POLITECNICO DI TORINO

Master's Degree
in Data Science and Engineering

Master Thesis

Towards more stable continuous-time functional diffusion processes

Supervisors
prof. Tatiana Tommasi
prof. Pietro Michiardi

Candidate
Alberto Foresti

Academic Year 2023-2024

Summary

Continuous-time functional diffusion processes demonstrated great potential in the generation of resolution-invariant data and in generalising diffusion models to different data types. However, training these models is challenging due to the high number of hyperparameters and the instability of the training process. Meta-learning is the dominant approach for training this kind of models. Two sets of parameters are used, where one specialises to the task at hand, while the other is computed at inference time to adapt the network for the current datum through few iterations of stochastic gradient descent. In this thesis, we propose a more stable approach to infer functional representations of data and avoid the pitfalls of meta-learning. We employ a different neural network to infer the set of parameters that specialises to the datum. This allows to preserve important properties, such as resolution invariance in case of visual tasks.
Additionally, we derive a new functional Stochastic Differential Equation (SDE) that is more stable and requires fewer hyperparameters. Specifically, we remove the drift term by showing that the infinitesimal generator can be set to the zero operator without losing the guarantee for the existence of the reverse diffusion process. Moreover, we show that it is possible to estimate the covariance operator of the Brownian term of the diffusion process directly from data as the empirical covariance of the dataset.
The proposed approach allows to reduce the number of hyperparameters, leading to a faster, simpler and cheaper training process. Remarkably, this new functional SDE resembles the variance exploding SDE, sharing similar properties, such as the unbounded variance in the forward diffusion process in the limit of infinite time.
Together, the two main contributions of this work provide a more stable and efficient way to train continuous-time functional diffusion processes. We validate the proposed method on a set of experiments, showing that it is more stable and requires fewer computational resources compared to the state of the art of continuous-time functional diffusion processes.
The implementation of this project was done using the *PyTorch* library, expanding and translating the codebase of the paper *Continuous-Time Functional Diffusion Processes*, originally written using *Jax*. Future work will be done to demonstrate the ability of the proposed method to scale to more complex datasets and bigger architectures.

Acknowledgements

Vorrei ringraziare i miei genitori, i miei zii, le mie nonne e i miei parenti tutti per il loro supporto e la loro fiducia in me nel corso degli anni.
Un ringraziamento speciale va a mia madre, che mi ha sempre sostenuto e si è sempre interessata in ogni mia scelta, anche tecnica, nonostante faccia tutt'altro nella vita. Ringrazio in egual misura mio padre, che con la sua esperienza sul campo mi ha sempre dato i giusti consigli e mi ha aiutato a trovare la strada giusta. Più volte mi sono ispirato a lui per dedizione e approccio ai problemi.
Ringrazio Giulia per avermi supportato in questi anni a Torino e in Francia, per avermi sopportato nei momenti di stress e per avermi rallegrato in tutti i momenti passati insieme. Grazie anche per essere stata un'ottima agente immobiliare.
Ringrazio i professori Pietro Michiardi e Giulio Franzese per avermi accolto in Francia con così poco preavviso e per avermi guidato in questo progetto e per aver permesso di potermi esprimere al meglio.
Ringrazio la professoressa Tatiana Tommasi per avermi dato le giuste dritte da Torino e per essermi stata vicino nella ricerca del progetto di tesi giusto per me.
Ringrazio i miei amici, in particolare Giulia, per aver condiviso l'ufficio con me e per avermi tirato su quando gli esperimenti non funzionavano. Rimarranno memorabili le nostre colazioni in ufficio e la settimana enigmistica dopo pranzo. Un grazie anche a Shaheer, per esserci stato dal primo giorno e per tutto il nostro percorso insieme tra progetti ed esami difficili. Ringrazio, quindi, tutti i miei amici a Torino per aver reso indimenticabile la mia esperienza universitaria. Un grazie anche agli amici che ho trovato in costa azzurra, per aver creato tanti ricordi in un erasmus comunque troppo corto e ai miei compagni dell'ASP, abbiamo trascorso poco tempo insieme ma ci siamo sempre divertiti un sacco. Ringrazio, infine tutti gli amici e mentori che ho trovato nel corso della mia vita a Savona e Gressoney, per avermi insegnato tanto e per avermi fatto crescere come persona. Un ringraziamento particolare va ad Elia per avermi accolto a casa sua durante la quarantena e a Luca, per le nostre passeggiate sul lungomare che ci tenevano sani durante la pandemia. Ringrazio in generale gli studenti del PoliTo per la comunità online che mi ha permesso di confrontarmi con altri colleghi e studiare meglio, specialmente durante il 2020 e il 2021.
Infine, un grazie agli asini dell'Eurecom, per tutte le volte in cui vi ho dato da mangiare e per tutte le volte in cui mi avete fatto compagnia uscendo dall'università.

Contents

List of Tables

List of Figures

If you cannot understand my
argument, and declare
it's Greek to me
you are quoting Shakespeare.

[B. Levin, Quoting Shakespeare]

Chapter 1

Introduction

In the last few years, we witnessed a surge in the development of generative Artificial Intelligence (AI) systems. Notably, the release of chatGPT to the public in late 2022 brought increasingly higher attention on the field. From that date, however, generative AI did not stop progressing, with the release of more advanced models like GPT-4 or DALL-E. Interestingly, the models pioneering the generative AI wave focused on textual tasks. This could sound unintuitive to a biologist, as in the history of the evolution of life forms on earth, language was a capability that developed solely in human and, in any case, after the development of vision. While this can motivate the different nature of intelligence in humans and AI, it also raises the question of whether generative AI can be applied to visual tasks or other different types of tasks. Remarkably, recent success in vision, like the release of DALL-E or SORA, suggests that the current paradigms of generative AI can be successfully applied to different fields. However, the models behind these successes are still not general enough to be applied to any kind of task. In fact, while humans can learn to perform any kind of task with limited information in a reasonable time due to millions of years of evolution, AI models can only rely on statistical correlations from huge pools of data produced and selected by humans. This affects the generalization capabilities of AI models, as they are bound to the inductive bias of the data they are trained on. For this reason, models such as GPT-4 fail to accomplish simple tasks that do not appear often in the data they have been trained on, such as solving the graph colouring problem Stechly et al. [2023].

1.1 Escaping the inductive bias

Diffusion based generative modelling emerged as a promising paradigm versatile enough to be bounded to any kind of data. Due to its reliance on mathematical assumptions and constructs such as Partial Differential Equations, there is stronger theoretical evidence compared to Generative Adversarial Networks (GANs), which rely on intuitions based on engineering. At the core of diffusion, however, lie deep learning architectures, such as transformers and Convolutional Neural Networks (CNNs). These architectures allow diffusion models to exploit the inductive bias for better performances. For instance, in

computer vision, CNNs offer a way to encode the information of the spatial structure of the image through convolution kernels. However, this kind of approach does not generalise well to different data types. This does not limit to transferring the architecture from different kind of media, such as image to audio, but also in scaling image models to different resolutions.
Representing a single data point as a function show promise as a way to generalise diffusion models, as functions are mathematical objects and do not rely on real world assumptions. This method, however, comes with several practical challenges. Firstly, functions are, by definition, infinite dimensional, as they are elements of a Hilbert space, whereas computer memory is finite.
Fortunately, we have several tools to deal with that issue. Most notably, Multi-Layer Perceptrons (MLPs) are universal function approximators Hornik et al. [1989], meaning that they can approximate any function to any degree of precision. Additionally, Fourier analysis allows us to exactly represent certain functions in a finite dimensional space, by decomposing them in a sum of sinusoidal functions, even with a limited amount of samples of the same function. In practice, however, stability in training diffusion models using functional representation is not guaranteed, and they can be hard to train, especially compared to traditional architectures such as CNNs and Vision Transformers (ViTs).
Additionally, considering the joint impact of the set of hyperparameters related to the functional representation network and the hyperparameter of the SDE of the diffusion model can cause a lot of trouble as:

1. It is not clear how the two sets of hyperparameters affect each other.
2. The training of the model can be unstable and require a lot of computational resources.

1.2 Contribution

In this work, we propose a novel approach to train diffusion models using functional representation on top of the work proposed by Franzese et al. [2024]. We start by illustrating the intuition with the help of a simple example: generating images at different resolutions. For this task, given a set of images, we can represent them using a functional representation. In this work, we treat each image as a function, however we maintain a shared core function representation, and then we specialise it by using another network to compute a set of parameters that modifies the core function. In this way we can enhance stability in training and exploit the inductive bias of the additional network. However, this comes with an increased complexity of the training process. For this reason, in this work we reason on reducing the number of hyperparameters of the SDE of the diffusion process for a simpler and cheaper training process. This work is structured as follows:

- In Chapter 2, we provide the mathematical foundations required to understand the results of this work.
- In Chapter 3, we provide a review of the state of the art in diffusion models and functional representation.

- In Chapter 4, we introduce the proposed method and the intuition behind it.
- In Chapter 5, we provide the results of the experiments conducted to validate the proposed method.
- In Chapter 6, we provide the conclusions and future work.

Chapter 2

Mathematical Foundations

Generative modelling takes inspiration from work in probability theory, measure theory, and functional analysis. In this chapter, we provide the mathematical foundations required to understand the results of this work. In particular, we start with mainstream material, leading the way to advanced concepts which will form the backbone of the work.

2.1 Measure theory

In this section, we provide the basics of measure theory for understanding the results of this work. The definitions in this section come from Salamon [2020] unless stated otherwise. We start by defining the notion of measurable space and sigma-algebra:

Definition 1 (Measurable space) *A* σ***-algebra*** $\mathcal{A}$ *on a set* X *is a collection of subsets of* X *such that:*

- $X \in \mathcal{A}$.
- *If* $A \in \mathcal{A}$, *then* $A^c \in \mathcal{A}$.
- *Every countable union of elements in* $\mathcal{A}$ *is also in* $\mathcal{A}$.

A ***measurable space*** *is a pair* $(X, \mathcal{A})$, *where* $\mathcal{A} \subset 2^X$ *is a* σ*-algebra.*

We follow by defining the notion of topology Hatcher [2009]:

Definition 2 (Topological space) *A topological space is a set* X *together with a collection* O *of subsets of* X, *called open sets, such that:*

- *The union of any collection of sets in* O *is in* O.
- *The intersection of any finite collection of sets in* O *is in* O.
- *Both* X *and* $\emptyset$ *are in* O.

O *is called a* ***topology*** *on* X.

A σ-algebra is called a Borel σ-algebra if it is generated by the open sets of a topological space. Moving forward, we introduce the concept of measurable maps:

Definition 3 (Measurable Map) *First, we define the* ***pre-image*** $f^{-1}(B)$ *of a set* B *under a function* $f : X \to Y$ *as the set of elements in* X *that are mapped to* B *by* f*, denoted as* $f^{-1}(B)$*. Then, a map* $f : X \to Y$ *is called measurable if for any measurable* $B \subset Y$*, then* $f^{-1}(B)$ *is a measurable subset.*

These definitions are useful for Lebesgue integrals. Informally, Lebesgue integrals use approximations of functions, called **simple functions** or **step functions**. Those are real-valued functions that admit only a finite number of values, that is, they have finite image. An important result of step functions is that a function $f : X \to [0, \infty]$ is measurable if and only if there exists a sequence of step functions s_n that converge to it: $\lim_{n\to\infty} s_n(x) = f(x)\ \forall x \in X$. We now have the elements to define the measure:

Definition 4 (Measure) *Let* $(X, \mathcal{A})$ *be a measurable space. A measure on* $(X, \mathcal{A})$ *is a function* $\mu : \mathcal{A} \to [0, \infty]$ *such that:*

- μ *is* σ*-additive:* $\mu(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$ *for any countable collection of disjoint sets* $\{A_i\}_i$*.*
- *There exists a measurable set* $A \in \mathcal{A}$ *such that* $\mu(A) < \infty$*.*

A ***measure space*** *is a triple* $(X, \mathcal{A}, \mu)$ *consisting of a measurable set, its* σ*-algebra and a measure.*

An important example of measure space is the **probability space**, where the measure of the entire space is one. More formally, we give the following definition:

Definition 5 (Probability Space) *A probability space is a measure space* $(\Omega, \mathcal{F}, P)$*, where:*

- *The set* Ω *is the sample space.*
- *The* σ*-algebra* $\mathcal{F} \subset 2^{\Omega}$ *is the set of events.*
- *The measure* $P : \mathcal{F} \to [0,1]$ *is the probability measure. It satisfies* $P(\Omega) = 1$*.*

We can now use all these elements to define Lebesgue integrals:

Definition 6 (Lebesgue Integral) *Given a measure space* $(X, \mathcal{A}, \mu)$ *and a measurable set* E*, let* $s : X \to [0, +\infty)$ *be a measurable step function of the form:*

$$s(x) = \sum_{i=1}^{n} a_i \chi_{A_i}(x) \tag{2.1}$$

With α_i *a positive real number and* $A_i \in \mathcal{A}$*. The Lebesgue integral of* s *over* E *is the number defined as:*

$$\int_E s d\mu = \sum_{i=1}^{n} \alpha_i \mu(A_i \cap E) \tag{2.2}$$

Additionally, let $f : X \to [0, +\infty)$ be a measurable function. The Lebesgue integral of f over E is the number defined as:

$$\int_E f d\mu = \sup \left\{ \int_E s d\mu : s \leq f, s \text{ simple} \right\} \tag{2.3}$$

We say that a function $f : X \to \mathbb{R}$ is **integrable** with respect to measure μ if it is measurable and if $\int |f| d\mu < +\infty$.
The extension to non-negative functions is straightforward, as we can define the integral of a function f as the difference between the integral of its positive and negative parts. A random variable $X : \Omega \to \mathbb{R}$ is an integrable function, by taking the integral over the entire space we obtain the expected value of X.

Completion of measure spaces A measurable space $(X, \mathcal{A}, \mu)$ is called **complete** if $N \in \mathcal{A}, \mu(N) = 0, A \subset N \implies A \in \mathcal{A}$. When a space is not complete, we can complete it by adding the necessary sets to the σ-algebra. More formally:

Theorem 1 *Let $(X, \mathcal{A}, \mu)$ be a measure space. Then we define a new σ-algebra:*

$$\mathcal{A}^* = \{E \subset X | \text{ there are measurable sets } A, B \in \mathcal{A} \text{ such that } A \subset E \subset B \wedge \mu(B \setminus A) = 0\} \tag{2.4}$$

*Then there is a unique measure $\mu^* : \mathcal{A}^* \to [0, +\infty]$ such that $\mu^*|_{\mathcal{A}} = \mu$. Moreover, μ^* preserves also integration, that is, if f is μ-integrable, then it is also μ^*-integrable and the integrals coincide. The triple $(X, \mathcal{A}^*, \mu^*)$ is a measure space and is called the **completion** of $(X, \mathcal{A}, \mu)$.*

L^p spaces In this work, we will often consider integrable functions. In this paragraph we formally define a space of integrable functions and generalise the notion of integrability. Given a measure space $(X, \mathcal{A}, \mu)$ a measurable function $f : \mathcal{A} \to \mathbb{R}$ is said to be p-integrable if:

$$\left(\int_X |f|^p d\mu \right)^{\frac{1}{p}} < +\infty \ \forall x \in \mathcal{A} \tag{2.5}$$

This integral is also called the **p-norm**, and it is denoted as $||f||_p$. The space of p-integrable functions is denoted as $L^p(\mu)$.

The Lebesgue measure The Lebesgue measure is particularly interesting, because it appears frequently with Euclidean spaces. To define the Lebesgue measure we define the property of **translation invariance** of a measure μ, that is $\mu(A+x) = \mu(A) \ \forall A \in \mathcal{A}, x \in X$. Where $A + x \coloneqq \{a + x : a \in A\}$ and $\mathcal{A}$ is the σ-algebra of all Borel sets in $\mathbb{R}^n$. It is important to denote that there is a unique measure $\mu : \mathcal{B} \to \mathbb{R}^n$ that is translation invariant and satisfies $\mu([0,1]^n) = 1$.

Definition 7 (Lebesgue Measure) *If we denote $(\mathbb{R}^n, \mathcal{B}, \mu)$ as the measure space of the measure defined earlier, then the completion of the measure space is the **Lebesgue measure space** $(\mathbb{R}^n, \mathcal{A}, \mu)$ and μ is the **Lebesgue measure**.*

More practically, the Lebesgue measure allows us to assign the value one to all unit cubes in $\mathbb{R}^n$.

The Radon-Nikodym Derivative A measurable space X is said to be **σ-finite** if it can be expressed as the union of countably many subsets with finite measures, that is:

$$X = \bigcup_{i=1}^{\infty} X_i, \text{ with } \mu(X_i) < \infty \ \forall i \tag{2.6}$$

For example, the Euclidean space is σ-finite with the Lebesgue measure because it is the union of countably many cubes, or balls, with finite volume. We now state an important theorem that allows us to relate different measures, we will use it later in the work:

Theorem 2 (Radon-Nikodym Theorem) *Let $(X, \mathcal{A}, \mu)$ be a σ-finite measure space and ν a measure on $\mathcal{A}$ such that μ is absolutely continuous with respect to ν. Then there exists a measurable function $f : X \to [0, +\infty)$ such that:*

$$\nu(A) \coloneqq \int_A f d\mu \ \forall A \in \mathcal{A} \tag{2.7}$$

*The function f is called the **Radon-Nikodym derivative** of ν with respect to μ and is denoted as $\frac{d\nu}{d\mu}$.*

The reverse implication is also valid, that this if we can find f then μ is absolutely continuous with respect to ν, but we will use the forward implication in this work.

2.2 Functional analysis

In this section we provide the basics of functional analysis required to understand the results of this work. The definition of this section follow the work Bühler [2017]. We start by defining *Metric Spaces*, which provide the fundamentals for Hilbert spaces and Banach spaces.

Definition 8 (Metric space) *A metric space is a pair (X, d), where X is a set and $d : X \times X \to \mathbb{R}$ is the **distance function** with the following properties:*

- *$d(x, y) \geq 0 \ \forall x, y \in X$, with equality if and only if $x = y$.*
- *$d(x, y) = d(y, x) \ \forall x, y \in X$.*
- *$d(x, y) \leq d(x, z) + d(z, y) \ \forall x, y, z \in X$ (**triangle inequality**).*

A **Cauchy sequence** over a metric space is a sequence $\{x_n\}_n$ such that for every $\epsilon > 0$ there exists an N such that $d(x_n, x_m) < \epsilon \ \forall n, m > N$. A metric space is **complete** if every Cauchy sequence converges to a point in the space.

Definition 9 (Normed vector space) *A normed vector space is a pair $(X, ||\cdot||)$, where X is a real vector space and $||\cdot|| : X \to \mathbb{R}$ is a function called norm with the following properties:*

- $||x|| \geq 0 \ \forall x \in X$, *with equality if and only if* $x = 0$.
- $||\alpha x|| = |\alpha|||x|| \ \forall x \in X, \alpha \in \mathbb{R}$.
- $||x + y|| \leq ||x|| + ||y|| \ \forall x, y \in X$ (***triangle inequality***).

A complete normed vector space is called a ***Banach space****.*

This definition plays a crucial role for Hilbert spaces, which only need an additional condition on their norm:

Definition 10 (Hilbert space) *A Hilbert space is a real or complex inner product space that is also a complete metric space with respect to the distance function induced by the inner product.*

The inner product of two vectors x and y in a Hilbert space $\mathcal{H}$ is denoted by $\langle x, y \rangle$. The norm of a vector x is defined as $\|x\| = \sqrt{\langle x, x \rangle}$.

The concept of Banach space allows us to define bounded linear operators. The term linear operator is an alias for linear map, which is a function between two vector spaces that preserves the operations of vector addition and scalar multiplication.

Definition 11 (Bounded Linear Operator) *Let* $(X, || \cdot ||_X)$ *and* $(Y, || \cdot ||_Y)$ *be two normed vector spaces, then a linear operator* A *is said to be bounded if there exists a constant* C *such that* $||Ax||_Y \leq C||x||_X \ \forall x \in X$.

The smallest constant C that satisfies the inequality is called the **operator norm** of A and is denoted by $||A||$.

An operator A is said to be **positive definite** if it satisfies:

$$\langle x, Ax \rangle \geq 0 \ \forall x \in X \tag{2.8}$$

Where $\langle \cdot, \cdot \rangle$ is the inner product of the Hilbert space X.
Note that, for a positive definite operator A, we can also define the square root $A^{\frac{1}{2}}$ as the operator that satisfies $A^{\frac{1}{2}} A^{\frac{1}{2}} = A$. Similarly, we can define the inverse of an operator A as the operator A^{-1} that satisfies $AA^{-1} = A^{-1}A = I$. Combining these two we can define $A^{-\frac{1}{2}}$ as the inverse of the square root of A.
We can also define kernel and image of a linear operator:

- The **kernel** of a linear operator A is the set of vectors x such that $Ax = 0$.
- The **image** of a linear operator A is the set of vectors y such that there exists a vector x such that $Ax = y$.

They resemble the concepts of null space and column space in linear algebra, with an extension to infinite dimensions. Analogously, we can also define the **trace** of an operator $A : X \to Y$:

$$\mathrm{tr}(A) = \sum_n \langle e_k, Ae_k \rangle \tag{2.9}$$

The trace is well-defined only when it is unique for any choice of base. Operators for which a trace can be defined are said to be trace-class operators. The space of bounded linear operators $X \to Y$ is denoted by $\mathcal{L}(X, Y)$ and $(\mathcal{L}(X, Y), ||\mathcal{L}||_{(X,Y)})$ forms a normed vector space.
Remarkably, we can relate the operators of two different spaces through the definition of adjoint operator:

Definition 12 *Let $A : X \to Y$ be a bounded linear operator between two Hilbert spaces X and Y. The adjoint operator $A^* : Y \to X$ is the unique operator that satisfies:*

$$\langle Ax, y\rangle_Y = \langle x, A^* y\rangle_X \ \ \forall x \in X, y \in Y \tag{2.10}$$

Adjoint operators will allow us to work with solutions of Partial Differential Equations (PDEs) more concisely.

2.2.1 Semicontinuous Semigroups

We now delve into a class of operators of paramount importance for the solutions of PDEs: semigroups.

Definition 13 (Strongly continuous semigroup) *Let X be a real Banach space, then a **one-parameter semigroup** of operators on X is a map $S : [0, +\infty) \to \mathcal{L}(X)$ such that for all $s, t \geq 0$:*

$$S(0) = I, \text{ the identity operator} \tag{2.11}$$
$$S(s + t) = S(s)S(t) \tag{2.12}$$

*A **strongly continuous semigroup** is a semigroup with the following property:*

$$\lim_{t\to 0} ||S(t)x - x|| = 0 \ \forall x \in X \tag{2.13}$$

We can also define the continuous version of semicontinuous semigroup, the only difference is that their domain is $\mathbb{R}$ and not $[0, +\infty)$. In this work, we will use the family of operators $S(t) = e^{At} = \sum_{k=0} \frac{t^k A^k}{k!}$, where $A : X \to X$ is a bounded linear operator. Strongly continuous semigroups answer to the task of finding a solution for a linear differential equation, the inverse problem is answered by infinitesimal generators:

Definition 14 (Inifnitesimal Generator) *Let X be a Banach space and let S be a strongly continuous semigroup. The infinitesimal generator of S is the operator $\mathcal{A} : dom(\mathcal{A}) \to X$, where $dom(\mathcal{A}) = \left\{x \in X | \lim_{t\to 0} \frac{s(t)x - x}{t} \text{ exists}\right\}$, such that:*

$$\lim_{t\to 0} \frac{S(t)x - x}{t} = \mathcal{A}x \ \forall x \in dom(\mathcal{A}) \tag{2.14}$$

2.3 Stochastic calculus

In this section, we provide the basics of stochastic calculus required to understand the results of this work. We follow the theoretical foundations laid out by Särkkä and Solin [2019]. Throughout this section, we assume random variables with values in $\mathbb{R}^m$ with $0 < m < +\infty$ finite. In general, we can write a Stochastic Differential Equation (SDE) as follows:

$$dX_t = f(X_t, t) + g(X_t, t)dW_t \tag{2.15}$$

Where:

- X_t is the random variable that characterises the random process described by 2.15.
- $f : \mathbb{R}^m \times [0, T] \to \mathbb{R}^m$ is a function called **drift coefficient**.
- $g : [0, T] \to \mathbb{R}^m$ is a function called **diffusion coefficient**.
- dW_t is a Brownian motion.

In practice, dW_t can be treated as infinitesimal white noise, more formally:

Definition 15 (Brownian motion) *The Brownian motion $W_t \in \mathbb{R}^m$ is a stochastic process characterised by a diffusion matrix $R \in \mathbb{R}^m \times \mathbb{R}^m$ such that:*

1. *Any increment $\Delta W_{s-t} = W_s - W_t$ with $s > t > 0$ is a Gaussian random variable with zero mean and covariance $R(s - t)$*
2. *$W_0 = 0$.*
3. *Increments are independent if their time intervals do not overlap.*

The Brownian motion is a fundamental object in stochastic calculus, it is the continuous-time analogue of a random walk. Moreover, Brownian motions cannot look into the future, but are only conditioned by the past. This property is formalized by the concept of filtration:

Definition 16 (Filtration) *Given a probability space $(\Omega, \mathcal{F}, P)$, a filtration $\mathcal{F}_t$ is a family of σ-algebras such that:*

- *$\mathcal{F}_t \subset \mathcal{F}_s$ for $t < s$.*
- *$\bigcup_t \mathcal{F}_t = \mathcal{F}$.*

A stochastic process is said to be adapted to the filtration $\mathcal{F}_t$ if it is $\mathcal{F}_t$-measurable for all $t \geq 0$. In this way, we can formalize the intuition that Brownian motions do not see into the future. In fact, we can define a filtration with respect to the time index such that the Brownian motion is adapted to it.

The scalar case of Brownian motion with diffusion matrix $R = 1$ is called standard Brownian motion. Since Brownian motions are nowhere differentiable, most definitions of integrals do not work. We must appositely define a new kind of integral for this case:

Definition 17 (Itô integral) *Given $g : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ and a standard Brownian motion W_t the Itô integral $\int_0^T g(X_t,t)dW_t$ is the number:*

$$\int_0^T g(X_t,t)dW_t = \lim_{N\to\infty} \sum_k g(X_{t_k},t_k)W_{t_k}(t_{k+1}-t_k) \tag{2.16}$$

Where $\{t_k\}_{k\in[1,2,\dots,N]}$ is a sequence of times in increasing order.

This formulation is similar to a Riemann integral, however Riemann integrals allow any choice of $t_k^* \in [t_k, t_{k+1}]$ for the computation of $g(X_{t_k^*}, t_k^*)$. In this case, the Brownian motion would be too irregular for such a series to converge. This shows the importance of fixing the sequence $\{t_k\}_{k\in[1,2,\dots,N]}$.
As an implication of this definition, Itô calculus has a different chain rule. For example, consider the following integral:

$$\int_0^t W_t dW_t = \frac{(W_t)^2}{2} - \frac{t}{2} \neq \frac{(W_t)^2}{2} \tag{2.17}$$

In general, the chain rule in Itô calculus has the following form:

Theorem 3 (Itô's formula) *Given a Itô process x_t, then given an arbitrary scalar function $\phi(x_t,t)$ the differential $d\phi$ is:*

$$d\phi = \frac{\delta\phi}{\delta t}dt + \sum_i \frac{\delta\phi}{\delta x_i}dx_i + \frac{1}{2}\sum_{i,j}\frac{\delta^2\phi}{\delta x_i \delta x_j}dx_i dx_j \tag{2.18}$$

Where mixed derivatives are coupled with the rules:

- $dW_t dW_t^T = Rdt$.
- $dtdW = 0$.
- $dWdt = 0$.

This formula extends to the non-scalar case as it holds to each component of a multi-dimensional Itô process. In any case any Itô process can be expressed as the solution of the SDE $dX_t = f(X_t,t)dt + g(X_t,t)dW_t$. The probability density $p(x,t)$ of the solution of the equation in 2.15, solves the following equation, called **Fokker-Planck-Kolmogorov (FPK)** equation:

$$\frac{\delta p(x,t)}{\delta t} = -\sum_i \frac{\delta}{\delta x_i}[f_i(x,t)p(x,t)] + \frac{1}{2}\sum_{i,j}\frac{\delta^2}{\delta x_i \delta x_j}\{[g(x,t)Qg(x,t)^T]_{ij}p(x,t)\} \tag{2.19}$$

The FPK equation allows to define the equivalent probability flow ordinary differential equation (ODE) in diffusion processes Song et al. [2020]. The solution of the FPK equation solves also another equation called **backward Kolmogorov equation**:

$$-\frac{\delta p(y,t|x,s)}{\delta s} = \mathcal{A}p(y,t|x,s), p(y,s|x,s) = \delta(x-y) \tag{2.20}$$

Where $t \geq s$ and $\mathcal{A}$ is an infinitesimal generator satisfying:

$$\mathcal{A}\phi = -\sum_i \frac{\delta}{\delta x_i}[\phi] f_i(x,t) + \frac{1}{2}\sum_{i,j} \frac{\delta^2}{\delta x_i \delta x_j}\{[g(x,t)Qg(x,t)^T]_{ij} p(x,t)\} \tag{2.21}$$

It may also be of practical interest to find a solution to 2.19. In this case, we can rely on the Kac-Feynman formula:

Theorem 4 (Kac-Feynman formula) *Let X_t be the solution of the SDE in 2.15 and $p(x,t)$ the solution of the FPK equation in 2.19. Then the solution of the following equation:*

$$u(x,t) = E[\phi(X_t)|X_0 = x] \tag{2.22}$$

is given by:

$$u(x,t) = \int_{\mathbb{R}^m} \phi(y) p(y,t|x,0) dy \tag{2.23}$$

2.4 The diffusion equation

In this work, we are interested in the diffusion equation, which is a simplified case of equation 2.15, where the diffusion term only depends on time:

$$dX_t = f(X_t,t)dt + g(t)dW_t \tag{2.24}$$

The FPK equation allows to find the drift term of a PDE that has the same distribution at final time as the reverse-time SDE.

2.4.1 Reversing the diffusion equation

Given a forward diffusion equation, an important problem consist on inverting it so that we can recover a stochastic process that goes back in time. In his work Anderson [1982] Anderson provides a formula using the Bayes theorem and the FPK equation:

$$p(x_t,t,x_s,s) = p(x_s,s|x_t,t)p(x_t,t), \text{ with } s \geq t \tag{2.25}$$

Then, by manipulating the FPK equation, the reverse time process associated to the forward process in eq.2.15 is:

$$d\bar{X}_t = \bar{f}(\bar{X}_t,t)dt + g(\bar{X}_t,t)d\bar{W}_t \tag{2.26}$$

Where $\bar{W}_t$ is a Wiener process independent of past increments of X_t and, equivalently, of future increments of $\bar{X}_t$. Whereas:

$$\bar{f}(X_t,t) = f(X_t,t) - \nabla_x[\log p(X_t,t) g(X_t,t) g^T(X_t,t)] g(X_t,t) g^T(X_t,t) \tag{2.27}$$

In the case of the diffusion equation, we get that g only depends on time. Hence, we can simplify:

$$\bar{f}(X_t,t) = f(X_t,t) - g^2(t)\nabla_x[\log p(X_t,t)] \tag{2.28}$$

This last equation was used by Song and Ermon in their formulation of generative modelling through SDEs Song et al. [2020].

2.5 Girsanov's theorem

Suppose we are given two stochastic processes driven by:

$$dX_t = f_1(X_t, t)dt + dW_t, \text{ with } X_0 = x_0 \tag{2.29}$$
$$dY_t = f_2(Y_t, t)dt + dW_t, \text{ with } Y_0 = x_0 \tag{2.30}$$

With dW_t being a Wiener process with diffusion matrix R. Let us also define the trajectories up to time T: $\mathcal{X} = \{X_{t_1}, X_{t_2}, ..., X_{t_n}\}$, $\mathcal{Y} = \{Y_{t_1}, Y_{t_2}, ..., Y_{t_n}\}$ such that $\{t_k\}$ are in increasing order and become dense in time as $n \to +\infty$. These trajectories allow us to define **path integrals**:

$$E[h(\mathcal{X}_t)] = \int h(\mathcal{X}_t)p(\mathcal{X}_t)d\mathcal{X}_t \tag{2.31}$$

The ratio of the probability law of the two paths is given by:

$$\frac{p(\mathcal{X}_t)}{p(\mathcal{Y}_t)} = Z(t) \tag{2.32}$$

Where:

$$Z(t) = \exp\Bigg(-\frac{1}{2}\int_0^t [f_1(Y_\tau, \tau) - f_2(Y_\tau, \tau)]^T R^{-1}[f_1(Y_\tau, \tau) - f_2(Y_\tau, \tau)]d\tau + \int_0^T [f_1(Y_\tau, \tau) - f_2(Y_\tau, \tau)]^T R^{-1} dW_\tau \Bigg)$$

This allows us to relate path integrals in the following way:

$$E[h(\mathcal{X}_t)] = E[Z(t)h(\mathcal{Y}_t)] \tag{2.33}$$

We can also define an induced probability measure $\tilde{p}(\mathcal{X}_t) = Z(t)p(\mathcal{X}_t)$ such that the following process $\tilde{W}_t$ is a Brownian motion under $\tilde{p}$ with covariance matrix R:

$$\tilde{W}_t = W_t - \int_0^t f_1(Y_\tau, \tau) - f_2(Y_\tau, \tau)d\tau \tag{2.34}$$

The true Girsanov theorem is a generalization of what we have seen so far. In the following statements we also explicitate the dependence of our processes with respect to the event space.

Theorem 5 (Girsanov) *Let $\theta_t : [0, T] \times \Omega \to \mathbb{R}^N$ be a stochastic process adapted to the filtration $\{\mathcal{F}_t\}_{t\geq 0}$driven by the N-dimensional Brownian motion $W(\omega, t)$ under probability measure $\mathbb{P}$. Let's define the process Z_t:*

$$Z(t, \omega) = exp\left(\int_0^t \theta^T(\tau, \omega)dW(\tau, \omega) - \frac{1}{2}\int_0^t \theta^T(\omega, \tau)\theta(\omega, \tau)d\tau\right) \tag{2.35}$$

Satisfying $E[Z(t,\omega)] = 1$*, then the process:*

$$\tilde{W}(t,\omega) = W(t,\omega) - \int_0^t \theta(\omega,\tau) \tag{2.36}$$

Is a Brownian motion under probability measure $\tilde{\mathbb{P}}$ *derived via:*

$$E\left[\frac{d\tilde{\mathbb{P}}}{d\mathbb{P}}(\omega)|\mathcal{F}_t\right] = Z(t,\omega) \tag{2.37}$$

We call this theorem a generalization, because the process θ cannot always be derived as a likelihood ratio. Additionally, the ratio of probability densities $Z(\omega,t)$ is the Radon-Nikodym derivative of $\tilde{\mathbb{P}}$ with respect to $\mathbb{P}$.
We can use Girsanov's theorem to find the weak solution of a SDE, consider the following SDE:

$$dX_t = \theta(X_t,t)dt + dW_t \tag{2.38}$$

Then by Girsanov theorem the expectation of any function h of X_t can be computed as:

$$E[h(X_t)] = E[Z(t)h(\tilde{X}_t)] \tag{2.39}$$

Where $Z(t)$ is defined as in the theorem and $\tilde{X}_t$ is the solution of the SDE:

$$\begin{cases} d\tilde{X}_t &= X_0 + dW_t \\ \tilde{W}_t &= W_t - \int_0^t \theta(X_0 + W_\tau)d\tau \end{cases} \tag{2.40}$$

This system solves the SDE 2.38 under the new path measure $\tilde{\mathbb{P}} = Z(t)\mathbb{P}$.

2.6 Stochastic calculus in infinite dimensions

In this section, we show how to extend the concept we introduced previously to infinite dimensions following the foundations laid out by Da Prato and Zabczyk [2014]. We start again with the Brownian motion. When the Brownian motion is Hilbert-valued, the diffusion matrix is not suitable to describe it, as we have infinite dimensions. For this reason, we use a symmetric and non-negative operator R called **covariance operator**. Given a suitable orthonormal basis $\{e_k\}_k$ for our Hilbert space $\mathcal{H}$ we have the following property:

$$Re_k = \lambda_k e_k \text{ for } \lambda_k \geq 0 \text{ and bounded} \tag{2.41}$$

Using this operator we can proceed with the definition of Brownian motion:

Definition 18 (Brownian motion in Hilbert spaces) *A* $\mathcal{H}$*-valued stochastic process* W *with covariance operator* R *is said to be an* R*-Wiener process if:*

- $W_0 = 0$.
- W *has continuous trajectories.*

- *Non-overlapping increments are independent.*
- *Increments* $W(t) - W(s)$ *with* $s < t$ *are normally distributed* $\mathcal{N}(0, (t-s)R)$.

When R is trace-class, we have useful properties that can also allow us to practically implement infinite dimensional SDEs in code. In particular, the Brownian motion admits the following, convergent, series expansion:

$$W_t = \sum_{j=1}^{+\infty} \sqrt{\lambda_j} \beta_t^j e_j \tag{2.42}$$

Where $\beta_j = \frac{1}{\sqrt{\lambda_j}} \langle W_t, e_j \rangle$ for $j \in \mathbb{N}^*$. Those β_j are mutually independent, real-valued Wiener processes. For practical applications, we still have the problem that we have infinite dimensions. However, we can use Fourier analysis and the sampling theorem to use a finite number of basis vectors.

2.6.1 SDEs in infinite dimensions

Nonlinear SDEs in infinite dimensions read as follows:

$$\begin{cases} dX_t = (\mathcal{A}X_t + f(X_t, t))dt + g(X_t, t)dW_t \\ X_0 = \zeta \end{cases} \tag{2.43}$$

Where $\mathcal{A}$ is a bounded linear operator, and both f and g are nonlinear functions. The process assumes a probability space $(\Omega, \mathcal{F}, P)$, with the canonical filtration $\mathcal{F}_t$. Most notably, this kind of equations admit a unique mild solution under some assumptions, where the term mild solution is defined as follows:

Definition 19 (Mild solution) *Given a process described by 2.43, a mild solution is a process* X_t *such that:*

- $P\left(\int_0^t |X_s|^2 ds < +\infty\right) = 1$, $P -$ *almost surely*
- *For any* $t \in [0, T]$*:*

$$X(t) = S(t)\zeta + \int_0^t S(t-s) f(X_s, s) ds + \int_0^t S(t-s) g(X_s, s) dW_s \; P - \text{almost everywhere} \tag{2.44}$$

Where $S(t) = e^A t$ *is a strongly continuous semigroup of operators.*

The list of complete assumptions for existence and uniqueness of mild solutions is available in Da Prato and Zabczyk [2014]. Informally, we can already see from the definition of mild solution that $\mathcal{A}$ is required to be the infinitesimal generator of $S(t)$. The most practical assumptions also require f and g to be Lipschitz and have linear growth.
A practical way to get a solution of a SDE in infinite dimensions is to use Girsanov's theorem:

Theorem 6 (Girsanov in infinite dimensions) *Given a process θ with values in $U_0 = R^{-\frac{1}{2}}(U)$ adapted to the natural filtration $\{\mathcal{F}\}_{t\geq 0}$ such that:*

$$\mathbb{E}\left[exp\left(\int_0^T \langle \theta_s, dW_s\rangle_0 - \frac{1}{2}\int_0^T |\theta_s|_0^2 ds\right)\right] \tag{2.45}$$

Then, the process $\tilde{W}$:

$$\tilde{W}_t = W_t - \int_0^t \theta_s dW_s,\ t \in [0,T] \tag{2.46}$$

Is a Wiener process with covariance operator R under the probability measure $\tilde{\mathbb{P}}$ defined by:

$$\mathrm{d}\tilde{\mathbb{P}} = exp\left(\int_0^T \langle \theta_s, dW_s\rangle_0 - \frac{1}{2}\int_0^T |\theta_s|_0^2 ds\right)\mathrm{d}\mathbb{P} \tag{2.47}$$

As we can see, the theorem is similar to the finite dimensional case, with the major difference that the inner product and the norm are defined in the Hilbert space U_0. We can use Girsanov's theorem to compute a change of measure that allows us to solve the SDE in 2.43 with a simpler form, similarly for what is done for finite dimensions.

Chapter 3

Machine Learning and Generative modelling

This work builds upon techniques from machine learning and generative modelling. In this section, we provide a brief overview of the most important concepts and techniques used in this work. In particular, we delve into the pillars of deep learning and generative modelling, while exploring different interesting approaches from both a theoretical and practical point of view which can lead to fascinating research directions.

3.1 Variational autoencoders

Variational Autoencoders (VAEs) Kingma and Welling [2013] belong to the class of Deep Latent Variable Models (DLVMs). DLVMs work with latent variables, a latent variable is part of the model we employ for our task that cannot be observed, and it is not part of the dataset. Conversely, it can be derived with a mathematical model from the known variables. Denoting the variables in our dataset by x and the latent variables z, the task of DLVMs is to approximate the marginal distribution:

$$p_\theta(x) = \int p_\theta(z|x)p_\theta(x)dz \tag{3.1}$$

Unfortunately, $p_\theta(x)$ and the posterior $p_\theta(z|x)$ are intractable. In order to make them tractable an encoder, or **inference model**, q_ϕ is used to model $q_\phi(z|x) \sim p_\theta(z|x)$. The generative model learns a joint distribution $p(x, z)$ by factorising it into a prior distribution over the latent space $p(z)$ and a stochastic decoder $p_\theta(x|z)$.

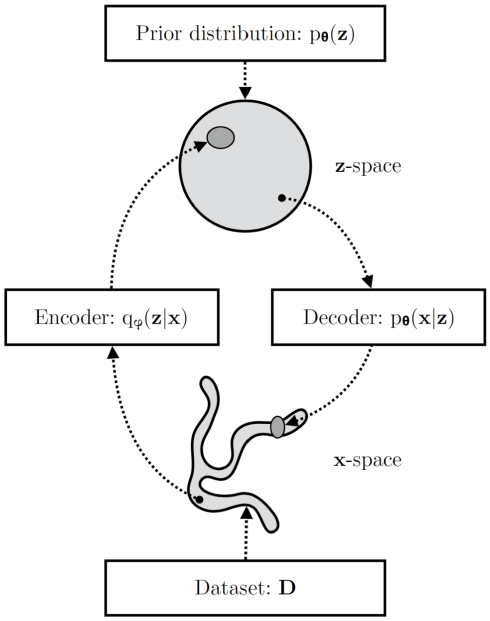

Figure 3.1: The working mechanism of the Variational Autoencoder Kingma et al. [2019]

3.1.1 Evidence Lower Bound

VAEs are optimised using the Evidence Lower BOund (ELBO). This quantity can be derived from the likelihood of the model as follows:

$$\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)q_\phi(z|x)}{q_\phi(z|x)p_\theta(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{\text{ELBO: } \mathcal{L}_{\phi,\theta}(x)} + \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]}_{D_{KL}(q_\phi(z|x)||p_\theta(z|x))}
\end{aligned}$$

In the second term D_{KL} is the Kullback-Leibler divergence, which measures a distance between two distributions. Hence, this term is quite straightforward as it measures the divergence between the inference model and the true posterior. The ELBO is more complicated to explain, but we can rearrange it with the following terms:

$$\mathcal{L}_{\phi,\theta}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] = \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \leq \log p_\theta(x) \quad (3.2)$$

Hence, it represents a lower bound to the likelihood of the model. Maximising the ELBO improves both the generative model and the inference model. There are, however, some difficulties in optimising both θ and ϕ at the same time. While for θ we get:

$$\nabla_\theta(\mathcal{L}_{\theta,\phi}(x) = \nabla_\theta \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z)\right] - \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z|x)\right]) \simeq \nabla_\theta \log p_\theta(x,z) \tag{3.3}$$

Doing the same for ϕ is more difficult since the expectation is taken with respect to a function of ϕ itself. In order to make it possible, the reparametrisation trick is used:

Theorem 7 (Reparametrization trick) *Given an inference model $q_\phi(z|x)$, a function of the latent variables $f(z)$ and random noise $\epsilon \sim p(\epsilon)$ such that $z = g(\phi, \epsilon, x)$, then:*

$$\nabla E_{q_\phi(z|x)}[f(z)] = \nabla_\phi E_{p(\epsilon)}[f(z)] = E_{p(\epsilon)}[\nabla_\phi f(z)] \simeq \nabla_\phi f(z) \tag{3.4}$$

In practice, the random component is moved to another variable so that it is possible to decouple gradient and expectation.

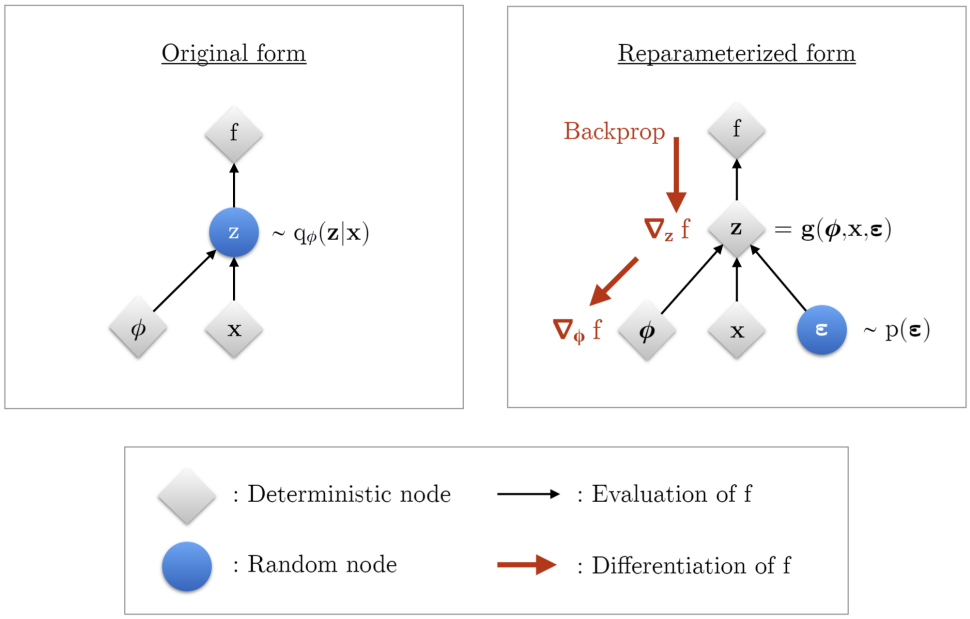

Figure 3.2: Visualization of the reparametrisation trick Kingma et al. [2019]

3.2 Generative Adversarial Networks

Generative Adversarial Networks Goodfellow et al. [2014] are a class of generative models that work by training two neural networks simultaneously. One network is called the **generator**, and it is responsible for generating new data points. The other network is called the **discriminator**, and it is responsible for distinguishing between real and

generated data points. The two networks are trained in a minimax game, where the generator tries to generate data points that are indistinguishable from real data points, while the discriminator tries to distinguish between real and generated data points. The objective function of the GAN is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.5}$$

Where $p_{\text{data}}(x)$ is the distribution of the real data points and $p_z(z)$ is the distribution of the noise that the generator G uses to generate new data points. The GAN objective function is non-convex and it is difficult to optimise. In practice, the GAN is trained using a two-step process:

1. The discriminator is trained to maximise the objective function.
2. The generator is trained to minimise the objective function.

This process is repeated until the generator is able to generate data points that are indistinguishable from real data points.

3.3 Learning with diffusion models

Diffusion processes can be used to learn to sample from a target distribution Sohl-Dickstein et al. [2015]. Consider a diffusion process, modelled by the probability distribution $p_\theta(x_0)$, where, by elementary probability:

$$p_\theta(x_0) = \int p(x_0|x_{[1:T]})dx_{[1:T]} \tag{3.6}$$

This is called the reverse process, and it is the process that generates the initial condition x_0 given the final condition x_T. We can assume that the reverse process behaves as a Markov chain, hence we can write:

$$p_\theta(x_0) = \int p(x_0|x_1)p(x_1|x_2)...p(x_{T-1}|x_T)dx_{[1:T]} \tag{3.7}$$

If the marginal distribution at time T is normally distributed:

$$p_\theta(x_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{3.8}$$

Then we can know the distribution of the conditional probabilities as well:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \sigma^2_\theta(x_t,t)) \tag{3.9}$$

Where $\mu_\theta(x_t,t)$ and $\sigma_\theta(x_t,t)$ are the mean and variance of the distribution at time t. Similarly, we have the following form for the posterior of the distribution process:

$$q(x_{t-1}|x_t,x_0) = \mathcal{N}(x_{t-1}; \sqrt{1-\beta_t}x_t, \beta_t\mathbf{I}) \tag{3.10}$$

Where β_t is the variance of the noise injected in the diffusion model from step $t-1$ to step t. This kind of model can be trained by minimising the following objective:

$$L \coloneqq E_q[-\log p_\theta(x_0)] \leq E_q\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] \tag{3.11}$$

Additionally, if we set $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, we can directly sample from the diffusion process at any time t by using the following distribution:

$$q(x_t|x_0) \sim \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{3.12}$$

This notation also helps in rewriting the training objective to make it more interpretable in terms of marginal distributions:

$$E_q\left[\underbrace{D_{KL}\left(q(\mathbf{x}_T \mid \mathbf{x}_0) \parallel p(\mathbf{x}_T)\right)}_{L_T} + \sum_{t>1} \underbrace{D_{KL}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right)}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}_{L_0}\right] \tag{3.13}$$

In particular, this objective is tractable since we know the distribution of the posterior $q(x_{t-1}|x_t, x_0)$, and we can sample from it:

$$q(x_{t-1}|x_t, x_0) \sim \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t\mathbf{I}) \tag{3.14}$$

Where:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t(1-\overline{\alpha}_{t-1})}}{1-\overline{\alpha}_t}x_t \qquad \tilde{\beta}_t = \frac{1-\overline{\alpha}_t}{1-\overline{\alpha}_{t-1}}\beta_t \tag{3.15}$$

3.4 Score-based diffusion models

In 2019 Song and Ermon introduced a new class of generative models based on score matching and Langevin dynamics Song and Ermon [2019]. The **score** of a probability distribution $p(x)$ is the gradient of the log-density of the distribution:

$$\mathbf{s}(x) = \nabla_x \log p(x) \tag{3.16}$$

In the score-matching task, the objective is to train a score $\mathbf{s}_\theta$ network which is able to minimise the following objective:

$$\frac{1}{2}\mathbb{E}_{p(x)}[||\mathbf{s}_\theta(x) - \nabla_x \log p(x)||^2] \tag{3.17}$$

This formulation, however, is intractable, as we do not know the true score of the distribution. In order to circumvent this problem, it is possible to estimate the true gradient using small Gaussian noise perturbations. In this way, we have a posterior distribution $q_\sigma(\tilde{x}|x)$ and for small noise scales $q_\sigma \sim p$. Using these elements, it has been proven that the following formulation is equivalent to 3.17 for small noise scales:

$$\frac{1}{2}\mathbb{E}_{q(\tilde{x}|x),p(x)}[||\mathbf{s}_\theta(\tilde{x}) - \nabla_x \log q(\tilde{x}|x)||^2] \tag{3.18}$$

In fact the optimal score network that minimises 3.18 achieves $\mathbf{s}_{\theta^*}(x) = \nabla_x \log q_\sigma(x)$, which is very similar to $\nabla_x \log p(x)$ for small σ.

Langevin Sampling In order to sample new data points we can use Langevin dynamics, starting from a prior distribution $\pi(x)$, usually pure noise, and iterating the following procedure:

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(x) + \sqrt{\epsilon}\mathbf{z} \tag{3.19}$$

Where ϵ is a step size and $\mathbf{z} \sim \mathcal{N}(0,1)$. For a sufficiently large number of steps T and a sufficiently small ϵ, we can guarantee $\tilde{x}_T \sim p(x)$ with a negligible error. As we do not know the true value of the score, in practice we use a score network to approximate it.

Drawbacks of unconditional score matching Score matching as we have described so far suffers from the following drawbacks:

- **Manifold hypothesis**: the manifold hypothesis states that data lie in lower dimensional space with respect to its representation. This does not allow score networks to be consistent.

- **Lower density region**: lower density regions are not well represented, as a consequence the score network cannot represent the score accurately.

- **Slow mixing**: the score network cannot recover the weights of components of mixed distributions. Consider $p(x) = (1-a)p_1(x) + ap_2(x)$, where p_1 and p_2 have disjoint support, then $\nabla_x \log p(x) = \nabla_x \log p_1(x) + \nabla_x \log p_2(x)$.

Song and Ermon found that adding random Gaussian noise was beneficial to these problematics as adding noise to data helped to lift data to high dimensions, covered low density regions and helped to mix disjoint mixtures. Building upon this intuition, they devised noise-conditional score networks, which estimate the score of the perturbed distribution given the noise scale σ. By learning a score network which approximated $\mathbf{s}_\theta(x,\sigma) \sim \nabla_x \log q_\sigma(x)$, they were able to sample from the true distribution $p(x)$ by using Langevin dynamics with diminishing noise scales $\{\sigma_i\}_{i=1\ldots N}$. For this reason, the new objective to minimise becomes a weighted sum of terms similar to the one introduced in 3.18:

$$\sum_i \lambda(i)\mathbb{E}_{q_\sigma(\tilde{x})}[||\mathbf{s}_\theta(\tilde{x},\sigma) - \nabla_x \log q_\sigma(\tilde{x})||^2] \tag{3.20}$$

Where $\lambda(i)$ is a weighting factor. The work Song and Ermon [2019] presented the following algorithm for using Langevin dynamics with noise-conditional score networks 1.

Algorithm 1 Annealed Langevin Dynamics

procedure AnnealedLangevinDynamics($\sigma_1, \sigma_2, ..., \sigma_N, \epsilon, T$)
  $\tilde{x}_0 \sim \mathcal{N}(0,1)$
  for $i = 1$ to N **do**
    $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_N^2}$
    for $t = 1$ to T **do**
      $\tilde{x}_t = \tilde{x}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_\theta(\tilde{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0,1)$
    end for
    $\tilde{x}_0 = \tilde{x}_T$
  end for
  return $\tilde{x}_T$
end procedure

This methodology for generative modelling will be called *Score matching with Langevin dynamics* (SMLD).

3.4.1 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPM) Ho et al. [2020] are a class of generative models based on the principles of score-based generative modelling. They offer a reparametrisation of the original work Song and Ermon [2019] by parametrizing the posterior of the generative process as:

$$p_\theta(x_{t-1}|x_t) \sim \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t)) \tag{3.21}$$

In particular, they set $\sigma_\theta(x_t, t) = \sigma_t^2\mathbf{I}$ as not trained parameters, while keeping $\mu_\theta(x_t, t)$ trainable. Using this reparametrisation, they found that the following objective is equivalent to the objective highlighted in 3.13:

$$\mathbb{E}_q\left[||\frac{1}{2\sigma_t^2}\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2\right] + C \tag{3.22}$$

Where:

- C is a constant independent of θ.
- $\tilde{\mu}_t(x_t, x_0)$ is the posterior mean of the forward process, defined as in 3.15.
- $q(x_{t-1}|x_t, x_0) \sim \mathcal{N}(x_t; \sqrt{1-\sigma_t^2}x_t, \sigma_t^2\mathbf{I})$

Moreover, if we expand terms we can rewrite the objective as:

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \mu_\theta\left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t\right)\right\|^2\right] \tag{3.23}$$

We can notice that we only need to predict the noise term, since x_t is already available. Hence, we can simply train a noise prediction model ϵ_θ as follows:

$$\mathbb{E}_{p(x),t}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon-\epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t)\right\|^2\right] \tag{3.24}$$

However, in practice, the weighting factor is discarded, and the training objective is just a mean square error over the noise predictions. Additionally, this simplified version relates strongly with score matching as they both try to perform denoising at different time scales.

3.4.2 Score based generative modelling through SDEs

In their work Song et al. [2020], Song and Ermon provided the foundations for using SDEs to model generative processes. The intuition lies its fundamental in modelling the noising process of diffusion models through SDEs by using the diffusion equation:

$$dX_t = f(X_t,t)dt + g(t)dW_t \tag{3.25}$$

By reversing the diffusion equation is then possible to obtain a generative process:

$$dY_t = (f(Y_t,T-t) - g^2(t)\nabla_x \log p(Y_t))dt + g(t)d\tilde{W}_t \tag{3.26}$$

Theoretically, Y_t represent a reverse process with respect to X_t, in the sense that $\lim_{T\to\infty} Y_T \sim X_0$ However, the score $\nabla_x \log p_(X_t)$ is an unknown quantity. This is where score matching comes into play. The authors propose to parametrise the score using a time-dependent neural network $\mathbf{s}_\theta$:

$$dY_t = (f(Y_t,T-t) - g^2(t)\mathbf{s}_\theta(X_t,t))dt + g(t)d\tilde{W}_t \tag{3.27}$$

The score network is trained by explicit score matching, including a time-weighting function $\lambda(t)$:

$$\theta^* = \arg\min_\theta E_t\left\{\lambda(t)E_{X_0}E_{X_t|X_0}[||\mathbf{s}_\theta(X_t,t) - \nabla_x \log p_{0t}(X_t|X_0)||_2^2]\right\} \tag{3.28}$$

More informally, we try to minimise the error in the score for all choices of initial condition and all choices of time. In practice, to obtain the ground truth for the score, we simulate the path of the SDE by sampling a training point, a training time and diffuse the training point with the noise according to the SDE path.
In their work, Song and Ermon have also shown that the noise schedules used in DDPM and SMLD correspond to the discretization of two different SDEs. For example, the DDPM performs perturbation with the following Markov chain:

$$x_{i+1} = \sqrt{\beta_i}x_i + \sqrt{1-\beta_i}\mathbf{z}, \quad \mathbf{z}\sim\mathcal{N}(0,I) \tag{3.29}$$

In continuous time, this corresponds to the following SDE:

$$dX_t = \frac{1}{2}\beta(t)X_t dt + \sqrt{\beta(t)}dW_t \tag{3.30}$$

This SDE is called **variance preserving** (VP) sde.

Similarly, for SMLD, the perturbation and the corresponding SDE are:

$$x_{i+1} = x_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, I) \tag{3.31}$$

$$dX_t = \sqrt{\frac{d\sigma^2(t)}{dt}}dW_t \tag{3.32}$$

This last SDE is called **variance exploding** (VE) sde. Additionally, the authors proposed a new type of SDE, called sub-VP sde where the variance of the associated stochastic process is always upper-bounded by the corresponding variance of the VP sde:

$$dX_t = -\frac{1}{2}\beta(t)X_t dt + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)ds})}dW_t \tag{3.33}$$

The advantage of this type of SDEs is that the perturbation kernel can be computed in closed form, allowing for an efficient training. In this way, during training we can perturb our samples in a single step and denoise them to compute the gradients for backpropagation.

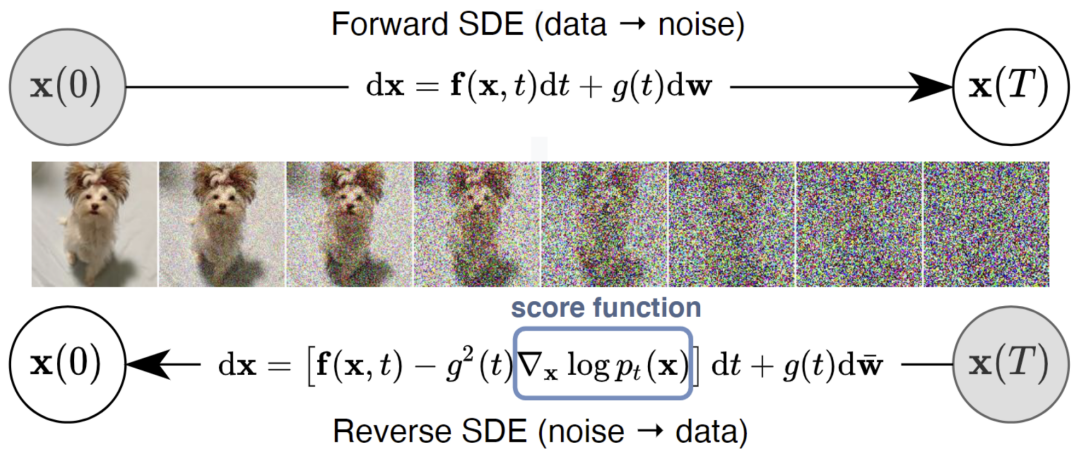

Figure 3.3: An example of diffusion process Song et al. [2020]

3.4.3 Sampling from the generative model

Once we trained a score network, we need to solve the reverse SDE to sample from the generative model. Several algorithms have been proposed to solve this problem, such as the Euler-Maruyama method, the Milstein method and the Runge-Kutta method.

Euler-Maruyama method

The Euler-Maruyama method Kloeden and Platen [2013] is a simple and efficient method to solve SDEs. In particular, given the SDE:

$$dX_t = f(X_t, t)dt + g(X_t, t)dt \tag{3.34}$$

And a discretization $\{t_k\}_{k\in[1,N]}$, we can discretize the SDE as follows:

$$X_{k+1} = X_n + f(X_k, t_k)(t_{k+1} - t_k) + g(X_k, t_k)(W_{k+1} - W_k) \tag{3.35}$$

Notice that we can set time steps to be equal, $t_{k+1} - t_k = \Delta t$. As a consequence the increments of the Brownian motion will be identically independently distributed $\Delta W \sim \mathcal{N}(0, \Delta t)$. Hence, we can rewrite:

$$X_{k+1} = X_n + f(X_k, t_k)\Delta t + g(X_k, t_k)\Delta W \tag{3.36}$$

This allows us to solve the SDE in a recursive manner, starting from the initial condition X_0. For the random increments we can use random number generators to sample from the normal distribution.

Milstein's method

For simplicity, we discuss this method in the case of a monodimensional SDE. Considering the same SDE as 3.34, the Milstein method Kloeden and Platen [2013] augment the Euler method by adding a term to the discretization:

$$X_{k+1} = X_k + f(X_k, t_k)\Delta t + g(X_k, t_k)\Delta W + \frac{1}{2}g(X_k, t_k)g'(X_k, t_k)(\Delta W^2 - \Delta t) \tag{3.37}$$

Where $g'(X_k, t_k)$ is the derivative of g with respect to X. The additional correction term, helps the algorithm converge faster than the Euler method.

Runge-Kutta method

The Runge-Kutta method Kloeden and Platen [2013], again builds on top of the Euler method:

$$X_{k+1} = X_k + f(X_k, t_k)\Delta t + g(X_k, t_k)\Delta W + \frac{1}{\sqrt{2\Delta t}}(g(t_k, \gamma_k) - g(t_k, X_k))(\Delta W^2 - \Delta t) \tag{3.38}$$

Where $\gamma_k = X_k + \alpha\Delta t + g(X_k, t_k)\sqrt{\Delta t}$ and α is a parameter. This method is again more accurate than the Euler method.

3.4.4 Solving the reverse diffusion SDE

In their work, Song and Ermon proposed a PC method to solve the reverse SDE Song et al. [2020]. Given a reverse diffusion SDE of the form:

$$dX_t = [f(X_t, t) - g(X_t, t)g^T(X_t, t)\mathbf{s}_\theta(X_t, t)]dt + g(t)d\tilde{W}_t \tag{3.39}$$

And its discretization:

$$X_{k-1} = X_k - [f(X_k, t_k) - g(X_k, t_k)g^T(X_k, t_k)\mathbf{s}_\theta(X_k, t_k)]\Delta t + g(X_k, t_k)\Delta\tilde{W} \tag{3.40}$$

We can use the Predictor Corrector (PC) method to generate new samples by solving the SDE. It involves two steps:

- A prediction step, which can be done with one of the methods illustrated before, like the Euler method.
- An iterative application of a correction step.

In particular, for the correction step Song and Ermon employed Langevin Markov Chain Monte Carlo (MCMC) sampling as in alg. 1. We illustrate an example of the PC method in alg. 2.

Algorithm 2 Predictor-Corrector Sampling for VP SDE

procedure PCSAMPLING($X_0, \epsilon, \beta, g, \mathbf{s}_\theta, \Delta t, T$)
  $X \leftarrow X_0$
  for $k = N$ to 0 **do**
    $X_{k-1} \leftarrow (2 - \sqrt{1 - \beta_k})X_k + \beta_k \mathbf{s}_\theta(X_k, t_k)$ ▷ Prediction step
    for $i = 1$ to N **do**
      $\mathbf{z} \sim \mathcal{N}(0, I)$
      $X_{k-1} \leftarrow X_{k-1} + \epsilon_k \mathbf{s}_\theta(X_{k-1}, t_{k-1}) + \sqrt{2\epsilon_k}\mathbf{z}$ ▷ Correction step
    end for
  end for
  return X
end procedure

It must be denoted that experiments Song et al. [2020] have shown that, while predictor-only algorithms perform well, corrector-only algorithms tend to present remarkably worse performances. The best performance is obtained by the joint application of prediction and correction steps.

3.4.5 A variational perspective on score-based generative modelling

In this work Huang et al. [2021], seek to understand how minimising a score-matching loss, like $\mathcal{L}_{ESM} = E[\frac{1}{2}||\mathbf{s}_\theta(X_t, t) - \nabla_x \log p(X_t)||^2]$ will impact the so-called plug-in reverse SDE:

$$dX_t = (gg^T \nabla_x \mathbf{s}_\theta(X_t, t) - f)dt + gdW_t \tag{3.41}$$

In particular, this analysis is carried out with the objective $X \sim Y$, where Y solves the diffusion SDE:

$$dY = f(Y_t, t)dt + g(t)d\tilde{W}_t \tag{3.42}$$

The authors provide a framework for estimating the likelihood of diffusion-based generative models. In particular, given a Itô SDE:

$$dX_t = \mu(X_t, t)dt + \sigma(t)dW_t \tag{3.43}$$

They were able, by using the FPK equation and the Feynman-Kac formula, to provide a probabilistic solution of the likelihood at time T given initial condition x:

$$p(X_T, T) = \mathbb{E}\left[p_0(Y_T)\exp\left(\int_0^T -\nabla\mu(Y_s, T-s)ds\right)|Y_0 = x\right] \tag{3.44}$$

Where Y_s is a process solving:

$$dY_s = -\mu(Y_s, T-s)ds + \sigma(T-s)d\tilde{W}_s \tag{3.45}$$

However, computing the likelihood in this way is intractable as it would require computing the expectation over all possible paths of Y_s. For this reason, the authors employed a change of measure to simplify the computation using Jensen's inequality:

$$\log p(x, T) \geq \mathbb{E}_{\mathbb{Q}}\left[\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}} + \log p_0(Y_T) - \int_0^T \nabla \cdot \mu \, ds \;\middle|\; Y_0 = x\right] =: \mathcal{E}^\infty \tag{3.46}$$

Where the quantity $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ is a Radon-Nikodym derivate and it can be easily computed using Girsanov's theorem:

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(\omega) \coloneqq \exp\left(\int_0^T a(\omega, s) \cdot dB'_s - \frac{1}{2}\int_0^T \|a(\omega, s)\|_2^2 \, ds\right) \tag{3.47}$$

Where $a(\omega, s)$ is a function such that the process $d\hat{B}_s$, described by the following SDE, is a Brownian motion under measure $\mathbb{Q}$:

$$d\hat{B}_s = dB_s - a(\omega, s)ds \tag{3.48}$$

Taking into account that the expectation over all the paths of the Brownian motion for $\int_0^T a(\omega, s)dB'_s$ is zero, we can rewrite the ELBO as:

$$\mathcal{E}^\infty \coloneqq \mathbb{E}_{\hat{B}_s}\left[-\frac{1}{2}\int_0^T \|a(\omega, s)\|_2^2 \, ds + \log p_0\left(Y_T\right) - \int_0^T \nabla \cdot \mu ds \;\middle|\; Y_0 = x\right] \tag{3.49}$$

Where Y_s is a process solving the following SDE:

$$dY_s = (-\mu + \sigma a)ds + \sigma d\hat{B}_s \tag{3.50}$$

Minimising this lower bound allows to learn the parameters of both the forward and reverse SDEs through numerical solvers. We can use this fact in the context of score-based generative modelling. Take the generative and the reparametrised diffusion equation:

$$dX_s = f(X_s, s)ds + g(s)d\tilde{W}_s \tag{3.51}$$
$$dY_t = (g(t)g^T(t)\mathbf{s}_\theta(X_t, t) - f(X_t, t))dt + g(t)dW_t \tag{3.52}$$

by setting $a \coloneqq g^T\mathbf{s}_\theta$ and rewriting 3.49 as:

$$\begin{aligned}\mathcal{E}^\infty \coloneqq \mathbb{E}_{\hat{B}_s}\Bigg[& \log p_0\left(Y_T\right) \\ & - \frac{1}{2}\int_0^T \left\| g^T(s)\mathbf{s}_\theta(Y_s, s)\right\|_2^2 ds \\ & - \int_0^T \nabla \cdot (g(s)g^T(s)\mathbf{s}_\theta(Y_s, s) - f(X_s, s))ds \;\Big|\; Y_0 = x\Bigg]\end{aligned} \tag{3.53}$$

we can use the ELBO to train the score network $\mathbf{s}_\theta$ and the generative process characterised by f and g. The key connection with score-based generative modelling emerges as the score matching objective maximises $\mathcal{E}^\infty$.

3.5 Continuous-time functional diffusion processes

A continuous-time functional diffusion process over the time interval $[0, T]$ is defined by the following $\mathcal{H}$-valued, infinite-dimensional SDE, where $\mathcal{H}$ is a Hilbert space:

$$\begin{cases} dX_t = (\mathcal{A}X_t + f(X_t, t))dt + dW_t \\ X_0 \sim \rho_0 \end{cases} \tag{3.54}$$

Where $\mathcal{A}$ is an infinitesimal generator of the strongly continuous semigroup of operators At, $f : \mathcal{H} \times [0, T] \to \mathcal{H}$ is a function, and W_t is a $\mathcal{H}$-valued Wiener process characterised by the covariance operator R. The system is characterised by a path measure $\mathbb{Q}$ and a time varying measure ρ_t.
If the covariance operator R is trace-class ($\mathrm{Tr}(R) < +\infty$) or cylindrical ($R = I$), then the process is described by the following reverse-time system:

$$\begin{cases} d\hat{X}_t = (-\mathcal{A}\hat{X}_t - f(\hat{X}_t, T - t) + RD_x \log \rho_{T-t}(\hat{X}_t))dt + d\hat{W}_t \\ \hat{X}_T \sim \rho_T \end{cases} \tag{3.55}$$

The reverse process is characterised by a reverse path measure $\hat{\mathbb{Q}}$. Note that we cannot define a density on an infinite-dimensional space through a Lebesgue measure. For this reason, we use marginal densities $\rho_t^{(d)}(x^i|x^{i\neq j})$, where $d\rho_t(x^i|x^{i\neq j}) = \rho_t^{(d)}(x^i|x^{i\neq j})dx^i$ and dx^i is a Lebesgue measure. In this way, by denoting $\{e^k\}$ the basis of $\mathcal{H}$, we can characterise the process as an infinite system of SDEs:

$$\begin{cases} d\hat{X}_t^k = (\langle -\mathcal{A}\hat{X}_t - f(\hat{X}_t, T-t), e^k\rangle + r^k \frac{\delta}{\delta x^k} \log \rho_{T-t}^{(d)}(\hat{X}))dt + d\hat{W}_t^k \\ \hat{X}_T \sim \rho_T \end{cases} \tag{3.56}$$

Where r^k is the projection of the covariance operator over e^k. As by Franzese et al. [2024] and Millet et al. [1989], we require the following assumptions for the densities:

Assumption 1 *The assumption is divided in two parts:*

- *Assume that the condition law x^i given x^j for $j \neq i$ has density $\rho_t^{(d)}(x^i|x^{i\neq j})$ w.r.t. the Lebesgue's measure on $\mathbb{R}$.*
- *Assume that $\int_{t_0}^1 \int_{D_J} |r^i \frac{d}{dx^i}(\rho_t^{(d)}(x^i|x^{i\neq j}))|dx^i \rho_t(dx^{j\neq i})dt < \infty$, for fixed subset $J \subset N, t_0 > 0$ and $D_J = \left\{(\prod_{j\in J} K_j) \times (\prod_{j\notin J\mathbb{R}}),\ K_j \text{ compact in } \mathbb{R}\right\} \cap L^2(\mathbb{R})$.*

Where $L^2(R) = \{x \in \mathcal{H} : \sum r^i (x^i)^2 < \infty\}$. The second point of the assumption is technically involved and it informally requires all the possible paths of the process to be well-behaved.

In addition, if the covariance operator is trace-class and the drift term is linear and bounded $b^k \in (-K,0)$, then the reverse diffusion process behaves as in 3.56.
In principle, we can use the system above to generate samples starting from noise. However, like in a standard diffusion-based model, we do not know the value of $D_x \log \rho_{T-t}^{(d)}(x)$. For this reason, we need a parametrised score function to approximate it. We define that score function as $\mathbf{s}_\theta(x,t) : [0,T] \times \mathcal{H} \times \mathbb{R}^m \to \mathcal{H}$.
When we try to approximate the generative dynamics with the score function, we obtain a different reverse system of SDEs:

$$\begin{cases} d\hat{X}_t = (-\mathcal{A}\hat{X}_t - f(\hat{X}_t, T-t) + R\mathbf{s}_\theta(\hat{X}_t, T-t))dt + d\hat{W}_t \\ \hat{X}_T \sim \chi_T \end{cases} \tag{3.57}$$

Notice that this process is characterised by a different path measure $\hat{\mathbb{P}}$, moreover the starting distribution is also different, as we can say $\chi_T \sim \rho_T$ only in the limit $T \to +\infty$.
In Franzese et al. [2024], the dynamics are simplified by setting $f(x,t) = 0$. Moreover, we simplify the drift term with the operator $bX_t = \mathcal{A}X_t + f(X_t, t)$. Additionally, the drift term is set to be linear, so that $\langle bX_t, e^k\rangle = b^k X_t^k$. In this way we can describe the reverse process with the following system of SDEs:

$$\begin{cases} d\hat{X}_t^k = (b^k \hat{X}_t^k + r^k \frac{\delta}{\delta x^k} \log \rho_{T-t}^{(d)}(\hat{X}))dt + d\hat{W}_t^k \\ \hat{X}_T \sim \rho_T \end{cases} \tag{3.58}$$

For the training objective, we can define the quantity:

$$\gamma_\theta(X_t, t) = R(\mathbf{s}_\theta(\hat{X}_t, t) - D_x \log \rho^{(d)}_{T-t}(\hat{X}_t)) \tag{3.59}$$

and we can minimise the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbb{Q}}\left[\int_0^T \|\gamma_\theta(\hat{X}_t, t)\|^2 dt\right] \tag{3.60}$$

In particular, this optimisation problem is equivalent to optimising the ELBO Franzese et al. [2024]. Unfortunately, the true score is not available. However, we can make use of the conditional score to create an equivalent optimisation problem:

$$\mathbb{E}_{\mathbb{Q}}\left[\int_0^T \|\gamma_\theta(\hat{X}_t, t)\|^2 dt\right] = \mathbb{E}_{\mathbb{Q}}\left[\int_0^T \|\tilde{\gamma}_\theta(\hat{X}_t, \hat{X}_0, t)\|^2 dt\right] + I \tag{3.61}$$

Where I is a quantity independent of θ and

$$\tilde{\gamma}_\theta(x, x_0, t) = R(\mathbf{s}_\theta(x, T-t) - D_x \log \rho^{(d)}_{T-t}(x|x_0)) \tag{3.62}$$

In the context of the simplified system, it is possible to compute the true score analytically:

$$D_x \log \rho^{(d)}_{T-t}(x) = -S(t)^{-1}(x - \exp(\mathcal{A}t)\mathbb{E}[X_0|X_t = x]) \tag{3.63}$$

$$D_x \log \rho^{(d)}_{T-t}(x|x_0) = -S(t)^{-1}(x - \exp(\mathcal{A}t)x_0) \tag{3.64}$$

Where $S(t)$ is a function of time and the drift and diffusion terms. This allows us to compute the conditional score analytically. The main effort of the work is to find a good denoiser for computing x_0.

3.5.1 Practical implementation

In practical implementation we cannot manage quantities in a Hilbert space $\mathcal{H}$. For this reason, we need to approximate the Hilbert space with a finite-dimensional space. Our data points $\{\tilde{x}^i\}_i$, which can be images or audio tracks, for example, are samples in $\mathbb{R}^m$. We treat those points are samples taken from their corresponding samples $\{x^i\}_i$ in $\mathcal{H}$. In this case, we define an N-sized grid $\{p_j\}_j$, such that $\tilde{x}^i_j = x^i[p_j]\ \forall i, j$. We can reconstruct x^i by interpolating $\tilde{x}^i$. Under some assumptions, we can have exact reconstruction Franzese et al. [2024]. When exact reconstruction is not possible, we can still compute the reconstruction error.

Interpolating data in a Hilbert space We can reconstruct data points in a Hilbert space given a finite collection of samples in a grid by using a set of interpolation functions $\{\xi_i\}_i$. Franzese et al. [2024] employ a set of functions obtained from the Fourier basis

$\{e^k\}_k := \{\exp(j2\pi kp) | k \in \mathbb{Z}\}$ and p is a coordinate of the grid. In this way, we can define the interpolating functions as follows:

$$\xi_i = \sum_{k=1}^{N} e^k \exp\left(-j2\pi k \frac{i}{N}\right) = \sum_{k=1}^{N} \exp\left(j2\pi k \left(p - \frac{i}{N}\right)\right) \tag{3.65}$$

Notice that the interpolating functions are a linear combination of the Fourier basis. We can then recover $x \in H$ from the corresponding $\tilde{x}$ by the following general procedure:

$$x = \sum_{i=1}^{N} \xi_i \tilde{x}_i \tag{3.66}$$

In our particular case, we can write:

$$x = \sum_{i=1}^{N} \xi_i \tilde{x}_i = \sum_{i=1}^{N} \sum_{k=1}^{N} \exp\left(j2\pi k \left(p - \frac{i}{N}\right)\right) \tilde{x}_i \tag{3.67}$$

Projecting data from a Hilbert space For practical implementation, we also need to define a projection operator mapping $\mathcal{H}$ to $\mathbb{R}^L$. Franzese et al. [2024] define a projection scheme through a set of $\{\zeta_i\}_i$ such that $\langle x, \zeta_i \rangle = x[p_i]$. This, together allows us to describe the reverse process as a finite system of SDEs of the same size as the grid:

$$\begin{cases} d\hat{X}_t[p_k] = (-\langle A \sum_i \hat{X}_t[p_i]\xi_i, \zeta_k \rangle + R\langle \mathbf{s}_\theta(\sum_i \hat{X}_t[p_i]\xi_i, T-t), \zeta_k \rangle)dt + d\hat{W}_t[p_k] \\ \hat{X}_T \sim \chi_T \end{cases} \tag{3.68}$$

In our particular case, Franzese et al. [2024] has shown that the finite system of SDEs can be written in terms of the Fourier transform $\mathcal{F}(z^i) = \sum_{i=1}^{N} z^i \exp(j2\pi \frac{i}{N})$ and its inverse $\mathcal{F}^{-1}(z^i) = \frac{1}{N} \sum_{i=1}^{N} z^i \exp(-j2\pi \frac{i}{N})$:

$$\begin{cases} d\hat{X}_t[p_k] = (-\mathcal{F}^{-1}(b^l \mathcal{F}(\hat{X}_t[p_i])^l)^k + \mathcal{F}^{-1}(r^l \mathbf{s}_\theta(\sum_i \hat{X}_t[p_i]\xi_i, T-t)))dt + d\hat{W}_t[p_k] \\ \hat{X}_T \sim \chi_T \end{cases} \tag{3.69}$$

Where b_l is the projection of the drift term over the basis ζ_l and r_l is the projection of the covariance operator over the basis ζ_l.

Finite representation of Hilbert space functions When computing the score for some $x \in \mathcal{H}$ and $t \in [0, T]$, we need a finite representation for our function, since we cannot manage infinite-dimensional vectors. More formally, we need a mapping $g : [0, T] \times \mathcal{H} \times \theta \to \mathbb{R}^m$. Franzese et al. [2024] used Implicit Neural Representations (INRs) and transformers to infer finite representations of the score function.

3.6 Implicit Neural Representations (INRs)

Any data point $x \in \mathcal{H}$, where $\mathcal{H}$ is a generic Hilbert space over the field of real numbers, can be represented as a function $f : I \to \mathbb{R}$, where I is some index space which we

use to index the components of any $x \in \mathcal{H}$. This type of data representation is called **Implicit Neural Representation** (INR). INRs offer many advantages over traditional neural networks, such as the ability to represent complex functions with fewer parameters and the ability to represent functions in a continuous space. In practical implementations, they found success in modelling 3D scenes; Park et al. [2019] employed INRs to model 3D shapes. In particular, they used a neural network to represent the signed distance function of a 3D shape. That is, a function $f_x : \mathbb{R}^3 \to \mathbb{R}$ associated to a scene x which, given any 3D point in x it returns the distance of that point from the surface of the targeted object.

3.6.1 Architectures for INRs

The authors of Park et al. [2019] used a neural network to represent the signed distance function of a 3D shape. They employed a feed-forward neural network with a series of fully connected layers with rectified linear units (ReLu) Agarap [2018] as activation functions. However, this kind of architecture struggles in modelling signals with high level of details, in particular they fail to accurately model the temporal and spatial derivates of the signals. For this reason, the authors of Sitzmann et al. [2020] proposed to introduce periodic activation functions in the neural network. They showed that by using periodic activation functions, the network can model signals with high level of details. In particular, they used the sine activation function $\sin(\omega x)$, where ω is a hyperparameter. The authors called this kind of networks Sinusoidal Neural Networks (SIREN).
A SIREN Φ is trained by minimizing a set of m constraints on data points and their derivatives $\{\mathcal{C}_m(a(x), \Phi(x), \nabla\Phi(x), ...)\}$. In particular, they minimise the following loss function:

$$\mathcal{L} = \int_\Omega \sum_{m=1}^{M} \mathbf{1}_{\Omega_m} ||\mathcal{C}_m(a(x), \Phi(x), \nabla\Phi(x), ...)|| dx \tag{3.70}$$

In practice, this loss function is enforced by sampling a set of coordinates of the desired data point x, $\{x_i\}_i$ and computing the loss function on those points. The authors showed that SIRENs can model complex functions with fewer parameters than traditional neural networks.
Regrettably, this approach forces the user to infer a set of parameter for each of the data points, increasing the need for compute and overall training time.

MLP modulations

In their work Dupont et al. [2022], the authors proposed an alternative methodology for extracting INRs using SIRENs. They proposed to use the SIREN to extract the common structure of many similar data points and then use modulations to extract the specific structure of each data point. In particular, they have shown that it is possible to extract modulations for each data point with a meta-learning approach involving just three stochastic gradient descent (SGD) Amari [1993] steps. More formally, modulations ψ are a set of additional parameters, with respect to the base network Φ with to base parameters θ and, while θ are shared among all the data points, ψ are specific to each data point. In general, modulations can be used to modify the existing network arbitrarily

using a user-specified methodology $f(\Phi, \theta, \psi)$. In practice, we only consider the following approaches:

- **Parameter modulations** Franzese et al. [2024]: the modulations are used to modify the parameters of the network. This approach is very versatile and powerful, but it scales with the number of parameters of the network.
- **Shift and scale modulations** Dupont et al. [2022]: the modulations are used to shift and scale the output of the base network at every layer. In this case, the modulations are defined as $\psi = (\gamma, \beta)$, where γ is a scaling factor and β is a shift factor. In the case of an MLP, the number of modulations scale as the number of layers of the network.
- **Latent modulations** Dupont et al. [2022]: this approach is similar to shift and scale modulations, except that it uses a shallow network to encode the modulations. In this way, it is possible to preserve the dimensionality of the modulations, independently of the size of the network.

3.7 Reinforcement learning

Reinforcement learning (RL) is a paradigm of machine learning; informally, it focuses on maximizing the goodness of the final outcome of a sequence of decisions. More formally, this paradigm assumes the following elements:

- **Agent**: the entity that learns the policy.
- **Environment**: the world in which the agent operates. It can assume different states.
- **Action**: the set of actions that the agent can perform on the environment.
- **Reward**: the feedback that the agent receives from the environment.

The agent performs actions on the environment, which can change state accordingly. In return, the environment provides a reward to the agent. The agent perceives the environment through **observations** and constructs an internal representation of the environment S_t^A. In general internal observations of agents are not isomorphic to the environment states. We have:

- **Fully observable environment**: the agent can observe the complete state of the environment: the internal representation is isomorphic to the environment state.
- **Partially observable environment**: the agent can only observe a part of the environment: the internal representation is *not* isomorphic to the environment state.

The agent's goal is to find a policy that maximises the expected reward in the long run. The reward is a way to encode the goals of the agents. Depending on the task this may be feasible or not:

- **Episodic task**: the agent interacts with the environment for a finite number of steps. The agent's goal is to maximise the reward: $G_t = \sum_{u=t}^{T} R_{u+1}$.
- **Continuing task**: the agent interacts with the environment for an infinite number of steps. The agent's goal is to maximise the expected discounted reward: $G_t = \sum_{u=t}^{T} \gamma^u R_{u+1}$.

Using the discounted reward we can define the **value function** of a state given a policy π as:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{3.71}$$

Similarly, we can define a function that gives the value of a state-action pair:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{3.72}$$

Markov decision processes (MDP) MDPs are an environment for RL agents. They are *fully observable* and rely on the **Markov property**: the next state of the environment only depend on the immediately previous one. When an MDP receives an action a in state s, it returns a reward r and a new state s'. The probability of transitioning from state s to state s' given action a is given by $P(s'|s, a)$. We can exploit this property to compute the expected reward:

$$p(s', r|s, a) = P(s'|s, a)R(r|s, a, s') \implies \mathbb{E}[r|s, a] = \sum_r r \sum_{s'} p(s', r|s, a) \tag{3.73}$$

If we include the policy of the agent, we can use it to compute the expected reward (**Bellman's equation**):

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s')] \tag{3.74}$$

We say that a policy is **optimal** with respect to another policy if its expected reward is larger in all the possible states.

3.7.1 Q-learning

Model free approaches to reinforcement learning are possible, they do not use a model of the environment and proceeds by trial and error. Q-learning is a model free approach to reinforcement learning. It is based on the concept of **Q-value**, which is the expected reward of a state-action pair. The Q-value is updated as follows:

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a')) \tag{3.75}$$

Where α is the learning rate and γ is the discount factor. The Q-value is updated using the **temporal difference** between the current Q-value and the expected Q-value. The Q-value is used to select the action to perform in the environment. In this case, the Q function is a **Q-table**, where we store values for each state-action pair. In this way it can

be difficult to map all the states, this is why it is crucial to represent the environment well, playing with symmetries. Modern approaches use deep learning methods to estimate the Q function, this is called **deep Q-learning**. The Q function is approximated using a neural network, which is trained using the temporal difference error.

3.7.2 Reinforcement learning for diffusion models

Sampling in the context of diffusion models can accept a formulation as an MDP, intuitively the reverse diffusion process entails a sequence of steps, and we are interested only in the final outcome of the sampling process. The authors of Black et al. [2023] modified Denoising Diffusion Probabilistic Models (DDPMs) Ho et al. [2020] proposing two techniques: **Reward-Weighted Regression**(RWR) and **Denoising Diffusion Policy Optimisation** (DDPO). RWR models the sampling task as a single step MDP, where, given a reward signal r, a dataset of samples x_0 and context c with conditional distribution $p(x_0|c)$, the DDPM loss is weighted by a factor w_{RWR}:

$$w_{RWR}(x_0|c) = \frac{1}{Z}\exp(\beta r(x_0, c)) \tag{3.76}$$

Where Z is a normalizing constant and β is the inverse temperature. However, the authors admit that this approach does not translate well to a RL problem as the loss in DDPM does not involve the exact computation of the log-likelihood but just a variational bound.

Denoising Diffusion Policy Optimisation (DDPO) is an approach that interprets the sampling process as a multistep MDP, with the following mapping:

- **State**: the triple $s_t = (c, x_t, t)$.
- **Action**: the next denoised data point $a_t = x_{t-1}$.
- **Policy**: the conditional distribution $p_\theta(x_t|x_{t-1}, c)$.
- **Reward**: a reward signal valid only for the final step $R(s_t, a_t) = 0$ if $t > 0$ else $r(x_0, c)$.

This formulation, allows the exact computations of the gradients of $\mathbb{E}_{c\sim p(c), x_0\sim p_\theta(x_0|c)}[r(x_0, c)]$. In turn, this allows the authors to use the REINFORCE algorithm Mohamed et al. [2020] to optimise the policy.

3.8 Transformers

In this section, we will cover the theory behind transformers and their application in diffusion-based generative modelling.

3.8.1 Attention

The transformer architecture firstly encountered wide success and application in the Natural Language Processing domain. This architecture builds upon the attention mechanism; in NLP attention weighs the importance of each word with respect to the other words. More formally, given the key matrix K and the query matrix Q, the attention is the matrix of weights:

$$w = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{3.77}$$

Where d_K is the dimension of the matrix K. Then, we can construct an attention layer: given a value matrix V in addition to the matrices K and Q, the attention is:

$$\text{Att}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3.78}$$

Attention resembles the concept of retrieval in databases, this is why the Q,K and V matrices have their name. The actual meaning of these matrices depends on the user, when $Q = K = V$ we talk about **self-attention**, whereas if they differ we talk about **cross-attention**. Modern transformer architectures employ multi-head attention, which is composed of multiple attention heads. In turns, each attention head projects its inputs with a learnable projection layer to form the Q, K and V matrices. Afterwards the attention of the head is computed and the output of each head is concatenated or passed through a linear layer to form the output of the multi-head attention.

3.8.2 The original transformer architecture

The original transformer architecture was designed for NLP tasks. It employs an encoder-decoder configuration where each block of the decoder attends to the output of its corresponding block. The encoder block is composed by multi-head attention followed by a linear layer, whereas the decoder uses three subcomponents:

- A masked multi-head attention, where tokens in the sequence can only attend to past tokens.
- A multi-head attention which attends to the output of the decoder block.
- A linear layer to project the output on an embedding space with a different dimension.

The attention mechanism does not retain spatial information, for this reason the input of the transformer is summed with the **positional encoding**. In the work by Vaswani et al. [2017], the authors employed sine and cosine functions to embed spatial information.

3.8.3 Vision transformer

Following the NLP revolution with transformer-based models like BERT and GPT, a similar approach was taken by computer vision scientists to create more powerful deep

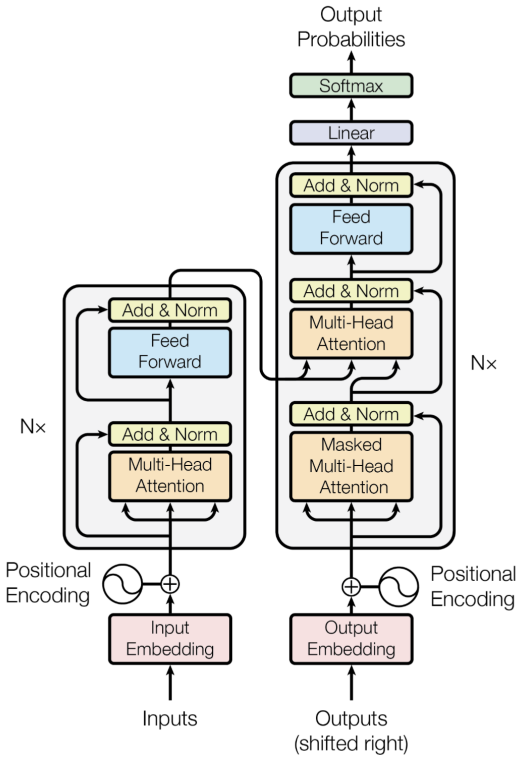

Figure 3.4: The architecture of the transformer Vaswani et al. [2017]

learning architectures. The authors of Dosovitskiy et al. [2020] proposed an architecture similar to BERT, with the differences being the creation of the input tokens. In this case the input tokens are flattened overlapping 2D patches. The authors retained the same positional embeddings as in the original transformer architecture. The key difference with CNNs is that they do not make large use of the inductive bias given by the image structure, instead that bias must be recovered with larger training times. For this reason, ViT has been put to test to verify its capabilities, and it was shown that it had a higher advantage with larger datasets. Additionally, like BERT, ViT takes great advantage with self-supervised pretraining. In particular, it employs a masked patch prediction task in order to produce good-quality general purpose image representations.

3.8.4 Vision transformer for diffusion models

The original ViT architecture is not specified for diffusion models or, more in general, pixel-level prediction tasks, such as semantic segmentation. For this reason, the authors of Bao et al. [2023], inspired by the U-net design Ronneberger et al. [2015], devised U-ViT. U-ViT parametrises a noise prediction function $\epsilon_\theta(x_t, t, c)$, where:

- x_t is the noise image.
- t is the time.

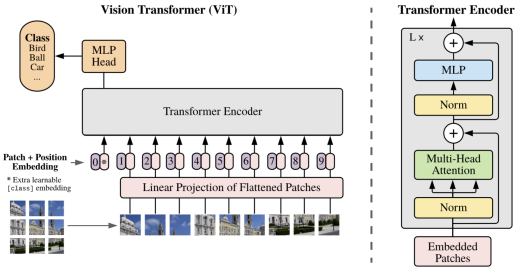

Figure 3.5: The architecture of ViT Dosovitskiy et al. [2020]

- c is the condition for adapting to conditional generation.

The architecture is similar to ViT, including using patches as input tokens, while the main differences lie in the embedding of time and condition. Time can be embedded either as a token in input or as adaptive layer normalization (AdaLn), which takes the embedding of the transformer block h and the projection of the time embedding into a linear layer to modulate the standard layer norm:

$$\text{AdaLn}(h, y) = y_s \text{Ln}(h) + y_b \tag{3.79}$$

The condition, instead, is treated as an input token. Additional changes with respect to ViT include long range connections, like in U-net, and the addition of a 3×3 convolutional block after the last transformer layer. This block has shown to be effective in reducing the number of artefacts in the generated image.

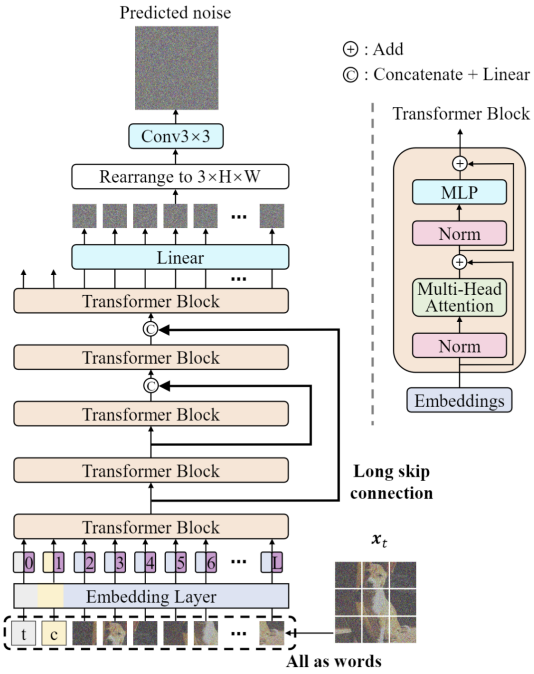

Figure 3.6: The architecture of U-ViT Bao et al. [2023]

Chapter 4

Proposed method

In this chapter, we highlight the methodologies for:

1. Computing Implicit Neural Representations (INRs) without meta-learning.
2. Reparametrise the Stochastic Differential Equations (SDEs) of the score-based generative model with fewer hyperparameters.

4.1 Modulation networks for INRs

In the previous chapter, we saw how a data point can be modelled as a function. Additionally, we explored methodologies to learn INRs efficiently, without the need to relearn from scratch the network for each data point. Recalling some elements from the previous chapter, we can define a data point $x \in \mathcal{H}$ as a function $x : I \to \mathbb{R}$, where I is some index space which we use to index the components of any $x \in \mathcal{H}$. Additionally, we define a base network $f(x[p_i], t; \theta, \phi)$, where:

- θ are the base parameters of the network.
- ϕ are the modulations of the network.
- t is the time.
- x is the input data point, that is, for example, an image or a soundtrack.
- p_i is the i-th coordinate of the input data point, that is, for example, the position of a pixel.

In their work Franzese et al. [2024], the authors used a meta-learning approach to learn the modulations of the denoising network, using the following optimisation objective:

$$\arg\min_{\theta} \sum_{x} \sum_{p} \|f(p, t; \theta, g(\theta, x)) - \tilde{x}[p]\|_2^2 \tag{4.1}$$

Where $\tilde{x}$ is the original, not noisy, sample and g is a meta-learning procedure:

$$g(\theta, x) = \arg\min_{\phi} \sum_{p} \|f(p, t; \theta, \phi) - x[p]\|_2^2 \tag{4.2}$$

This procedure is needed so that we can make f a functional representation of our input data point and specialise the network to any data point at choice. In this work, we use other networks to compute modulations to avoid performing meta-learning. In particular, we use a modulation network g, like a ViT, for inferring the modulations ϕ of the base network, in our case an MLP. It follows that $g(x; \psi)$ is simply another network with its set of parameters ψ. For this reason, we can rewrite the training objective as follows:

$$\arg\min_{\theta,\psi} \sum_{x} \sum_{p} \|f(p, t; \theta, g(x; \psi)) - \tilde{x}[p]\|_2^2 \tag{4.3}$$

It must be remarked that the modulation network treats the input sample as a whole, and it does not manipulate coordinates directly. This allows to use the inductive bias of modern deep learning architectures, whereas the meta-learning approach must learn the specific structure of the data point from scratch.

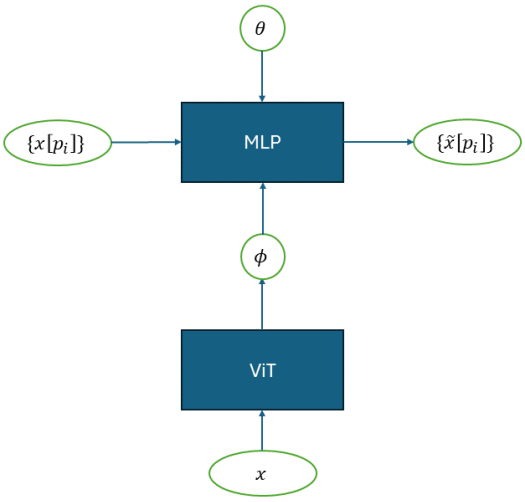

Figure 4.1: The mechanism of the modulation network for skipping meta-learning

In the practical implementation, we used an MLP as a base network and a ViT as a modulation network. In order to produce the right number of parameters, the ViT was modified to include two linear layers on top of the latent representations. The two layers were used to implement the latent modulation approach as in Dupont et al. [2022]: one layer produced the latent modulations, while the second layer decoded the latent vector into the modulations of the MLP.

4.2 Reparametrisation of the score-based generative model

Functional diffusion processes suffer from higher instabilities than standard score based diffusion models due to having to deal with infinite dimensional data. In particular, we have additional parameters such as the b^k terms, as in the simplified system of SDEs for the forward process:

$$\begin{cases} dX_t^k = b^k X_t^k dt + dW_t^k \\ X_0 \sim \rho_0 \end{cases} \tag{4.4}$$

Additionally, we have the terms due to the projection of the covariance operator r^k, as we can see from the simplified reverse process:

$$\begin{cases} d\hat{X}_t^k = (-b^k \hat{X}_t^k - r^k \frac{\delta}{\delta x^k} \log \rho_{T-t}^{(d)}(\hat{X}))dt + d\hat{W}_t^k \\ \hat{X}_T \sim \rho_T \end{cases} \tag{4.5}$$

These parameters are not learned and must be inferred from the data. This leads to a larger expense of time and compute resources. Additionally, (b^k, r^k) are infinite sets of parameters, as they correspond to the basis of the Hilbert space $\mathcal{H}$ of the diffusion process. In the practical implementation, however, we employ a finite set of functions derived from the Fourier basis $\{e^k\}_k$. In this case, we have N (b^k, r^k) parameters, where N is the number of points that we have sampled for our function $x \in \mathcal{H}$, or, in a more practical scenario, the number of pixels in an image. Importantly, this also implies that the number of hyperparameters scale with the dimensionality of the data, which makes it more difficult to adapt functional diffusion processes to higher dimensional data.
For this reason, in this section we propose different methodologies to simplify functional diffusion processes as follows:

1. We reformulate the SDE by removing the drift terms b^k.

2. We estimate the covariance operator R directly from the data.

4.2.1 The driftless diffusion process

In this section, we propose to remove the drift parameters b^k. In this way, we can obtain a simpler formulation for continuous-time functional diffusion processes that are also easier to train. We propose two different derivations of the same result, one more grounded to theory and the other more practical. The most practical derivation is based on Taylor expansions which allow us to approximate the SDE under the assumption of a very small drift term. The more theoretical derivation is based on the concept of infinitesimal generators and semigroups, which allow us to rewrite the SDE without approximations.

Removing the drift term with Taylor approximation We start by setting $b^k \sim O((dt)^2)$. In this way, we can approximate the diffusion process as follows:

$$\begin{cases} dX_t^k = dW_t^k + o(dt) \\ X_0 \sim \rho_0 \end{cases} \tag{4.6}$$

Moreover, if we require $r^k \sim O(dt)$, we also have that $b^k \sim o(r^k)$. This can help us in approximating factors for the reverse process:

$$\begin{cases} d\hat{X}_t^k = -r^k \frac{\delta}{\delta x^k} \log \rho_{T-t}^{(d)}(\hat{X})dt + d\hat{W}_t^k + o(dt) \\ \hat{X}_T \sim \rho_T \end{cases} \tag{4.7}$$

Additionally, we can also simplify the analytical expression of the score, as by Franzese et al. [2024], in the case of the Fourier transform $\mathcal{F}$ and its inverse $\mathcal{J}$ we have:

$$\frac{d}{dx^i} \log \rho_t^{(d)}(x^i|x^{i\neq j}) = -(s^i)^{-1} \left(x^i - \int_{x_0^i} \exp(tb^i) x_0^i \rho_t^{(d)}(x_0^i|x) dx_0^i \right) \tag{4.8}$$

Where $s^i = r^i \frac{\exp(2b^i t)-1}{2b^i}$. By using Taylor's expansion, one can easily prove that we get the following approximation:

$$\frac{d}{dx^i} \log \rho_t^{(d)}(x^i|x^{i\neq j}) \simeq -\frac{1}{tr^i} \left(x^i - \int_{x_0^i} x_0^i \rho_t^{(d)}(x_0^i|x) dx_0^i \right) \tag{4.9}$$

In this way, the SDE is completely characterised by the covariance operator R, similarly to the Variance Exploding SDE Song et al. [2020].

Reformulating the SDE We now proceed with an approach more grounded to theory, given the infinite dimensional SDE:

$$\begin{cases} dX_t = \mathcal{A}X_t + dW_t \\ X_0 \sim \rho_0 \end{cases} \tag{4.10}$$

We design $\mathcal{A}$ to be the infinitesimal generator of the following semigroup $S(t)$:

$$S(t)x = x \tag{4.11}$$

That is $\mathcal{A}$ always generates the identity operator. We can verify $\mathcal{A}$ is indeed an infinitesimal generator as the following limit holds for any $x \in \mathcal{H}$:

$$\mathcal{A}x = \lim_{t\to 0} \frac{S(t)x - x}{t} = \lim_{t\to 0} \frac{x - x}{t} = 0 \tag{4.12}$$

Hence, we can rewrite the SDE as follows:

$$\begin{cases} dX_t = dW_t \\ X_0 \sim \rho_0 \end{cases} \tag{4.13}$$

Moreover, by Da Prato and Zabczyk [2014], we can write a weak solution of the previous system as:

$$X_t = X_0 + \int_0^t dW_s \tag{4.14}$$

Additionally, as per Franzese et al. [2024], we have a closed form for the score function:

$$D_x \log \rho_t^{(d)}(x) = -\mathcal{S}(t)^{-1}(x - \exp(t\mathcal{A})\mathbb{E}[X_0|X_t = x]) \tag{4.15}$$

Where:

$$\mathcal{S}(t) = \int_{s=0}^t \exp((t-s)\mathcal{A})R\exp((t-s)\mathcal{A}^\dagger)^T ds \tag{4.16}$$

Which in our case simplifies as:

$$\tilde{\mathcal{S}}(t) = \int_{s=0}^t Rds = tR \tag{4.17}$$

This allows us to simplify 4.15 as:

$$D_x \log \rho_t^{(d)}(x) = -\tilde{\mathcal{S}}(t)^{-1}(x - \mathbb{E}[X_0|X_t = x]) \tag{4.18}$$

Then, we must verify if the SDE is time reversible. In the case $R = I$, one can easily verify that assumption I from Franzese et al. [2024] is feasible:

$$\int_0^T \left(b^i(X_t,t)\right)^2 dt + \sum_{j\neq i} \mathbb{E}\left[\int_0^T \left(b^j(X_t,t) - \mathbb{E}\left[b^j(X_t,t) \mid \mathcal{F}_t^{(i)}\right]\right)^2 dt\right] < \infty, \quad Q^{(i)} \text{ a.s.} \tag{4.19}$$

We can notice that it is always valid if $b^i(X_t, t) = 0$. In the case of trace-class operators, we must verify the feasibility of another set of assumptions, namely assumptions II, III, IV and VI from Franzese et al. [2024]. We start from assumption II, where we must have a constant K such that, for every pair of data points (x, y):

$$\forall x \in L^2(R), \sup_t \left\{\sum_i r^i \left(b^i(x,t)\right)^2\right\} + \sum_i (r^i)^2 \leq K\left(1 + \sum_i r^i (x^i)^2\right) \tag{4.20}$$

$$\forall x, y \in L^2(R), \sup_t \left\{\sum_i r^i \left(b^i(x,t) - b^i(y,t)\right)^2\right\} \leq K \sum_i r^i (x^i - y^i)^2 \tag{4.21}$$

We can see that in our case, having $b^i(x, t) = 0$, we observe:

$$\sum_i (r^i)^2 \leq \sup_t \left\{\sum_i r^i \left(b^i(x,t)\right)^2\right\} + \sum_i (r^i)^2 \leq K\left(1 + \sum_i r^i (x^i)^2\right) \tag{4.22}$$

$$0 \leq \sup_t \left\{\sum_i r^i \left(b^i(x,t) - b^i(y,t)\right)^2\right\} \leq K \sum_i r^i (x^i - y^i)^2 \tag{4.23}$$

Which means that this assumption can be satisfied only counting on r^i.
Assumption III, instead, corresponds to H5 from Millet et al. [1989]. By the same work we have that H5 is satisfied if the following holds:

1. $b^i(x,t)$ does not depend on t.
2. $b^i(x,t)$ only depends on x^i.
3. Assumption I is satisfied.

We can clearly see that $b^i(x,t) = 0$ satisfies all the conditions.
Assumption IV requires the following condition or another condition from Millet et al. [1989]. We stick with just the following because it is always satisfied for $b^i(x,t) = 0$:

$$\forall x, y \in L^2(R), \sup_t \left\{ \sum_i r^i \left(b^i(x,t) - b^i(y,t) \right)^2 \right\} \leq K \sum_i (r^i)^2 (x^i - y^i)^2 \tag{4.24}$$

Finally, assumption VI does not involve a restriction on $b(x,t)$. This implies that equation 4.13 is time-reversible and its time-reverse formulation reads as follows:

$$\begin{cases} d\hat{X}_t = RD_x \log \rho^{(d)}_{T-t}(\hat{X}_t)dt + d\hat{W}_t \\ \hat{X}_0 \sim \rho_T \end{cases} \tag{4.25}$$

with the same expression for the score function as the right-hand side of equation 4.9, which in functional terms reads as follows:

$$\begin{cases} d\hat{X}_t = -t^{-1}(\hat{X}_t - \mathbb{E}[X_0 | X_t = \hat{X}_t])dt + d\hat{W}_t \\ \hat{X}_0 \sim \rho_T \end{cases} \tag{4.26}$$

4.2.2 Estimating the covariance operator from data

The goal of generative modelling is to generate good quality samples, which translates in the ability to sample from the distribution ρ_0. In our case, intuition tells us it will be easier for the model to capture the distribution of the data points if we set the covariance operator R to be the empirical covariance of the data points. It follows that the model can start sampling to a distribution which is close to ρ_0 and then refine the samples with the reverse diffusion process.
In order to extract this ideal prior distribution ρ_T, we start with the simplified process as discussed in the previous section:

$$\begin{cases} dX_t = dW_t \\ X_0 \sim \rho_0 \end{cases} \tag{4.27}$$

In particular, we know that as $T \to \infty$, the distribution of X_T will converge to the distribution of the Wiener process, which is a Gaussian distribution with mean 0 and covariance operator R. Our aim is to choose the closest possible Gaussian distribution to the data points, which can be done by estimating the covariance operator R from data:

$$\hat{R} = \sum_{i}^{N} \frac{|\mathcal{F}(x_i)|^2}{N} \tag{4.28}$$

In this way, the covariance does not change space, it just increases in intensity. This allows the model to start sampling from a distribution close to the data points and then refine the samples with the reverse diffusion process. While estimating the covariance operator would require estimating infinite parameters, it is important to denote that we only need to estimate the projection of the covariance operator for some choice of functions used in our implementation to represent the data points.
In our case, we can play with the Fourier basis as in Franzese et al. [2024]. Firstly, we recall the forward system:

$$\begin{cases} dX_t[p_k] = \langle \mathcal{A} \sum_i X_t[p_i]\xi_i, \zeta_k \rangle + dW_t[p_k] \\ X_0[p_k] \sim \rho_0[p_k] \end{cases} \tag{4.29}$$

In our case this system simplifies as:

$$\begin{cases} dX_t[p_k] = dW_t[p_k] \\ X_0[p_k] \sim \rho_0[p_k] \end{cases} \tag{4.30}$$

The reverse system, instead, is more complicated, as shown by Franzese et al. [2024]:

$$\begin{cases} d\hat{X}_t[p_k] = (-\mathcal{F}^{-1}(b^l \mathcal{F}(\hat{X}_t[p_i])^l)^k + \mathcal{F}^{-1}(r^l \mathcal{F}(\mathbf{s}_\theta(\sum_i \hat{X}_t[p_i]\xi_i, T-t))^l))dt + d\hat{W}_t[p_k] \\ \hat{X}_T \sim \chi_T \end{cases} \tag{4.31}$$

Again, this simplifies in our case by the removal of the drift term:

$$\begin{cases} d\hat{X}_t[p_k] = \mathcal{F}^{-1}(r^l \mathcal{F}(\mathbf{s}_\theta(\sum_i \hat{X}_t[p_i]\xi_i, T-t))^l)dt + d\hat{W}_t[p_k] \\ \hat{X}_T \sim \chi_T \end{cases} \tag{4.32}$$

Additionally, if we explicitate the score function using the equivalent of 4.18 using a denoiser to approximate the expectation:

$$\mathbf{s}_\theta(X_t, t) = -\tilde{S}^{-1}(t)(X_t - \mathbb{E}[X_0|X_t]) \tag{4.33}$$

We can then write the reverse system as:

$$\begin{cases} d\hat{X}_t[p_k] = -\frac{1}{t}(\hat{X}_t[p_k] - f(p_k, t; \theta, g(\theta, \sum_i \hat{X}_t[p_i]\xi^i)))dt + d\hat{W}_t[p_k] \\ \hat{X}_T \sim \chi_T \end{cases} \tag{4.34}$$

As we can see, this approach simplifies the SDE and allows a more stable computation of the reverse process due to the absence of parameters in the drift term and the fact that the Fourier transform is rendered unnecessary. Importantly, we must stop the reverse diffusion at a time $t > 0$ in the simulation due to numerical instabilities as it is the case for the Variance Exploding SDE Song et al. [2020].
It follows that we can implement the sampling procedure through the following algorithm:

Algorithm 3 Sampling from driftless SDE

procedure DRIFTLESS SDE SAMPLING(X_0, f_θ, N, T)
  $\Delta T \leftarrow \frac{1}{N}T$
  for $k = N$ to 1 **do**
    $t \leftarrow \frac{k}{N}T$
    $\overline{X}_k \leftarrow X_{k-1} - \Delta T(Rt)^{-1}(X_{k-1} - f_\theta(X_{k-1}, t))$
    $X_k \leftarrow \overline{X}_k + \underbrace{(R\Delta T)^{\frac{1}{2}}\mathrm{Re}(\mathcal{F}^{-1}(\mathcal{F}(\mathcal{N}(0,1)) \cdot \exp(2\pi j\mathcal{N}(0,1))))}_{\text{Reverse noise } \Delta W}$
  end for
  return $f_\theta(X_1, \Delta T)$
end procedure

Note that if one sets $\sigma^2 = Rt$, the algorithm is equivalent to the predictor sampling algorithm of the Variance Exploding SDE Song et al. [2020]. This algorithm can be improved following the Euler-Maruyama method and adding predictor steps as in Song et al. [2020]. In the experiments, we employ the Euler-Maruyama version of the same algorithm with a corrector step.

Chapter 5

Experiments

We dedicate this chapter to the description of the experiments and their results.

5.1 Data

For this project, we employed the MNIST dataset LeCun et al. [1998] as a benchmark for our generative model. We refrained from using more complex datasets, such as CIFAR-10 Krizhevsky et al. [2009], due to the computational cost of training the model and due to the highly theoretical nature of this work, whose methodology is unexplored in the literature. The MNIST dataset is composed of 28×28 grayscale images of handwritten digits from 0 to 9. We rescaled the dataset to 32×32 pixels as done by Franzese et al. [2024].

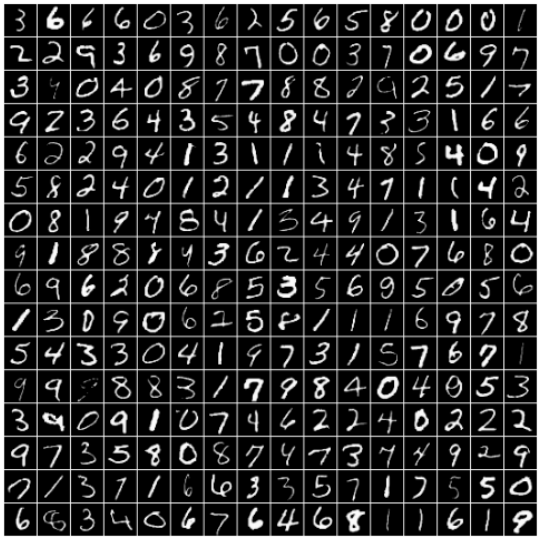

Figure 5.1: Example of images from the MNIST dataset

5.2 Software and code

The code for this work has been written in Python, using mainly the *PyTorch* library Paszke et al. [2019] and *Huggingface* libraries such as *Diffusers* Team [2024] and *Transformers* Wolf et al. [2020]. The codebase was developed by following the structure of Franzese et al. [2024]; however, the aforementioned work was developed in *Jax*, while we decided to rely on *PyTorch* and *Huggingface*. Therefore, with this work we contributed in making the codebase more accessible to the public, as *PyTorch* is a more widely used library than *Jax*. Additionally, the usage of the *transformers* library allows us to upload the model to the *Huggingface Hub*, making it easier to share the model with the community. It is also worth noting that the modularity of the network allows for more creative approaches with modulation network. For example, one can try different modulations networks, such as convolutional neural networks or multi-layer perceptrons.

5.3 Experimental settings

This work gives the opportunity to try many experimental setups. Firstly, we compare a pure, transformer-based approach with a modulation network approach. Secondly, we experiment with the new SDE using a transformer-based approach.
For our experiments we used a U-ViT network with $O(1M)$ parameters, as in Bao et al. [2023]. For what concerns the modulation network, we used a ViT with $O(1M)$ parameters, while for the base network we employed an MLP with $O(100K)$ parameters.
We first compare the performances of different architectures over the MNIST dataset using the SDE by Franzese et al. [2024], and then compare the performances of the same architectures using the driftless SDE. All the models used for evaluation have been trained for 500000 iterations, with a batch size of 64 and adaptive learning rate with a warm-up period of 1000 iterations. We used the Adam optimiser with a learning rate of 10^{-5} and a weight decay of 0.03. Additionally, we used the same data augmentation techniques as in Franzese et al. [2024], namely random horizontal flipping. The models used for evaluation also employ Exponential Moving Average (EMA) Song et al. [2020] for their weights, with an EMA rate of 0.9999.

5.3.1 Results with general SDE

We illustrate some samples generated by the modulation network approach and by the U-ViT network. The samples have been generated with a PC algorithm, using one correction step. In particular, we show the samples generated by training our models on the type of noise constructed appositely for MNIST by Franzese et al. [2024]. We can notice that the samples are of similar quality, however, the ones produced by the modulation network approach are smoother, as in fig. 5.2, with respect to the samples produced by the U-ViT in fig. 5.3.
We also compared the two models trained on white noise, in this case we got similar results, with the samples produced by the modulation network approach being smoother than the ones produced by the U-ViT network (fig. 5.5 and fig. 5.4).

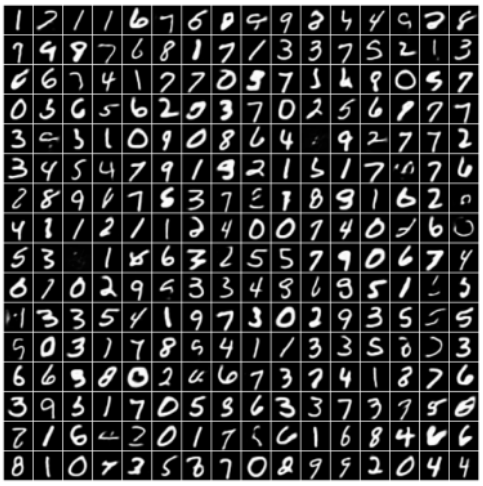

Figure 5.2: Samples generated by the modulation network approach trained on coloured noise

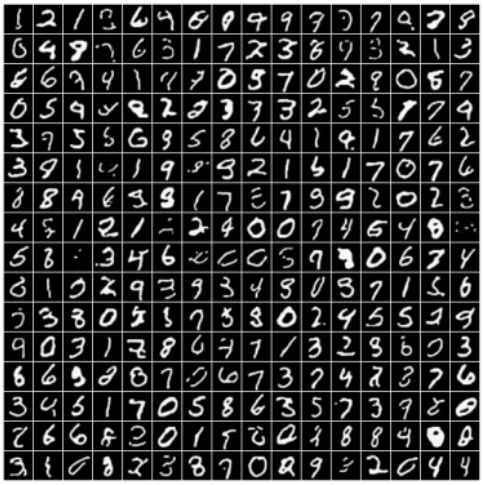

Figure 5.3: Samples generated by the U-ViT network trained on coloured noise

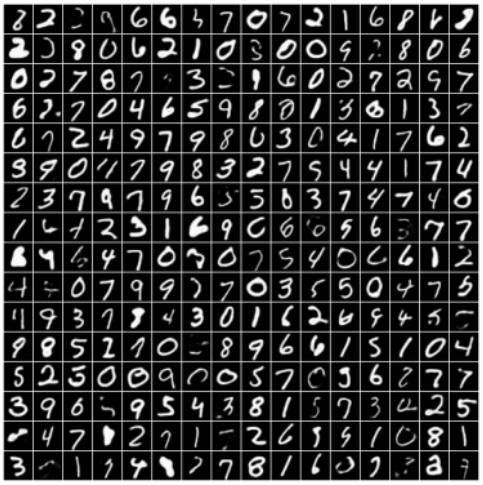

Figure 5.4: Samples generated by the modulation network approach trained on white noise

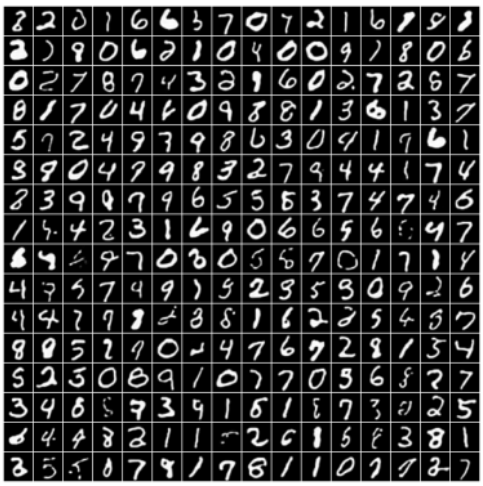

Figure 5.5: Samples generated by the U-ViT network trained on white noise

Performance as denoisers We also compared the performance of the two models in terms of denoising capabilities. First, we show an original batch of data and we perturb it with some random noise:

Figure 5.6: Real batch of data

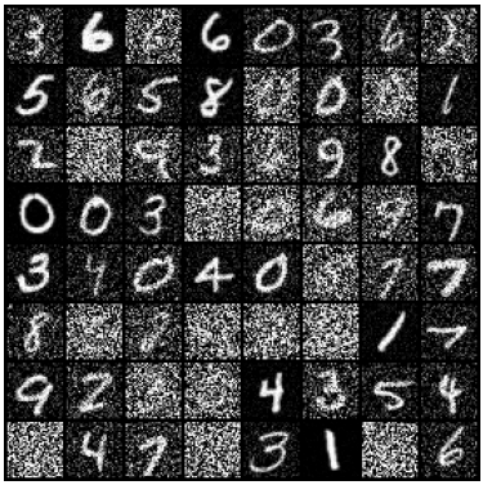

Figure 5.7: Noisy batch of data

Then, we show the denoised batch of data produced by the modulation network approach and by the U-ViT network:

Figure 5.8: Data denoised using the modulation network approach

Figure 5.9: Data denoised using the U-ViT network

We can see that both models are able to denoise the data, however, both may produce different digits than the real ones in presence of high noise. We repeat the same procedure with coloured noise, with similar results:

Figure 5.10: Real batch of data

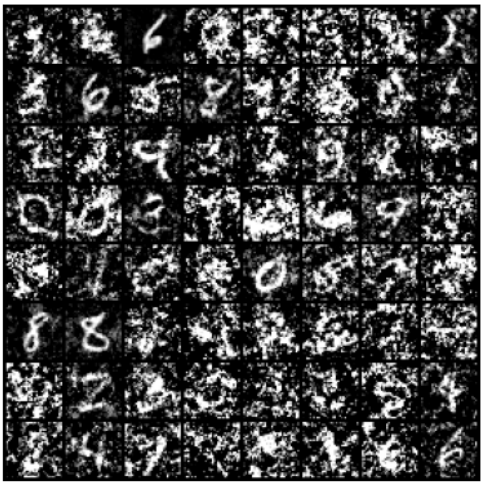

Figure 5.11: Noisy batch of data

Figure 5.12: Data denoised using the modulation network approach

Figure 5.13: Data denoised using the U-ViT network

5.3.2 Results with driftless SDE

In this section, we experiment with the driftless SDE. We use the U-ViT architecture to focus on the diffusion equation and employ different diffusion times $T = \{10,20,100\}$ to see how the model behaves. In this case, the samples have been generated using a predictor-only algorithm. Additionally, we compare the performances when the covariance operator is set to the identity, that is in the case of white noise, and when it is estimated from the data. We show the samples generated by the U-ViT network with the driftless SDE for different diffusion times with white noise:

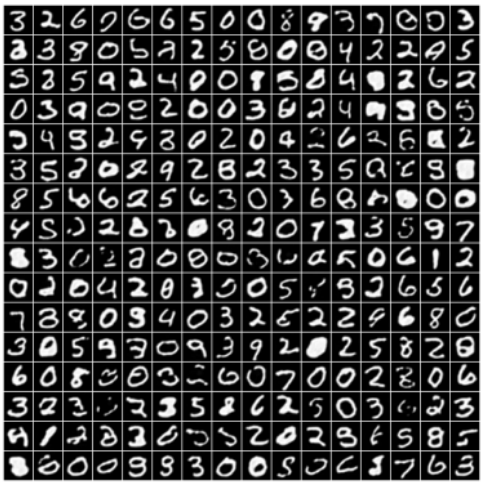

Figure 5.14: Images generated with $T = 10$ and white noise

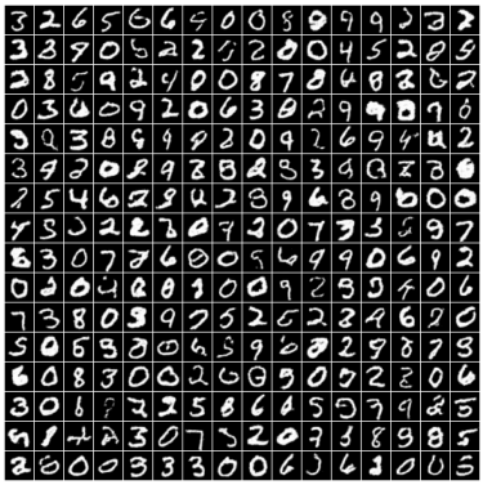

Figure 5.15: Images generated with $T = 20$ and white noise

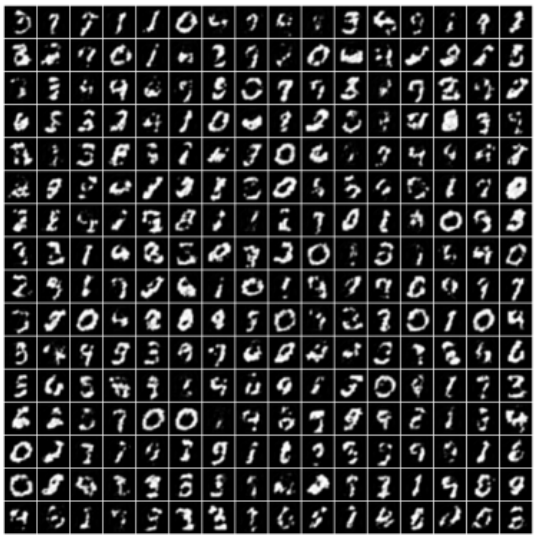

Figure 5.16: Images generated with $T = 100$ and white noise

We can notice how diffusion times affect the quality of the samples. In particular, we can see that the samples generated with $T = 100$ are of worse quality than the ones generated with $T = 10$ and $T = 20$.
Now, we repeat the same analysis when estimating the covariance operator from the data. However, we denote that models trained with $T = 10$ and $T = 20$ benefitted by increasing the diffusion time during inference by a factor of 1.5.

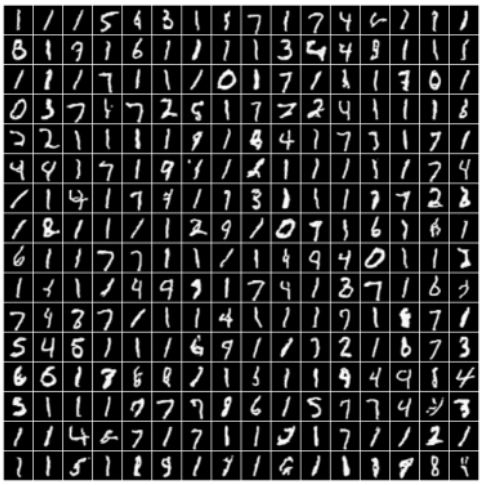

Figure 5.17: Images generated with estimated covariance operator and $T = 20$ in both training and inference

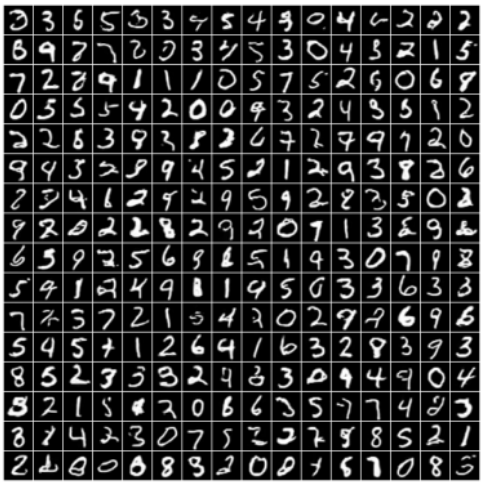

Figure 5.18: Images generated with estimated covariance operator and $T = 20$ in training and $T = 30$ in inference

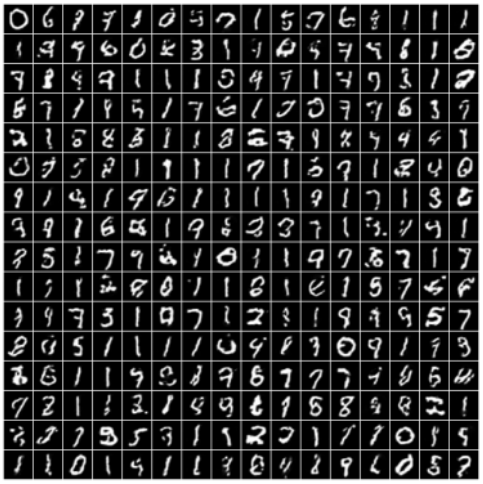

Figure 5.19: Images generated with estimated covariance operator and $T = 100$ in both training and inference

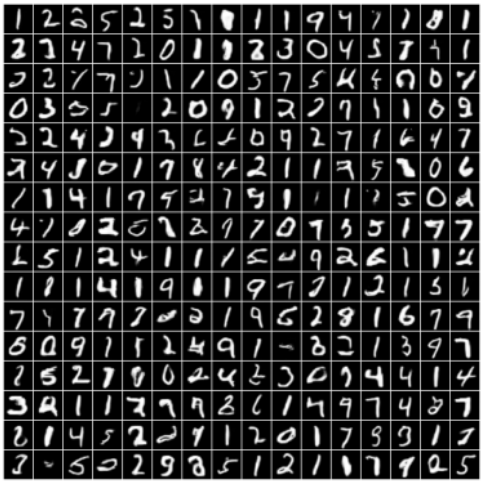

Figure 5.20: Images generated with estimated covariance operator and $T = 10$ in training and $T = 15$ in inference

In order to correct this behaviour, we tried training the model sampling diffusion times from the distribution such that $\log(t) \sim \text{Uniform}(\log(T_{\min}), \log(T_{\max}))$, similarly to what is done for training diffusion models for the VE SDE, where the noise scales are sampled with the same distribution during training.

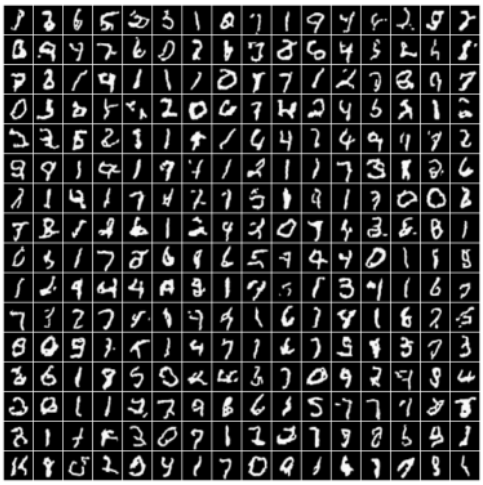

Figure 5.21: Images generated with estimated covariance operator, estimated covariance operator, $T = 20$ and following VE SDE best practises

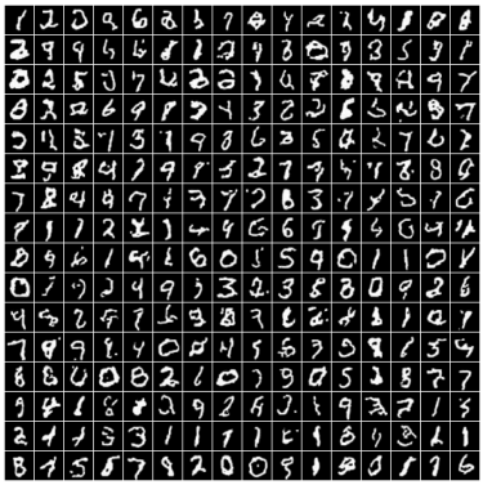

Figure 5.22: Images generated with estimated covariance operator, estimated covariance operator, $T = 100$ and following VE SDE best practises

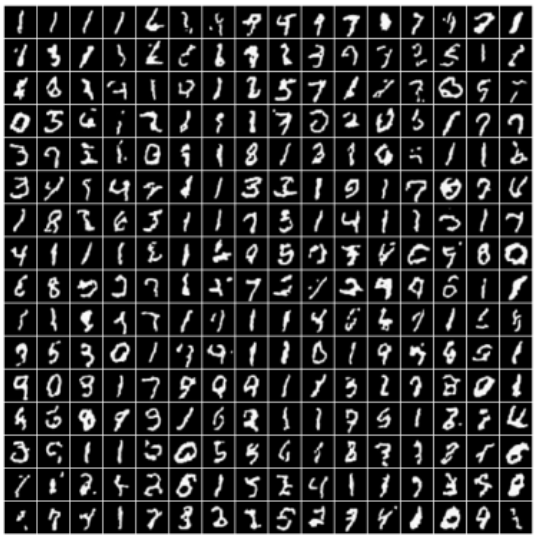

Figure 5.23: Images generated with estimated covariance operator, estimated covariance operator, $T = 1000$ and following VE SDE best practises

Our experiments have shown that following the practises of the VE SDE, we can improve the robustness of the model towards arbitrary choices of diffusion times in training and inference.

5.3.3 Evaluation

In order to evaluate our models, we employed the Fréchet Inception Distance (FID) Heusel et al. [2017]. The FID is a metric that measures the similarity between two sets of images, in our case the real images and the generated images. The FID is calculated by first computing the statistics of the activations of the Inception network Szegedy et al. [2016] for the real images and the generated images. Then, the FID is calculated as the Fréchet distance between the two multivariate Gaussian distributions. The lower the FID, the more similar the generated images are to the real images.
The drawback of this metric is that it does not measure abstraction, since it is based on the similarity of the generated samples with respect to an existing dataset. In our case, however, we can use the FID to compare the quality of the samples generated by the different models since we are not aiming to generate new digits but to recreate existing ones with plausible writing styles.
Firstly, we show the FID scores for the models trained with the general SDE:

	Modulation approach	U-ViT
White noise	22.26	14.88
Coloured noise	26.01	**13.00**

Table 5.1: Results of the FID score for the models trained with the general SDE

As we can see, the U-ViT outperforms the modulation network approach in terms of FID score. However, as shown by the previous generated samples, the quality of the samples is similar, with the modulation network approach producing smoother samples. This difference in FID score can be explained by hyperparameters tuning and network configuration, since transformers networks are widespread in applications and more heuristic is available to tune them.
We also trained a modulation approach model with $O(10M)$ parameters with white noise to verify if the quality of the samples could improve. From the generated samples in fig. 5.24, we can verify that this approach shows promise in scaling with respect to the number of parameters. For what concerns the driftless SDE, instead, we can see that the performances come close to the performances with the general SDE. In particular, we show the FID scores for the models trained with the driftless SDE:

	$T = 10$	$T = 20$	$T = 100$
$R = \hat{R}$	**19.97**	22.24	63.52
$R = I$	40.99	23.56	140.63

Table 5.2: Results of the FID score for the models trained with the driftless SDE

We can notice how estimating the covariance operator from data improves performances with respect to setting $R = I$.
We also experiment by including the best practises for training and sampling with the VE SDE, however they significantly reduce the performances.

$T = 20$	$T = 100$	$T = 1000$
43.83	**41.38**	47.19

Table 5.3: Results with VE SDE best practises

This shows that the driftless SDE still requires appropriate heuristic to be trained and sampled to its full potential. In particular, the heuristic of the VE SDE could reasonably be altered without revolutionizing them, as we can see samples produced with this approach have a higher level of details and do not show oversmoothing.

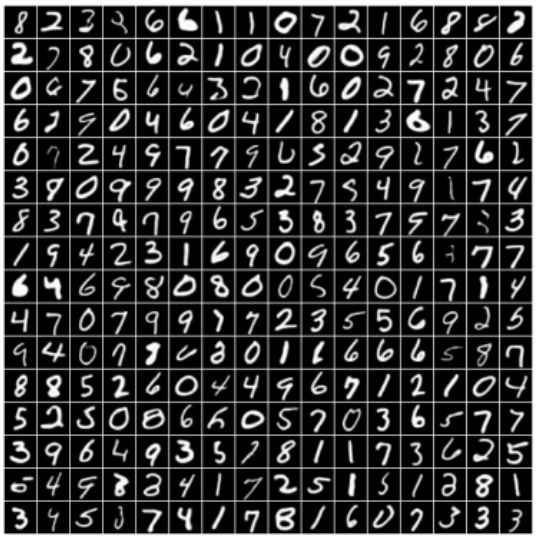

Figure 5.24: Images generated with a bigger modulation approach model using the general SDE

Chapter 6

Conclusions

In this work, we have explored the intricacies of continuous-time functional diffusion processes. These processes allow diffusion in infinite-dimensional spaces, which enable interesting properties such as resolution invariance in image synthesis. We have proposed a new architecture, which is able to integrate transformers with modulation networks, allowing for a more stable training while preserving properties of infinite-dimensional diffusion.
We have also proposed a new formulation of the SDE: the driftless SDE, which removes the drift term, making the model easier to train and more stable. Interestingly, we discovered that the driftless SDE can be seen as a generalization of the Variance Exploding SDE, which is a well-known model in the literature. This allowed us to use the same best practises for the Variance Exploding SDE to improve the driftless SDE. Our experiments confirm that the proposed methodologies perform similarly as the approaches present in the literature, with similar scaling properties and denoising capabilities.
Different research directions can be explored starting from this work:

- The driftless SDE can be further improved with techniques such as importance sampling, or tested with different SDE solvers.
- The model can be tested on more complex datasets, such as CIFAR-10, to see how it scales with higher dimensional data.
- The proposed method can be extended to different modalities and multimodal environments, such as audio and video.
- The model can be tested on different tasks, such as super resolution of images.

Further development in these directions can be beneficial for many practical use cases. For instance, our approach can be extended for the generation of videos at arbitrary resolution and arbitrary frame rate. It must also be remarked that this work could be used for malicious intents, such as the generation of deepfakes.

Bibliography

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

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

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models, 2023.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Salamon Bühler. Functional analysis. https://people.math.ethz.ch/~salamon/PREPRINTS/funcana.pdf, 2017.

Giuseppe Da Prato and Jerzy Zabczyk. *Stochastic equations in infinite dimensions*. Cambridge university press, 2014.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Emilien Dupont, Hyunjik Kim, SM Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. *arXiv preprint arXiv:2201.12204*, 2022.

Giulio Franzese, Giulio Corallo, Simone Rossi, Markus Heinonen, Maurizio Filippone, and Pietro Michiardi. Continuous-time functional diffusion processes. *Advances in Neural Information Processing Systems*, 36, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In

Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Allen Hatcher. Notes on introductory point-set topology. http://pi.math.cornell.edu/~hatcher/Top/TopNotes.pdf, 2009.

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

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

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34:22863–22876, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*, volume 23. Springer Science & Business Media, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Annie Millet, David Nualart, and Marta Sanz. Time reversal for infinite-dimensional diffusions. *Probability theory and related fields*, 82(3):315–347, 1989.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21 (132):1–62, 2020.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Dietmar A. Salamon. Measure and integration. https://people.math.ethz.ch/~salamon/PREPRINTS/measure.pdf, 2020.

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

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

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

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

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

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. *arXiv preprint arXiv:2310.12397*, 2023.

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

Diffusers Team. Diffusers: A pretrained diffusion model library, 2024. URL https://github.com/huggingface/diffusers. Version 0.27.2.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.