t))\,\mathrm{d}t + g(T-t)\,\mathrm{d}\tilde{B}_t.\end{aligned}$$

Therefore, under $\mathbb{P}$, Y has the same dynamics as under $Q_T^{'}$, so $\mathbb{P} = Q_T^{'}$.

Now we compute the KL divergence:

$$\begin{aligned}\mathrm{KL}(Q_T^{''}\|Q_T^{'}) &= \mathbb{E}_{Q_T^{''}}\left[\log\left(\mathcal{E}(\mathcal{L})_T^{-1}\right)\right] \\ &= \mathbb{E}_{Q_T^{''}}\left[\int_0^T \|b_t\|^2\,\mathrm{d}t - \int_0^T b_t \cdot \mathrm{d}B_t\right] \\ &= \mathbb{E}_{Q_T^{''}}\left[\int_0^T \|b_t\|^2\,\mathrm{d}t\right] \\ &= \mathbb{E}_{Q_T^{''}}\left[\int_0^T g(T-t)^2\left\|s_\theta(T-t,Y_t) - \nabla \log p(T-t,Y_t)\right\|^2\,\mathrm{d}t\right] \\ &\le \|g\|_\infty^2 \cdot \mathbb{E}_{Q_T^{''}}\left[\int_0^T \left\|s_\theta(T-t,Y_t) - \nabla \log p(T-t,Y_t)\right\|^2\,\mathrm{d}t\right] \\ &\le \|g\|_\infty^2 \cdot T \cdot \varepsilon^2.\end{aligned}$$

Applying Pinsker's inequality:

$$\mathrm{d_{TV}}(Q_T^{'}, Q_T^{''}) \le \sqrt{\frac{1}{2}\mathrm{KL}(Q_T^{''}\|Q_T^{'})} \le \varepsilon\|g\|_\infty\sqrt{\frac{T}{2}}. \tag{20}$$

Combining inequalities (19) and (20) gives the final result. □

B.2 Proof of Corollary 5.5

We first recall the formula for the KL divergence between two multivariate Gaussians:

Lemma B.4 (KL divergence between Gaussians)**.** *Let $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ be two Gaussian distributions in $\mathbb{R}^d$, with Σ_1, Σ_2 symmetric positive definite. Then,*

$$\mathrm{KL}\left(\mathcal{N}(\mu_1, \Sigma_1) \| \mathcal{N}(\mu_2, \Sigma_2)\right) = \frac{1}{2}\left(\mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^\top \Sigma_2^{-1}(\mu_2 - \mu_1) - d + \log \frac{\det \Sigma_2}{\det \Sigma_1}\right).$$

We now prove Corollary 5.5.

From equation (7), the law of X_T given $X_0 = x$ is

$$X_t \mid X_0 = x \sim \mathcal{N}\left(e^{-\frac{1}{2}\int_0^t \beta(s)\,\mathrm{d}s}\, x, \left(1 - e^{-\int_0^t \beta(s)\,\mathrm{d}s}\right) I\right).$$

We have the identity :

$$p(T, \cdot) = \mathbb{E}_{X_0 \sim p_{\mathrm{data}}(\cdot)}[p(T, \cdot \mid X_0)],$$

Using the convexity of the KL divergence and Jensen's inequality:

$$\begin{aligned}
\mathrm{KL}(p(T, \cdot) \| p_{\mathrm{noise}}(\cdot)) &= \mathrm{KL}\left(\mathbb{E}_{p_{\mathrm{data}}(\cdot)}[p(T, \cdot \mid X_0)] \middle\| \mathcal{N}(0, I)\right) \\
&\leq \mathbb{E}_{p_{\mathrm{data}}(\cdot)}\left[\mathrm{KL}\left(p(T, \cdot \mid X_0) \| \mathcal{N}(0, I)\right)\right].
\end{aligned}$$

Each conditional law is Gaussian with mean $\mu = e^{-\frac{1}{2}\int_0^T \beta(s)\,\mathrm{d}s} X_0$ and covariance $\Sigma = (1 - e^{-\int_0^T \beta(s)\,\mathrm{d}s})I$. Applying Lemma B.4, we get:

$$\begin{aligned}
\mathrm{KL}\left(p(T, \cdot \mid X_0) \| \mathcal{N}(0, I)\right) &= \frac{1}{2}\left(\mathrm{Tr}(\Sigma) + \|\mu\|^2 - d + \log \frac{1}{\det(\Sigma)}\right) \\
&= \frac{1}{2}\left(d(1 - e^{-\int_0^T \beta(s)\,\mathrm{d}s}) + e^{-\int_0^T \beta(s)\,\mathrm{d}s}\|X_0\|^2 - d + d \log \frac{1}{1 - e^{-\int_0^T \beta(s)\,\mathrm{d}s}}\right) \\
&= \frac{1}{2}\left(e^{-\int_0^T \beta(s)\,\mathrm{d}s}\|X_0\|^2 - d\left(e^{-\int_0^T \beta(s)\,\mathrm{d}s} + \log(1 - e^{-\int_0^T \beta(s)\,\mathrm{d}s})\right)\right).
\end{aligned}$$

Since $\beta(t)$ is bounded away from zero, we have $\int_0^T \beta(s)\,\mathrm{d}s \to \infty$ as $T \to \infty$, so $e^{-\int_0^T \beta(s)\,\mathrm{d}s} \to 0$.

Moreover, the function $\phi(x) := x + \log(1 - x)$ satisfies $\phi(x) \geq 0$ for $x \in [0, \delta]$ with $\delta > 0$ sufficiently small. Hence, for T large enough, the expression

$$-d\left(e^{-\int_0^T \beta(s)\,\mathrm{d}s} + \log(1 - e^{-\int_0^T \beta(s)\,\mathrm{d}s})\right)$$

is non-positive, and we can write

$$\mathrm{KL}(p(T, \cdot \mid X_0) \| \mathcal{N}(0, I)) \leq \frac{1}{2} e^{-\int_0^T \beta(s)\,\mathrm{d}s}\|X_0\|^2.$$

Taking expectation over $X_0 \sim p_{\mathrm{data}}(\cdot)$, we get:

$$\mathrm{KL}(p(T, \cdot) \| p_{\mathrm{noise}}(\cdot)) \leq \frac{1}{2} e^{-\int_0^T \beta(s)\,\mathrm{d}s}\, \mathbb{E}_{p_{\mathrm{data}}(\cdot)}[\|X_0\|^2].$$

Applying Pinsker's inequality:

$$\mathrm{d_{TV}}(p(T, \cdot), p_{\mathrm{noise}}(\cdot)) \leq \exp\left(-\frac{1}{2}\int_0^T \beta(s)\,\mathrm{d}s\right) \sqrt{\frac{1}{2}\mathbb{E}_{p_{\mathrm{data}}(\cdot)}[\|X_0\|^2]}.$$

Substituting this bound into Theorem 5.3 concludes the proof of Corollary 5.5. □

B.3 Proof of Theorem 5.7

We begin by recalling the following classical inequality:

Proposition B.5 (Generalized Grönwall's inequality)**.** *Let $y : [0,T] \to \mathbb{R}$ be a continuously differentiable function satisfying*

$$\frac{\mathrm{d}}{\mathrm{d}t}y(t) \leq a(t)\, y(t) + b(t),$$

for all $t \in [0,T]$, where $a : [0,T] \to \mathbb{R}$ and $b : [0,T] \to \mathbb{R}$ are continuous functions. Then,

$$y(T) \leq y(0)\, \exp\left(\int_0^T a(s)\,\mathrm{d}s\right) + \int_0^T b(s)\, \exp\left(\int_s^T a(r)\,\mathrm{d}r\right)\,\mathrm{d}s.$$

To prove Theorem 5.7, we use a coupling argument. Let (U_t) and (V_t) be solutions of the following coupled SDEs.

$$\begin{cases} \mathrm{d}U_t = \big(-f(T-t,U_t) + g^2(T-t)\nabla \log p(T-t,U_t)\big)\,\mathrm{d}t + g(T-t)\,\mathrm{d}B_t, \\ \mathrm{d}V_t = \big(-f(T-t,V_t) + g^2(T-t)s_\theta(T-t,V_t)\big)\,\mathrm{d}t + g(T-t)\,\mathrm{d}B_t, \end{cases}$$

where (U_0, V_0) is an optimal coupling between $p(T,\cdot)$ and $p_{\text{noise}}(\cdot)$ for the Wasserstein-2 distance. We define

$$\Delta_t := \mathbb{E}[\|U_t - V_t\|^2],$$

and aim to bound Δ_T, since we remark that

$$W_2^2(p_{\text{data}}(\cdot), \mathcal{L}(Y_T)) \leq \Delta_T.$$

Applying Itô's formula to $\|U_t - V_t\|^2$, we obtain:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\|U_t - V_t\|^2\right] &= 2\,\mathbb{E}\left[\langle U_t - V_t,\ -f(T-t,U_t) + f(T-t,V_t)\rangle\right] \\ &\quad +2\,\mathbb{E}\left[\langle U_t - V_t,\ g^2(T-t)\left(\nabla \log p(T-t,U_t) - s_\theta(T-t,V_t)\right)\rangle\right]. \end{aligned}$$

We split the second term as:

$$\begin{aligned} \nabla \log p(T-t,U_t) - s_\theta(T-t,V_t) = \nabla \log p(T-t,U_t) - \nabla \log p(T-t,V_t) \\ + \nabla \log p(T-t,V_t) - s_\theta(T-t,V_t). \end{aligned}$$

Thus, the derivative becomes:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\|U_t - V_t\|^2\right] &= -2\,\mathbb{E}\left[\langle U_t - V_t, f(T-t,U_t) - f(T-t,V_t)\rangle\right] \\ &\quad + 2g^2(T-t)\,\mathbb{E}\left[\langle U_t - V_t,\ \nabla \log p(T-t,U_t) - \nabla \log p(T-t,V_t)\rangle\right] \\ &\quad + 2g^2(T-t)\,\mathbb{E}\left[\langle U_t - V_t,\ \nabla \log p(T-t,V_t) - s_\theta(T-t,V_t)\rangle\right]. \end{aligned}$$

We now bound each term :

- By Hypothesis 5.6(1), we have

$$\langle U_t - V_t,\ f(T-t,U_t) - f(T-t,V_t)\rangle \geq r_f(T-t)\|U_t - V_t\|^2.$$

- By Hypothesis 5.6(2), we use the Lipschitz continuity of $\nabla \log p$, and Cauchy–Schwarz:

$$\begin{aligned}&\langle U_t - V_t,\ \nabla \log p(T-t, U_t) - \nabla \log p(T-t, V_t)\rangle \\ &\quad \le \|U_t - V_t\| \cdot \|\nabla \log p(T-t, U_t) - \nabla \log p(T-t, V_t)\| \\ &\quad \le L\|U_t - V_t\|^2.\end{aligned}$$

- Using Young's inequality:

$$\begin{aligned}&\langle U_t - V_t,\ \nabla \log p(T-t, V_t) - s_\theta(T-t, V_t)\rangle \\ &\quad \le h\|U_t - V_t\|^2 + \frac{1}{4h}\left\|\nabla \log p(T-t, V_t) - s_\theta(T-t, V_t)\right\|^2.\end{aligned}$$

Taking expectations and combining, we get :

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta_t \le \left(-2r_f(T-t) + 2Lg^2(T-t) + 2hg^2(T-t)\right)\Delta_t + \frac{g^2(T-t)}{2h}\,\varepsilon^2,$$

where we used Hypothesis 5.2 (i.e., $\|s_\theta - \nabla \log p\|_\infty \le \varepsilon$).

Define the function

$$u(t) := \int_{T-t}^{T} \left(-2r_f(s) + (2L+2h)g^2(s)\right)\mathrm{d}s.$$

so that the inequality becomes:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta_t \le \Delta_t\, u'(t) + \frac{\varepsilon^2}{2h}\, g^2(T-t).$$

We can now apply Grönwall's inequality to get:

$$\Delta_T \le \Delta_0\, e^{u(T)} + \frac{\varepsilon^2}{2h}\int_0^T g^2(T-t)\, e^{u(T)-u(t)}\,\mathrm{d}t.$$

Since $\Delta_0 = W_2^2(p(T,\cdot), p_{\text{noise}}(\cdot))$, we conclude:

$$W_2^2(p_{\text{data}}(\cdot), \mathcal{L}(Y_T)) \le W_2^2(p(T,\cdot), p_{\text{noise}}(\cdot))\, e^{u(T)} + \frac{\varepsilon^2}{2h}\int_0^T g^2(t) e^{u(T)-u(T-t)}\mathrm{d}t.$$

□

B.4 Proof of Corollary 5.10

Lemma B.6 (Squared Wasserstein-2 distance between Gaussians)**.** *Let $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ be two Gaussian distributions in $\mathbb{R}^d$, with Σ_1, Σ_2 symmetric positive definite. Then,*

$$W_2^2\left(\mathcal{N}(\mu_1, \Sigma_1),\ \mathcal{N}(\mu_2, \Sigma_2)\right) = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2}\right)^{1/2}\right).$$

We adapt the argument from Theorem 5.7, replacing the use of Lipschitz continuity with a bound derived from the strong log-concavity of the data distribution.

Let (U_t, V_t) be the coupled processes defined as follows:

$$\mathrm{d}U_t = \big(-f(T-t,U_t) + g^2(T-t)\nabla \log p(T-t,U_t)\big)\,\mathrm{d}t + g(T-t)\mathrm{d}B_t,$$
$$\mathrm{d}V_t = \big(-f(T-t,V_t) + g^2(T-t)s_\theta(T-t,V_t)\big)\,\mathrm{d}t + g(T-t)\mathrm{d}B_t,$$

where $f(t) = -\frac{1}{2}\beta(t)$, $g(t) = \sqrt{\beta(t)}$, and s_θ is the estimated score.

We define $\Delta_t := \mathbb{E}[\|U_t - V_t\|^2]$. The computation of $\frac{\mathrm{d}}{\mathrm{d}t}\Delta_t$ follows the same steps as in Theorem 5.7, with the exception of the term :

$$\langle U_t - V_t,\ \nabla \log p(T-t,U_t) - \nabla \log p(T-t,V_t)\rangle.$$

In Theorem 5.7, this term was controlled using Lipschitz continuity. Here, we assume that $p_{\text{data}}(\cdot)$ is κ-strongly log-concave, and in the case of the VP setting, according to Proposition 7 from [5], this implies that $\log p(T-t,\cdot)$ is $a(T-t)$-strongly concave, with

$$a(T-t) := \kappa\left(e^{-\int_0^{T-t}\beta(s)\,ds} + \kappa\int_0^{T-t} e^{-\int_s^{T-t}\beta(v)\,dv}\beta(s)\,ds\right)^{-1}.$$

Hence, $\log p(T-t,\cdot)$ satisfies the inequality

$$\langle x-y,\ \nabla \log p(T-t,x) - \nabla \log p(T-t,y)\rangle \le -a(T-t)\|x-y\|^2.$$

Therefore, we obtain this inequality :

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta_t \le -2r_f(T-t)\Delta_t - 2g^2(T-t)a(T-t)\Delta_t + 2hg^2(T-t)\Delta_t + \frac{\varepsilon^2}{2h}g^2(T-t).$$

Define:

$$u_{\text{VP}}(t) := \int_{T-t}^{T}\big(-2r_f(s) - 2a(s)g^2(s) + 2hg^2(s)\big)\,ds = \int_{T-t}^{T}\beta(s)\,(1+2h-a(s))\,ds$$

which gives :

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta_t \le \Delta_t u'_{\text{VP}}(t) + \frac{\varepsilon^2}{2h}g^2(T-t).$$

Applying Grönwall's inequality:

$$\Delta_T \le \Delta_0 e^{u_{\text{VP}}(T)} + \frac{\varepsilon^2}{2h}\int_0^T g^2(t)e^{u_{\text{VP}}(T)-u_{\text{VP}}(T-t)}dt.$$

Now observe that for all $s \in [0,T]$,

$$a(s) = \frac{\kappa}{e^{-\int_0^s\beta(v)\,dv} + \kappa\int_0^s e^{-\int_v^s\beta(u)\,du}\beta(v)\,dv} = \frac{\kappa}{\kappa + (1-\kappa)e^{-\int_0^s\beta(v)\,dv}} \ge \min(\kappa,1),$$

using the fact that $0 \le e^{-\int_0^s\beta(v)\,dv} \le 1$.

This implies :

$$u_{\text{VP}}(T) \le \int_0^T \beta(s)(1+2h-2\min(\kappa,1))\,ds \le \beta_{\max}(1+2h-2\min(\kappa,1))T,$$
$$u_{\text{VP}}(T) - u_{\text{VP}}(T-t) \le \beta_{\max}(1+2h-2\min(\kappa,1))t.$$

Bounding the initial Wasserstein distance. We use convexity of the Wasserstein-2 distance in its first argument:

$$W_2^2(p(T,\cdot), p_{\text{noise}}(\cdot)) \leq \mathbb{E}_{X_0 \sim p_{\text{data}}(\cdot)} \left[W_2^2(p(X_T \mid X_0), \mathcal{N}(0, I)) \right].$$

Given that $X_T \mid X_0 \sim \mathcal{N}(\mu_T(X_0), \Sigma_T)$ with:

$$\mu_T(X_0) = e^{-\frac{1}{2}\int_0^T \beta(s)\,ds} X_0, \qquad \Sigma_T = \left(1 - e^{-\int_0^T \beta(s)\,ds}\right) I,$$

we obtain from Lemma B.6:

$$W_2^2(p(T,\cdot), p_{\text{noise}}(\cdot)) = \mathbb{E}_{X_0 \sim p_{\text{data}}(\cdot)} \left[e^{-\int_0^T \beta(s)\,ds} \|X_0\|^2 + d\left(2 - e^{-\int_0^T \beta(s)\,ds} - 2\sqrt{1 - e^{-\int_0^T \beta(s)\,ds}}\right)\right].$$

Using the approximation

$$1 - \sqrt{1-x} \leq x, \quad \text{for } x \in [0,1],$$

with $x = e^{-\int_0^T \beta(s)\,ds}$

we find :

$$W_2^2(p(T,\cdot), p_{\text{noise}}(\cdot)) \leq e^{-\int_0^T \beta(s)\,ds} \mathbb{E}_{p_{\text{data}}(\cdot)}[\|X_0\|^2] + d \cdot e^{-\int_0^T \beta(s)\,ds}.$$

Using $\int_0^T \beta(s)\,ds \geq \beta_{\min} T$, we obtain:

$$\Delta_0 \leq W_2^2(p(T,\cdot), p_{\text{noise}}(\cdot)) \leq e^{-\beta_{\min} T} \mathbb{E}_{p_{\text{data}}(\cdot)}[\|X_0\|^2] + d \cdot e^{-\int_0^T \beta(s)\,ds}.$$

Final bound. Combining all bounds, we get:

$$\begin{aligned}\Delta_T \leq &\left(e^{-\beta_{\min} T} \mathbb{E}_{p_{\text{data}}(\cdot)}[\|X_0\|^2] + d \cdot e^{-\int_0^T \beta(s)\,ds}\right) \cdot e^{\beta_{\max}(1+2h-2\min(\kappa,1))T} \\ &+ \frac{\varepsilon^2}{2h} \int_0^T \beta(T-t) e^{\beta_{\max}(1+2h-2\min(\kappa,1))t}\,dt.\end{aligned}$$

For $\kappa > \frac{1}{2}$, we choose $h < \min(\kappa, 1) - \frac{1}{2}$, hence $(1 + 2h - 2\min(\kappa, 1)) < 0$, and we obtain:

$$\begin{aligned}W_2^2(p_{\text{data}}(\cdot), \mathcal{L}(Y_T)) \leq &\, e^{-(\beta_{\min} + \beta_{\max}(2\min(\kappa,1) - 1 - 2h))T} \left(\mathbb{E}_{p_{\text{data}}(\cdot)}[\|X_0\|^2] + o(e^{-\beta_{\min} T})\right) \\ &+ \frac{\varepsilon^2}{2h} \cdot \frac{1}{2\min(\kappa,1) - 1 - 2h}.\end{aligned}$$

This completes the proof □

B.5 Proof of Corollary 5.11

We follow the same approach as in Appendix B.4, applying Theorem 5.7 to the CVP SDE, whose drift and diffusion coefficients are given by

$$f(t) = \frac{1}{2}\beta(t), \qquad g(t) = \sqrt{\beta(t)}.$$

The associated noise distribution is Gaussian, given by

$$p_{\text{noise}}(\cdot) \sim \mathcal{N}\left(0,\ \left(e^{\frac{T}{2}(\beta_{\max} + \beta_{\min})} - 1\right) I\right).$$

We define, for any $t \in [0, T]$,

$$u_{\mathrm{CVP}}(t) := \int_{T-t}^{T} \beta(s) \left(-1 + 2h - \frac{2\kappa}{e^{\int_0^s \beta(v)\,dv} + \kappa \int_0^s e^{\int_v^s \beta(u)\,du} \beta(v)\,dv} \right) ds,$$

Using Theorem 5.7, we obtain

$$\Delta_T := W_2^2(p(T,\cdot), p_{\mathrm{noise}}(\cdot)) \le \Delta_0\, e^{u_{\mathrm{CVP}}(T)} + \frac{\varepsilon^2}{2h} \int_0^T \beta(t)\, e^{u_{\mathrm{CVP}}(T) - u_{\mathrm{CVP}}(T-t)}\, dt.$$

We now analyze each term in this expression.

Upper bound on $u_{\mathrm{CVP}}(T)$. We expand:

$$u_{\mathrm{CVP}}(T) = \int_0^T \beta(s)\,(-1+2h)\; ds - 2\kappa \int_0^T \frac{\beta(s)}{e^{\int_0^s \beta(v)\,dv} + \kappa \int_0^s e^{\int_v^s \beta(u)\,du} \beta(v)\,dv}\, ds.$$

The first term satisfies

$$\int_0^T \beta(s)(-1+2h)\, ds \le -\int_0^T \beta(s)\, ds + 2\beta_{\max} h T.$$

For the second term, we use the identity

$$\frac{1}{e^{\int_0^s \beta(v)\,dv} + \kappa \int_0^s e^{\int_v^s \beta(u)\,du} \beta(v)\,dv} = \frac{1}{(1+\kappa) e^{\int_0^s \beta(v)\,dv} - \kappa},$$

which leads to the inequality

$$\frac{1}{e^{\int_0^s \beta(v)\,dv} + \kappa \int_0^s e^{\int_v^s \beta(u)\,du} \beta(v)\,dv} \ge \frac{1}{\kappa+1} e^{-\int_0^s \beta(v)\,dv}.$$

Therefore, we obtain

$$\int_0^T \frac{\beta(s)}{e^{\int_0^s \beta(v)\,dv} + \kappa \int_0^s e^{\int_v^s \beta(u)\,du} \beta(v)\,dv}\, ds \ge \frac{1}{\kappa+1} \int_0^T \beta(s)\, e^{-\int_0^s \beta(v)\,dv}\, ds.$$

We now use the fact that

$$\int_0^T \beta(s)\, e^{-\int_0^s \beta(v)\,dv}\, ds = 1 - e^{-\int_0^T \beta(v)\,dv},$$

which directly implies

$$\int_0^T \beta(s)\, e^{-\int_0^s \beta(v)\,dv}\, ds \ge 1 - e^{-\beta_{\min} T}.$$

Thus,

$$u_{\mathrm{CVP}}(T) \le -\int_0^T \beta(s)\, ds + 2\beta_{\max} h T - \frac{2\kappa}{\kappa+1} \left(1 - e^{-\beta_{\min} T}\right).$$

Upper bound on $u_{\mathrm{CVP}}(T) - u_{\mathrm{CVP}}(T-t)$. By definition,

$$\begin{aligned} u_{\mathrm{CVP}}(T) - u_{\mathrm{CVP}}(T-t) &= \int_{T-t}^{T} \beta(s) \left(-1 + 2h - \frac{2\kappa}{e^{\int_0^s \beta(v)\,dv} + \kappa \int_0^s e^{\int_v^s \beta(u)\,du} \beta(v)\,dv} \right) ds \\ &\leq \int_{T-t}^{T} \beta(s)(2h-1)\,ds \\ &\leq \beta_{\max}(2h-1)t. \end{aligned}$$

Upper bound on Δ_0. In the CVP setting, the conditional law of X_T given X_0 is Gaussian and admits the following closed form:

$$X_T \mid X_0 \sim \mathcal{N}\left(X_0\, e^{\frac{1}{2}\int_0^T \beta(s)\,ds},\ \left(e^{\int_0^T \beta(s)\,ds} - 1 \right) I \right).$$

Let $\mu_T := e^{\frac{1}{2}\int_0^T \beta(s)\,ds}$ and $\sigma_T^2 := e^{\int_0^T \beta(s)\,ds} - 1$, and denote by $p_{\mathrm{noise}} = \mathcal{N}(0, \sigma_T^2 I)$ the reference distribution.

Using Lemma B.6 and the convexity of the squared Wasserstein distance, we obtain:

$$\Delta_0 := W_2^2(p(T,\cdot), p_{\mathrm{noise}}(\cdot)) \leq \mathbb{E}_{p_{\mathrm{data}}(\cdot)} \left[W_2^2 \left(\mathcal{N}(\mu_T X_0, \sigma_T^2 I),\ \mathcal{N}(0, \sigma_T^2 I) \right) \right] = \mu_T^2\, \mathbb{E}_{p_{\mathrm{data}}(\cdot)}[\|X_0\|^2].$$

Thus,

$$\Delta_0 \leq e^{\int_0^T \beta(s)\,ds}\, \mathbb{E}_{p_{\mathrm{data}}(\cdot)}[\|X_0\|^2].$$

Conclusion. Combining all bounds, we obtain the inequality

$$W_2^2(p_{\mathrm{data}}(\cdot), \mathcal{L}(Y_T)) \leq e^{-2\left(\frac{\kappa}{1+\kappa} - \beta_{\max} h T + \mathcal{O}(e^{-\beta_{\min} T})\right)}\, \mathbb{E}_{p_{\mathrm{data}}(\cdot)}[\|X_0\|^2] + \frac{\varepsilon^2}{2h(1-2h)},$$

which completes the proof. □