



Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA



UNIVERSITÀ
DEGLI STUDI
FIRENZE

DOTTORATO DI RICERCA IN FISICA E ASTRONOMIA

CICLO XXXVIII – PNRR NextGeneration EU

Quantum Diffusion Models for Generative AI towards real-world applications

Settore Scientifico Disciplinare

PHYS-04/A

Dottorando

Dott. Marco Parigi

Marco Parigi

Supervisore

Prof. Filippo Caruso

Filippo Caruso

Coordinatore

Prof. Giovanni Modugno

Giovanni Modugno

Quantum Diffusion Models for Generative AI towards real-world applications

PhD thesis. University of Florence

© 2025

Author: Marco Parigi.

All rights reserved.

This thesis has been typeset by $\text{\LaTeX}$ and the UniFiTh class.

Version: April 13, 2026

The consultation of the thesis is free. Unless a specific authorization is obtained from the author, the thesis can be, however, downloaded and printed only for strictly personal purposes related to study, research and teaching, with the explicit exclusion of any use that has – even indirectly – a commercial nature.

Author's email: marco.parigi@unifi.it

Dedicated to B.

Acknowledgements

Questi tre anni, se pur segnati da sfide e momenti di difficoltà, sono stati pieni di soddisfazioni. Desidero esprimere la mia gratitudine a tutte le persone che mi hanno accompagnato in questo percorso.

Un particolare pensiero va ai miei genitori, per il sostegno e l'incoraggiamento costante nel coltivare la mia passione per la fisica quantistica.

Un sentito ringraziamento al mio supervisore, Filippo, per le opportunità che mi ha offerto, per aver creduto fin dall'inizio nel progetto e per essere stato una guida preziosa in questi anni.

Ringrazio Stefano, inizialmente collega e poi amico, con cui ho condiviso lunghe giornate di lavoro tra modelli matematici e codici, e altrettante conversazioni e confronti sulla vita. L'emozione di vedere le prime immagini generate con nostro modello resterà per sempre un ricordo indelebile.

Un grazie va anche a Francesco, con cui ho trascorso tre intensi mesi durante il mio periodo di ricerca a Barcellona, che hanno contribuito in modo importante alla mia crescita personale e professionale.

Además, deseo agradecer a mis supervisores en Barcelona, Miguel y Alba, por su cálida hospitalidad y el entusiasmo con el que me recibieron.

Finalmente, un agradecimiento especial a mi novia Valeria, quien estuvo a mi lado durante la redacción de esta tesis. Su apoyo constante, energía y su actitud siempre positiva fueron fundamentales para mí.

List of Publications

International Peer-reviewed Journals

1. **Parigi, M.**, Khosrojerdi, M., Caruso, F., and Banchi, L. *Analysis of Quantum Image Representations for Supervised Classification*. AVS Quantum Science 8(1), 013801 (2026).
2. **Parigi, M.**, Martina, S., and Caruso, F. *Quantum-Noise-Driven Generative Diffusion Models*. Advanced Quantum Technologies 8(12), 2300401 (2025). To the best of our knowledge, this article¹ represents the first generalization of classical generative diffusion models to the quantum domain. Furthermore, our proposed cover illustration was selected for the back cover of the issue [1], which is available here: Back cover view.

Preprints

1. **Parigi, M.**, Martina, S., Venturelli, F. A., and Caruso, F. *Physics-inspired Generative AI models via real hardware-based noisy quantum diffusion*. Preprint arXiv:2505.22193 (2025).

¹The published article is the final version of the arXiv preprint released on 23 August 2023, which is available at the following link: <https://arxiv.org/abs/2308.12013v1>.

Contents

Acknowledgements	iii
Publications	iv
Introduction	1
1 Artificial Intelligence, Machine Learning and Deep Learning	4
1.1 Definitions and conceptual hierarchy	4
1.2 Learning problems	6
1.2.1 Data as experience	7
1.2.2 Tasks and models	8
1.2.3 Loss function as performance measure	9
1.3 Training	10
1.3.1 Risk minimization	11
1.3.2 Maximum likelihood estimation	12
1.3.3 Optimization	13
1.4 Methods for ML and DL problems	15
1.4.1 Linear models	15
1.4.2 Artificial neural networks	15
1.4.3 Kernel methods	19
2 Generative AI and Diffusion Models	23
2.1 Historical overview	23
2.2 The importance of synthetic data	25
2.3 Deep generative modeling	26
2.3.1 Discriminative vs generative models	26
2.3.2 Latent space in data generation	26
2.4 Diffusion models	28
2.4.1 Denoising diffusion probabilistic models	29
2.4.2 Score-Based diffusion model	32
2.4.3 Discrete denoising diffusion probabilistic models	34
3 Quantum Computation	36
3.1 Quantum bit	36
3.2 Multi-qubit systems	38
3.3 Quantum logic gates	39
3.3.1 Single-qubit gates	40

3.3.2	Two-qubit gates	43
3.3.3	Universal gate sets	45
3.4	Measurements and expectation values	45
3.5	DiVincenzo criteria for quantum computation	47
4	Quantum Computation in the NISQ Era	49
4.1	NISQ devices	49
4.2	Density operator formalism	51
4.2.1	Bloch sphere representation	52
4.2.2	Evolution	52
4.2.3	Measurements	53
4.2.4	The reduced density operator	53
4.2.5	Fidelity and von Neumann entropy	54
4.3	Quantum operations	55
4.3.1	Quantum system coupled to environment	56
4.3.2	Kraus representation	57
4.3.3	Quantum operation as CPTP map	58
4.4	Examples of quantum operations on a single qubit	59
4.4.1	Bit flip channels	59
4.4.2	Phase flip channel	59
4.4.3	Depolarizing channel	60
4.4.4	Amplitude damping	61
4.4.5	Phase damping	62
4.5	Kossakowski-Lindblad-Gorini master equation	62
5	Quantum Machine Learning	64
5.1	Overview and the four approaches	64
5.2	Quantum encoding for classical data	66
5.2.1	Basis encoding	66
5.2.2	Amplitude encoding	66
5.2.3	Angle encoding	67
5.3	Variational quantum circuits as learning models	67
5.3.1	Deterministic quantum learning models	68
5.3.2	Probabilistic quantum learning models	69
5.4	Training	70
5.5	Quantum kernels	72
6	Quantum Encoding for Image Classification	74
6.1	Background and motivation	74
6.2	Quantum encoding for images	75
6.2.1	Tensor network representation	75
6.2.2	Flexible Representation of Quantum Images	77
6.2.3	Novel Enhanced Quantum Representation	78
6.2.4	Quantum Probability Image Encoding	79
6.3	Our results	80

7	Quantum Diffusion Models: A Novel Paradigm in Quantum AI	87
7.1	The rise of quantum diffusion models	87
7.2	Results	89
7.2.1	Classical-Quantum Generative Diffusion Model	89
7.2.2	Quantum-Classical Generative Diffusion Model	90
7.2.3	Quantum-Quantum Generative Diffusion Model	93
7.2.4	Quantum stochastic dynamics for image generation	95
7.2.5	Real hardware-based noisy quantum diffusion	100
7.2.6	Applications to medical and real-world datasets	103
8	Conclusions	106
A	Cross-validation	109
B	Classical random walks and quantum walks	110
C	MNIST image generation	113
	Bibliography	135

Introduction

The history of intelligence is closely tied to that of humanity. Since the earliest civilizations, humans have sought to understand and reproduce the mechanisms of thought. From Greek philosophy, with Plato and Aristotle investigating the nature of the mind and consciousness, to Cartesian rationalism and the Enlightenment, intelligence has been considered a fundamental faculty of humanity, capable of understanding the world around us by abstracting universal concepts from experience.

With the Industrial Revolution, the concept of intelligence became increasingly intertwined with technology: machines were no longer mere instruments of mechanical force but began to embody rudimentary forms of calculation and automation. In 1837, Charles Babbage designed the *Analytical Engine*, a mechanical device intended to perform logical computations, which is considered the first prototype of a general-purpose computer [2].

Later, with the advent of computer science in the 20th century, the idea of *Artificial Intelligence* emulating the human one began to take shape. In 1943, Warren S. McCulloch and Walter Pitts proposed a mathematical model of the behavior of a biological neuron [3]. A few years later, in 1950, the British mathematician Alan Turing introduced his famous *Turing Test*, a method to evaluate whether a machine can demonstrate intelligent behavior indistinguishable from that of a human [4]. Since then, the relationship between humans and intelligence has evolved into an ongoing dialog between philosophy, science, and technology. In the late 1950s, Frank Rosenblatt implemented the *Perceptron*, a model inspired by the functioning of the human brain [5, 6]. However, the limitations of these early approaches led to a period of stagnation known as (the first) *AI winter*, marked by reduced funding and interest. A renaissance occurred with the introduction of *backpropagation* [7], which enabled efficient training of multilayer neural networks and laid the foundation for modern *Deep Learning* [8].

Recently, the advent of *Generative Artificial Intelligence* has marked a significant turning point in the history of human-computer interaction. In particular, The emergence of deep generative models such as Generative Adversarial Networks (GANs) [9], Variational Autoencoder (VAEs) [10] and Diffusion Models (DMs) [11] represented a decisive starting point allowing neural networks not only to analyze and classify data, but also to create meaningful and original content. In 2017, the Transformer architecture [12] introduced a new paradigm, which enabled the construction in 2018 of GPT-1 [13], the first pre-trained generative language model. Generative AI has experienced widespread adoption and significant improvements thanks to advances in Transformer-based architectures and the development of Denoising Diffusion Probabilistic Models [14], which have achieved state-of-the-

art performance in image and audio synthesis, surpassing GANs and VAEs in terms of quality and diversity [15, 16]. Today, platforms such as DALL·E [17], Stable Diffusion [18] and Midjourney [19], which are based on diffusion generative models, represent a significant advance in image generation, such as 3D images, artwork, and photorealistic drawings based on text descriptions. Furthermore, Google DeepMind [20] recently announced the releases of Gemini Diffusion [21], a diffusion-based Large Language Model.

In parallel with the advancement of AI technologies, *Quantum Computing* has developed from the fundamental idea of using quantum mechanical properties to manipulate and process information encoded in quantum states. In the 1980s, Paul Benioff [22] and Richard Feynman [23] laid the theoretical foundations by envisioning quantum machines capable of performing computation and simulating complex physical systems beyond the capabilities of classical computers. Later, Shor’s algorithm for factorization [24] and Grover’s algorithm for search [25] demonstrated the revolutionary potential of quantum computation. Despite significant engineering progress, the current available devices belong to the *Noisy Intermediate-Scale Quantum* (NISQ) era [26], characterized by a limited number of noisy qubits. This has led to the necessity of designing quantum algorithms oriented toward specific application domains. This phase represents a transition period in which research is directed toward identifying concrete applications and bridging the gap between theoretical models and practical implementations.

In this context, the convergence of artificial intelligence and quantum computing plays a crucial role, as it has the potential to enhance the stability and efficiency of quantum algorithms. Building on this synergy, the field of *Quantum Machine Learning* (QML) has emerged [27, 28], harnessing the properties of quantum systems to improve the performance of machine learning tasks such as classification, regression, and generative modeling.

The main focus of this thesis is on *Quantum Diffusion Models* (QDMs), a novel framework in generative artificial intelligence, proposed by us in 2023 [29]. The idea is to employ Quantum Information Science tools to address highly complex problems or even to directly process quantum data, which are often difficult to implement on classical hardware. A crucial aspect is the use of *quantum noise*, which in this thesis is not treated as a limitation but rather as a valuable resource for quantum generative models.

This thesis is structured as follows. In Chapter 1, we begin by introducing the fundamental concepts of artificial intelligence, machine learning, and deep learning, providing the essential tools for the construction and implementation of structured artificial neural networks such as Multilayer Perceptron (MLP). We also present kernel methods, which are powerful techniques for classification tasks that extend data representation beyond linear models.

Chapter 2 is dedicated to generative artificial intelligence, offering an overview of the most widely used models such as GANs and VAEs, and discussing the role of stochastic dynamics as a key ingredient in generative tasks. Particular attention is devoted to diffusion models, which currently represent the state of the art in this field. We also discuss their different formulations, including continuous-time and discrete-time approaches, as well as extensions for categorical data types.

In Chapter 3, we introduce the main tools of quantum computing in an ideal

noiseless scenario, starting from the fundamental notion of the quantum bit, in which information is encoded. Next, we highlight the differences between the quantum and classical paradigms, emphasizing the advantages of quantum computing, such as the ability to exploit superposition and entanglement to represent and process information in ways inaccessible to classical systems. Furthermore, we describe how to process and manipulate quantum information stored in a (multi-) qubit system using quantum logic gates.

Chapter 4 extends this discussion to currently available quantum devices, which are inherently noisy and consist of only a few hundred qubits. Here, we introduce the formalism of *Open Quantum Systems* as a framework for modeling the dynamics of such devices. We begin with a description of the current state of quantum processors, which are known as Noisy Intermediate-Scale Quantum (NISQ) devices. We then present the density operator formalism, which is crucial when dealing with systems whose states are not exactly known, followed by the formalism of quantum operations, which allows us to describe both unitary and non-unitary dynamics and to mathematically model quantum noise.

Next, Chapter 5 introduces the emerging field of quantum machine learning, a promising area of research that combines machine learning techniques with the unique features of quantum mechanics. We discuss the main approaches, strategies for encoding classical data into quantum states, the construction of quantum neural networks, and the use of quantum kernels.

The last part of this thesis presents the results of my doctoral research. In Chapter 6, we examine four quantum techniques for encoding grayscale images, evaluating their ability to compress classical information when encoded in quantum states. We also study binary image classification using quantum kernels and compare their performance with that of classical kernels on the MNIST [30] and Fashion-MNIST [31] datasets. The performance of quantum kernels based on the four quantum image representations is comparable to that of linear kernels.

Chapter 7 constitutes the core of the manuscript, in which we extend classical generative diffusion models to the quantum domain, introducing the Quantum Diffusion Model paradigm. We begin formalizing the three scenarios: hybrid dynamics, partially quantum dynamics, and fully quantum dynamics, each supported by numerical simulations. We further analyze the impact of quantum and hybrid diffusion processes on image generation, demonstrating that the interplay between classical and quantum dynamics can yield more robust generative models, capable of producing image datasets with lower Fréchet Inception Distance (FID) scores [32]. Next, we propose and implement a hybrid quantum–classical diffusion model based on quantum walks, executed on real NISQ hardware. In particular, we propose a model that uses the intrinsic quantum noise of the device to generate MNIST images. Finally, we present preliminary results that demonstrate the applicability of our model to medical imaging tasks. Specifically, we apply our model to generate colored images from BloodMNIST [33] of size 64×64 and 200×200 grayscale MRI scans from BraTS2020 [34].

Chapter 1

Artificial Intelligence, Machine Learning and Deep Learning

Artificial intelligence is a branch of computer science that aims to design systems capable of performing tasks by emulating some human capabilities. These tasks include learning from data, language understanding, pattern recognition, creation of new content, and autonomous decision-making. In this chapter, we attempt to provide an overview of the broad field of artificial intelligence, with a focus on machine learning and deep learning [8, 28]. In Section 1.1, we begin by defining the terms artificial intelligence, machine learning, and deep learning, their hierarchical relationships, and their respective domains of application. Next, in Section 1.2, we define the structure of a learning problem from a computational perspective, describing its three fundamental elements: *data*, *models*, and *loss function*. Section 1.3 describes the training procedure and optimization techniques used for supervised and unsupervised learning problems. This chapter concludes in Section 1.4 where we describe the computational model of *artificial neural networks* and the kernel method of *support vector machines*.

1.1 Definitions and conceptual hierarchy

Today, terms such as Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) have entered the public consciousness. Although they are often used interchangeably, they actually represent hierarchically related but distinct disciplines, as illustrated in Figure 1.1.

The term “artificial intelligence” was formally coined in 1955 in a workshop proposal by John McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon [35]. It includes all technologies and systems that enable machines to emulate human behavior and cognitive abilities to solve complex problems such as reasoning, learning from experience, interacting with other systems, making autonomous decisions, creating new content, and understanding language [36]. Thus, it is the broadest field in the hierarchy that includes ML and DL as subfields.

ML is a branch of AI that focuses on the ability of machines *to learn* relationships based on data to make predictions and decisions, using a different computational paradigm [37, 38]. In traditional computer science paradigms, programmers explicitly

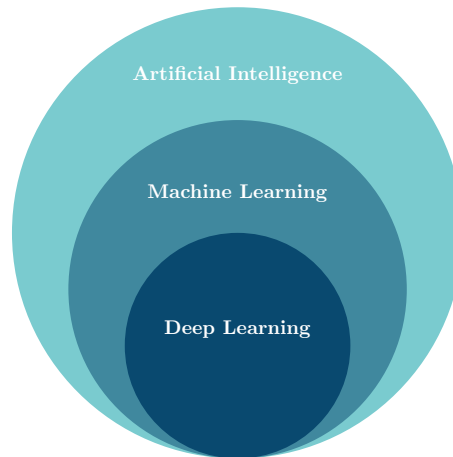


Figure 1.1. Hierarchical relationship among Artificial Intelligence, Machine Learning, and Deep Learning.

provide computers with a set of instructions to transform input data into appropriate output. In contrast, in ML algorithms, the programmer provides the machines with input data, and starting from an agnostic mathematical model, the systems learn to recognize patterns and structures within the data through an iterative process known as *training*. In particular, a ML algorithm is trained to *autonomously* identify the most useful representation of the input data within a set of transformations called *hypothesis space*, to perform and solve a specific task. Depending on the type of learning problem, there exist various techniques or algorithms that can be implemented, such as regression models, support vector machines, k-nearest neighbors, clustering, and others. However, one of the most popular approaches today involves the use of *artificial neural networks*, which are computational models inspired by the information processing mechanisms of the human brain. Artificial neural networks consist of mathematical representations of connected processing units called *artificial neurons* that are organized into *layers*. An input layer usually receives the input data, and an output layer produces the final result. In between, there can be few *hidden layers* responsible for learning the optimal function between input and output¹. Due to their *flexible* structure, neural networks can solve a wide variety of real-world data problems.

Finally, DL is a branch of ML that relies on computational models known as *deep neural networks*, namely artificial neural networks consisting of a large number of hidden layers (hence the term “deep”) [8, 40]. Unlike traditional ML algorithms, which typically require manual preprocessing operations on input data to improve model performance, the more sophisticated architecture of deep neural networks enables DL systems to automatically learn hierarchical and increasingly abstract representations directly from a large volume of raw input data. Due to this capability, along with advances in hardware components such as Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs), DL systems have had a profound

¹There exist also other architectures that do not follow the above feed-forward structure, such as the Hopfield networks [39], whose inventor was awarded the Nobel Prize in Physics in 2024.

impact in fields such as medical diagnostics, drug discovery, computer vision, finance, and many others.

In summary, AI is the overarching concept that includes all technologies that are designed to emulate human behavior. ML is a field of AI that focuses on learning problems, allowing machines to learn and draw inferences from data. DL is a powerful approach within ML, capable of solving complex problems that involve a large amount of high-dimensional data.

1.2 Learning problems

In ML and DL, algorithms are capable of autonomously learning from data. But what do we mean by the term *learning*? A succinct definition is provided by Tom M. Mitchell in [41] as follows: “A computer program is said to learn from experience E with respect to a certain class of tasks T and a performance measure P if its performance on tasks T , as measured by P , improves with experience E .” From a computational perspective, the experience E is the set of data samples we provide to algorithm; the task T is associated with, and mathematically represented by, a function that the algorithm aims to learn in order to achieve its goal; and the performance measure P is the metric used to evaluate the performance of the algorithm. Learning problems can be categorized into three main types:

- **Supervised Learning:** the algorithm learns from a set of labeled data samples a function that maps the inputs to the corresponding outputs.
- **Unsupervised Learning:** the algorithm processes a set of unlabeled data samples to detect hidden patterns or structures within the data. This approach is often applied in clustering, dimensionality reduction, or generative models.
- **Reinforcement Learning:** an agent interacts with the environment and learns through a trial-and-error strategy, guided by rewards and penalties, to optimize its actions over time to achieve a specific goal. This approach is used in applications such as autonomous driving, robotics, and video games.

In the following of this thesis, we focus on supervised and unsupervised learning problems, providing a formal definition, and presenting their corresponding pipelines.

Supervised learning problems

Supervised learning problems involve labeled data samples and can be formulated as follows:

Definition 1. *Given an input domain $\mathcal{X}$ and an output domain $\mathcal{Y}$, a data set $\mathcal{D} = \{(x^1, y^1), \dots, (x^M, y^M)\}$ of pairs $(x^m, y^m) \in \mathcal{X} \times \mathcal{Y}$, with $m = 1, \dots, M$, where inputs x^m and target outputs y^m are sampled from a joint probability distribution $p(x, y)$. In addition, given a new unclassified input $x \in \mathcal{X}$, the task is to predict the corresponding output $y \in \mathcal{Y}$.*

Note that labeled data sets are often expensive to acquire and manage in practical applications, which can lead to significant computational demands.

Unsupervised learning problems

Unsupervised learning problems involve unlabeled data samples and can be formulated as follows:

Definition 2. *Given an input domain $\mathcal{X}$ and a data set $\mathcal{D} = \{x^1, \dots, x^M\}$ of input samples $x^m \in \mathcal{X}$ drawn from a probability distribution $p(x)$, the task is to draw a new sample from p .*

The definition is relevant for generative tasks; however, unsupervised learning also involves tasks such as clustering and dimensionality reduction, which are not addressed in this manuscript.

The standard approach to address the two problems described above is to define a family of models $\{f\}$, represented mathematically as a set of functions f that map input data from a domain $\mathcal{X}$ to output data in a domain $\mathcal{Y}$, or alternatively as probability distributions p over the input space $\mathcal{X}$. From this family of models, the goal is to identify the function that best captures the underlying structure of the data.

In supervised learning, this goal is formally known as *risk minimization*. The concept of a “best model” is quantified through a figure of merit formalized as a loss function $L(f(x), y)$, which evaluates the discrepancy between the prediction of the model $f(x)$ for a given input x and the corresponding target output y . The optimal model is the one that minimizes the expected loss over all possible data. However, in practical scenarios, only a finite set of samples is available. As a result, the challenge of the learning process is to select a model that not only fits the observed data, but also generalizes to unseen data.

In unsupervised learning, the absence of labeled data means that we cannot define a loss function based on comparisons between predictions and targets. In this context, the learning objective can be interpreted as minimizing the distance between the true data distribution and the estimated distribution by the model.

Next, we provide a detailed description of the three fundamental components of a learning process, that is, data, models, and loss functions.

1.2.1 Data as experience

Learning algorithms rely on data to make predictions and decisions. Thus, data represents the fuel for the learning process. In ML and DL, a common assumption is that the data samples are *independently and identically distributed (i.i.d.)*, which means that each sample is drawn independently of the others and from the same underlying probability distribution.

For simplicity, in the following we will assume that the input data domain $\mathcal{X}$ is the space² $\mathbb{R}^n$. A data sample is represented as a vector $\mathbf{x} \in \mathbb{R}^n$ (we use bold notation only when necessary), where each element x_i describes a distinct *feature* of the sample. For this reason, data samples are also commonly referred to as *feature*

²In some applications, such as image processing, the inputs are naturally represented as multi-dimensional tensors $\mathbb{R}^{l \times m \times \dots}$, however, these can often be flattened into one-dimensional vectors, which justifies the simplification to $\mathbb{R}^n$.

vectors. For example, in image data, the features correspond to the pixel intensity values.

Data samples are often preprocessed to enhance the predictive performance of models [42]. Some common techniques are *feature selection*, which aims to identify the most relevant features of each sample; *feature scaling*, which changes some statistical properties such as mean and variance to normalize the data and avoid unwanted effects due to differing scales; and *feature engineering*, which involves manually constructing informative features. However, with the emergence of deep learning architectures, the importance of data preprocessing has become less central to ML. The multilayered structure of deep neural networks allows DL systems to automatically learn hierarchical and abstract representations directly from raw input data. Due to this capability, DL models are well suited for handling complex, high-dimensional data and solving complex tasks such as image recognition, natural language processing, and audio analysis.

1.2.2 Tasks and models

The second central concept in learning algorithms is the model. Depending on the task, models are described mathematically by a function that maps inputs to outputs, or a probability distribution whose role is to reproduce the statistical properties of a given set of data samples. Thus, we can distinguish between two main classes of models: *deterministic* and *probabilistic* models.

Deterministic models

In a supervised learning problem, a deterministic model can be formalized as follows:

Definition 3. *Let $\mathcal{X}$ be an input domain and $\mathcal{Y}$ be an output domain for a supervised machine learning problem. A deterministic model is described by a parameterized function*

$$f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y}, \quad (1.1)$$

such that $f_{\theta}(x) = y$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and where θ is a set of parameters that identify a specific model. In some cases, a set of hyperparameters may be introduced to provide additional flexibility to the model.

The nature of the output domain $\mathcal{Y}$ depends on the task of the problem. For example, in a *multiclass classification* problem, the output domain $\mathcal{Y}$ is a finite set of k discrete classes. The goal of is to find the best function $f_{\theta} : \mathbb{R}^n \rightarrow \{1, \dots, k\}$, such that when $f_{\theta}(\mathbf{x}) = y$ the model assigns an input vector $\mathbf{x}$ to the corresponding class label identified by a numeric code y . In a regression problem, instead, the output domain $\mathcal{Y}$ is the space of real numbers $\mathbb{R}$. In this case, the goal is to learn the best function $f_{\theta} : \mathbb{R}^n \rightarrow \mathbb{R}$ that describes the relationship between the input data and the output.

Probabilistic models

The second class of models are probabilistic models, which are defined in terms of probability distributions. We consider both supervised and unsupervised probabilistic models.

Definition 4. Let $\mathcal{X}$ be an input domain and $\mathcal{Y}$ be an output domain of a learning problem. For a supervised learning problem, a probabilistic model can be defined by a full joint distribution

$$p_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1], \quad p_\theta(x, y), \quad x \in \mathcal{X}, y \in \mathcal{Y}, \quad (1.2)$$

or a conditional distribution

$$p_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1], \quad p_\theta(y|x), \quad x \in \mathcal{X}, y \in \mathcal{Y}. \quad (1.3)$$

In unsupervised learning, a probabilistic model is defined by a distribution

$$p_\theta : \mathcal{X} \rightarrow [0, 1], \quad p_\theta(x), \quad x \in \mathcal{X}. \quad (1.4)$$

From an algorithmic perspective, a probabilistic model can operate in two distinct ways. In the first case, the algorithm aims to approximate the underlying probability distribution $p_{\text{data}}(x)$ of a given set of data. This class of models is known as *density estimators*, or also as *explicit generative models*. In the context of generative AI, two examples of this case are the families of algorithms known as Variational Autoencoders and Diffusion Models [43]. Alternatively, a probabilistic model may be an algorithm that generates samples without parametrically estimating the data distribution $p_{\text{data}}(x)$. This class of models is called *implicit generative models*, a prominent example of which is the family of algorithms known as Generative Adversarial Networks. We will describe the classes of generative algorithms in more detail in Chapter 2.

1.2.3 Loss function as performance measure

So far we have introduced the concepts of experience E , represented by the data, and task T associated with and formalized by the model. We now describe the third essential component of learning paradigms: the performance measure P . Given a family of possible models $\{f_\theta\}$, each identified by a different set of parameters θ , we need a measure that quantify the performance of each model. This quantity is formalized by an objective function called *loss function* that allows us to select the best model in the model family. There are a wide variety of different loss functions, each suited to different tasks and data types. Next, we outline the loss functions that are essential for understanding the methodologies discussed in this work.

In supervised learning, the most straightforward loss function is based on the *accuracy* of a model, defined as the number of correctly classified examples over the total number of examples. Formally, the accuracy is computed by the following equation:

$$\text{accuracy} = \frac{1}{M} \sum_{m=1}^M 1(f_\theta(x^m) = y^m), \quad (1.5)$$

where M is the size of the training data set, y^m is the true label associated with the sample x^m , and $1(\cdot)$ is the indicator function that returns 1 if the condition is true, 0 otherwise. The corresponding loss function is the *error*, which is defined as:

$$\text{error} = 1 - \text{accuracy}. \quad (1.6)$$

In a learning process, loss functions play a dual role, i.e., not only do they serve as evaluation metrics but they also guide the optimization of model parameters. In fact, during the training process, the model attempts to minimize the loss function in order to improve its predictive performance. Optimization techniques, such as gradient descent, calculate the gradient of the loss function with respect to the model parameters. Therefore, it is crucial that the loss functions be *differentiable* to enable gradient-based optimization techniques. For this reason, the concepts of metric and loss function are often treated separately in the literature: the former is used without constraints only to evaluate the model, while the latter is usually a proxy for the metric and should be formulated so that it can be used during the training optimization process.

One of the most popular differentiable loss functions used in regression tasks is the *Mean Squared Error* (MSE) between prediction $f_\theta(x)$ and target y . It is mathematically expressed as:

$$L(f_\theta(x), y) = \frac{1}{M} \sum_m (f_\theta(x^m) - y^m)^2. \quad (1.7)$$

Another important loss function adopted in classification tasks is the *Cross-Entropy*, which is often used to compare the multidimensional target $\mathbf{y} = (y_1, \dots, y_D)$, where $y \in \{0, 1\}$, with the output probability distribution on the predictions $\mathbf{p} = (p_1, \dots, p_D)$. The Cross-Entropy loss is given by:

$$L(\mathbf{p}, \mathbf{y}) = - \sum_{d=1}^D y_d \log p_d \quad (1.8)$$

A special case is *Binary Cross-Entropy*, which is typically used for binary classification problems. It is defined as:

$$L(p, y) = - [y \cdot \log p + (1 - y) \cdot \log(1 - p)], \quad (1.9)$$

where $y \in \{0, 1\}$ is the true target and p is the predicted probability $p \in [0, 1]$. Binary cross-entropy quantifies the performance of the model by penalizing confident but incorrect predictions more heavily than less confident ones. A lower value indicates better model performance and accuracy.

For unsupervised learning tasks, the quality and performance of a model depend on whether the model provides us with probability $p(x)$, or whether it only returns generated samples x . In the former case, the loss function is typically expressed via the *likelihood function*,

$$L(x) = p(\mathcal{D}|\theta), \quad (1.10)$$

which measures the probability of observing a set of data points $\mathcal{D}$ given a specific set of parameters θ . In the latter case, a common strategy is to turn the unsupervised into a supervised learning problem and use the supervised loss functions. More details will be discussed in Section 1.3.2.

1.3 Training

The training procedure involves the integration of data, models, and loss functions. Supervised and unsupervised learning differ in the way these components are combined to address a given task. In supervised learning, model training is based

on the principle of *risk minimization*, which aims to identify the best model by minimizing the expected loss of a model over the data distribution. In contrast, unsupervised learning strategies depend on the nature of the probabilistic model. When the model is designed to generate samples without explicitly learning the underlying distribution, the problem is often transformed into a supervised learning task using the generated data. Alternatively, if the model aims to reconstruct the data distribution, the best model is typically chosen by the *maximum likelihood estimation* strategy. In the following, we summarize the two approaches.

1.3.1 Risk minimization

In statistical learning theory, the principle of risk minimization refers to the process of finding a model that minimizes the expected loss of a model over the data distribution when making predictions. The expected loss is also called the *generalization error*. The risk minimization principle can be formulated as follows:

Definition 5. Let the inputs $x \in \mathcal{X}$ and the outputs $y \in \mathcal{Y}$ of a supervised machine learning problem be sampled from a distribution $p(x, y)$. The expected loss (or risk) of a model f_θ with loss function L is defined as

$$R_{f_\theta} = \mathbb{E}[L(f_\theta(x), y)] = \int_{\mathcal{X} \times \mathcal{Y}} L(f_\theta(x), y) p(x, y) dx dy. \quad (1.11)$$

The integral runs over all possible data pairs (x, y) and the expectation is taken over the data distribution. A lower value of the risk means better model performance. However, in real-life applications, we only have access to a finite set of data samples, and therefore the integral defined in Eq. 1.11 is impossible to compute. For example, $\mathcal{X}$ being the domain of black and white images of 1 000 pixels, the number of possible images would be 2^{1000} . Consequently, we estimate the risk based on the samples in the dataset $\mathcal{D}$ (as in Definition 1) using the *empirical risk* given by:

$$\hat{R}_{f_\theta} = \hat{\mathbb{E}}[L(f_\theta(x), y)] = \frac{1}{M} \sum_{m=1}^M L(f_\theta(x^m), y^m). \quad (1.12)$$

The goal of training is to select the most suitable model from a parameterized family by minimizing the empirical risk over the dataset. This corresponds to finding the optimal set of parameters:

$$\theta^* = \min_{\theta} \hat{R}_{f_\theta}. \quad (1.13)$$

However, selecting the optimal model by minimizing empirical risk on a given dataset can lead to a model that performs well on the training data but does not generalize to unobserved data. This phenomenon is known as *overfitting*. Various techniques and strategies are adopted to avoid the overfit known as *regularization*. For example, selecting less flexible model families, applying strategies such as early stopping, or adding a regularization term $r(\theta)$ to the loss function L , and minimizing the overall cost function:

$$C(\theta) = \hat{R}_{f_\theta} + r(\theta). \quad (1.14)$$

The regularization term is a function of the model parameters and imposes additional constraints.

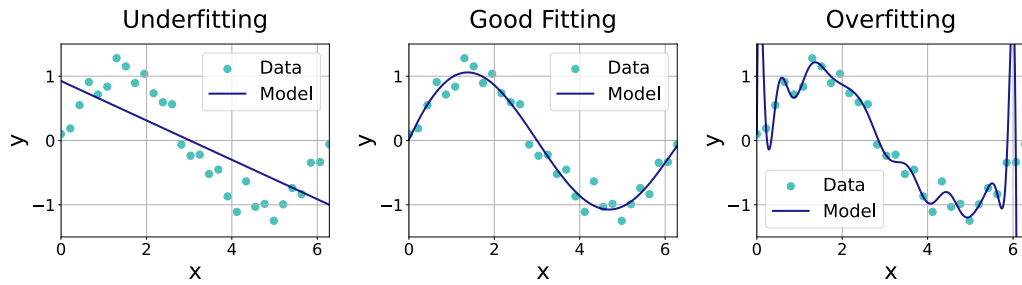


Figure 1.2. Illustration of three possible scenarios for a ML model. On the left, the model is too simple to represent the structure of the data, leading to underfitting scenario. In the middle, the model captures relevant patterns well, balancing complexity and generalization ability. On the right, the model is highly complex and fits the training data perfectly, leading to overfitting scenario, thus compromising its ability to generalize to unseen data.

Finding a model that generalizes well based on training data is one of the main challenges in machine learning problems. This is typically achieved by the so-called *bias-variance trade off*. In fact, models with high bias are too simple to capture the complexity of the data structure. In contrast, a high variance generates models that are too flexible and overfit the training data. Therefore, balancing *expressiveness* and *capacity* is a fundamental requirement to build models that generalize well beyond the training data. Figure 1.2 illustrates the three possible scenarios: underfitting (high bias, low variance), good fits (balanced bias and variance) and overfitting (low bias, high variance). To identify the minimum of the bias-variance trade-off (i.e., the optimal model), during the training procedure, the data set is typically divided into three subsets. A *training set*, which is used to minimize the cost function, a *validation set*, which is used to tune hyperparameters, and a *test set*, which is adopted to evaluate the model on unseen data.

1.3.2 Maximum likelihood estimation

In unsupervised learning, the training procedure aims to minimize the distance between the model distribution and the original data distribution. However, as mentioned above, the probabilistic model can be designed to address two different tasks: generate samples without explicitly estimating the data distribution, or learn an approximated form of the original data distribution. In the former case, data distribution is only accessible via generated samples, and the problem is turned into a supervised learning problem. For instance, in adversarial learning, a generative model is trained to fool a supervised model that has to distinguish (classify) real from generated samples. In the latter case, if a probabilistic model is a density estimator, a common strategy adopted to evaluate the performance of the model is *maximum likelihood estimation*. The goal is to maximize the likelihood that the training samples were generated by the model distribution. In the following, we apply the principle of maximum likelihood estimation in the context of unsupervised learning.

Suppose that we have a training data set $\mathcal{D}$ of observed data x^m , with $m = 1, \dots, M$. The goal of training is to find an optimal set of θ parameters for which

the model can reproduce the observed data with a high probability. This can be achieved using the Bayes theorem [44]:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}, \quad (1.15)$$

In Eq. 1.15, the *posterior* $p(\theta|\mathcal{D})$ is the conditional probability that a certain set of parameters accurately describes the underlying process given the observed data; the *prior* $p(\theta)$ describes our assumption on the best set of parameters before observing the data; the *likelihood* $p(\mathcal{D}|\theta)$ is the probability of seeing the set data given certain parameters; and the term $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta$ is a normalization factor. Thus, the task of estimating the probability of a set of parameters given the data is turned into evaluating the likelihood of observing the data under a model defined by those parameters. Assuming that the data samples are drawn independently, we can factorize the likelihood and write:

$$p(\mathcal{D}|\theta) = \prod_{m=1}^M p_{\theta}(x^m), \quad (1.16)$$

where $p_{\theta}(x)$ is the model distribution that we have chosen. The expression on the right can be calculated by evaluating the model M times (i.e., one for each data sample) and then multiplying the results. However, for high-dimensional parameters, the exact solution quickly becomes computationally intractable. Therefore, we use the maximum likelihood estimation strategy [45] to find the best model. This is achieved by maximizing the likelihood $p(\mathcal{D}|\theta)$ of Eq. 1.16. In practice, to make the problem easier to solve computationally, we maximize the log-likelihood:

$$\log p(\mathcal{D}|\theta) = \log \prod_{m=1}^M p_{\theta}(x^m) = \sum_{m=1}^M \log p_{\theta}(x^m). \quad (1.17)$$

In summary, the problem of the maximum likelihood estimation reduces to identifying the parameters θ that maximize Eq. 1.17, a task that can be addressed using the optimization methods presented in the next paragraph.

1.3.3 Optimization

In the previous section, we saw that solving a machine learning problem translates into solving an optimization problem. The mathematical theory of optimization is the field that studies how to solve such problems. The nature of the optimization problem and the tools available to solve it depend largely on the type of learning model and the loss function. The most important distinction is between *convex*³ and *non-convex* problems, where convexity refers to the objective function. Convex problems have a solid and extensive mathematical theory and are generally easy to compute and solve. In contrast, non-convex problems are significantly more difficult to treat [46]. For example, the mean-squared error loss function combined with a

³A convex set is a set in which, for every pair of points, the segment joining them is entirely contained within the set. A function $f : \mathcal{X} \rightarrow \mathbb{R}$ is convex if $\mathcal{X}$ is a convex domain and if a line connecting any two points of the function lies above the function itself.

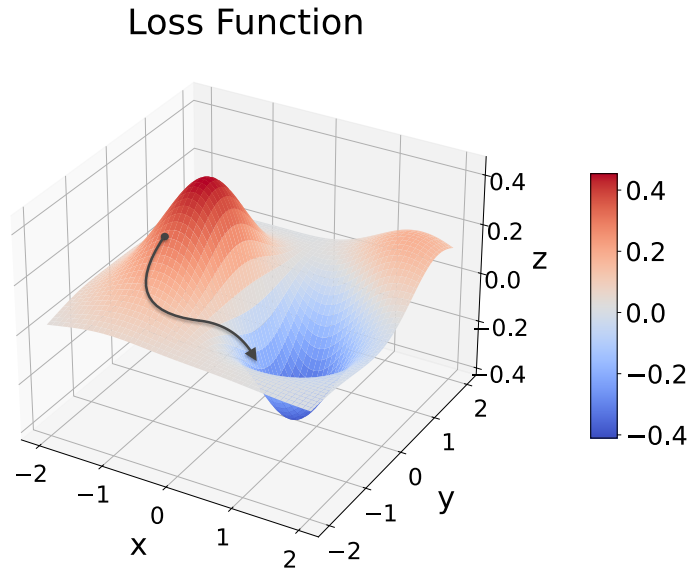


Figure 1.3. Illustration of gradient descent optimization techniques on a three-dimensional surface. The trajectory (black) follows the direction of maximum descent to minimize the loss function.

model function that is linear in its parameters constitutes a simple convex quadratic optimization problem that has a closed-form solution. However, many real-world ML (and DL) problems are non-convex optimization problems. In such cases, the most common approaches involve iterative searches that compute better candidates for the parameters of a model step by step.

One of the most well-established optimization techniques is *gradient descent*, which iteratively updates the parameters of a cost function $C(\theta)$ as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)}), \quad (1.18)$$

where η is a hyperparameter called the *learning rate* and t is the iteration index. Since the gradient, $\nabla C(\theta^{(t)})$, points towards increasing the cost function, its opposite leads us towards lower values of the function. Note that Eq. 1.18 is in general form, in practical implementation both the cost function and its gradient depend on the entire training dataset as in Eq. 1.14. The learning rate determines the step size at which the model parameters are updated during each iteration of the optimization process. If the learning rate is set too high, the algorithm may overshoot the minima of the cost function, leading to divergence or instability. Conversely, if it is too low, the optimization may require a large number of iterations to converge and can get stuck in local minima. In Figure 1.3 an example of the dynamics of gradient descent is illustrated.

Another widely adopted optimization technique is *Stochastic Gradient Descent* (SGD), which estimates the gradient at each iteration using only a subset of the training data. Traditionally, SGD relied on a single sampled training example per update. However, modern implementations typically employ *mini-batches* that are small selected subsets of the data, to compute the gradient more efficiently. The

stochastic nature of SGD introduces randomness into the optimization process, which can help the algorithm escape local minima [47, 48]. However, this randomness can also lead to fluctuations in the training procedure, especially when small batch sizes are used.

There are different variants of (SGD), each aimed at improving convergence speed and training stability. For example, these variants can involve dynamically adjusting the learning rate during training (an approach used in optimizers such as Adagrad, or incorporate momentum, in which parameter updates at iteration t depend on updates to the previous iteration $t - 1$ (strategy used in optimizers such as Momentum and Adam).

1.4 Methods for ML and DL problems

In ML and DL there is a wide range of models and methods, each with specific characteristics and applications. Next, we will introduce only the models and methods essential for understanding the topics covered in this manuscript. For a more complete overview, see [37, 8].

1.4.1 Linear models

Linear models represent a class of deterministic models representing a fundamental tool for the conceptual understanding and construction of more advanced models, such as artificial neural networks and support vector machines. In supervised problems, they are typically used for classification and regression tasks.

Mathematically, a linear model is formulated by the inner product between the input data vector $\mathbf{x} \in \mathbb{R}^n$ and a parameter vector $\mathbf{w} \in \mathbb{R}^n$ and is expressed by the function:

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}. \quad (1.19)$$

The elements w_i of the parameter vector are often called the *weights* of the model. In some cases, a scalar term $b \in \mathbb{R}$, called *bias*, is added. However, this can be incorporated directly into the inner product by expanding $\mathbf{w}$ and $\mathbf{x}$ with an additional dimension such that $w_0 = b$ and $x_0 = 1$.

An important aspect of linear models is that, when combined with a convex loss function, such as the mean squared error, they give rise to convex optimization problems [28]. These problems have a unique global minimum and are well understood by optimization theory.

1.4.2 Artificial neural networks

Artificial Neural Networks (ANNs) are computational models originally inspired by studies of the mechanisms of learning and information processing in the human brain. The basic processing units in the brain are electrically active cells known as neurons. When activated, a neuron generates an electrical impulse that travels along its axon to specialized junctions called synapses. Synapses form connections with other neurons by releasing chemical messengers known as neurotransmitters. These neurotransmitters modulate the activity of adjacent neurons, promoting or

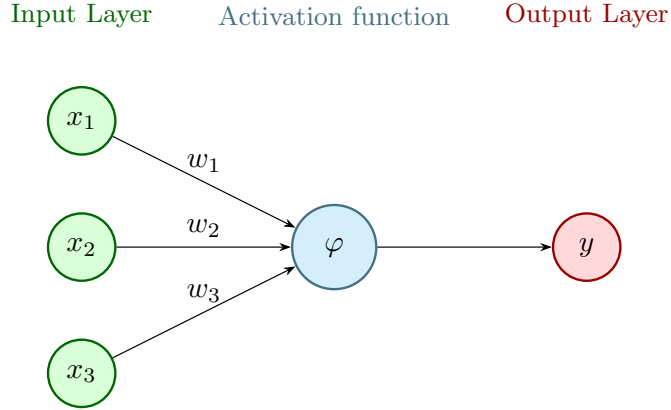


Figure 1.4. Illustration of the perceptron model. The inputs x_i are each displayed as nodes of a graph, called *neurons* or *units*. The input units are connected to an output unit by weighed edges w_i . An activation function φ is applied to weighted input, $\sum_{i=1}^3 w_i x_i$, which returns an output y . The bias term is here omitted for simplicity.

inhibiting their firing, thus shaping the dynamic flow of information through neural networks. In the history of neural computation, one of the most influential models is the *perceptron* introduced by Warren S. McCulloch and Walter Pitts in 1943 [3] and simulated by Frank Rosenblatt in 1957 [5, 6]. This model mathematically captures the fundamental dynamics of a single biological neuron and represents the first example of ANN capable of learning from data through supervised training. Although limited to linearly separable problems, the perceptron laid the foundation for modern neural network models.

Perceptron

The perceptron is the simplest ANN architecture. It consists of a single layer and is formally described by a function:

$$f_{\mathbf{w}}(\mathbf{x}) = \varphi(\mathbf{w}^T \mathbf{x}), \quad (1.20)$$

where the term $\mathbf{w}^T \mathbf{x}$ is defined as in Eq. 1.19 (with the bias term integrated in the weights) and φ is called the *activation function* that acts as follows:

$$\varphi(a) = \begin{cases} 0, & \text{if } a < 0, \\ 1, & \text{if } a \geq 0. \end{cases} \quad (1.21)$$

Thus, the perceptron is a linear model that produces binary outputs. The nonlinear function φ was originally inspired by the *integrate-and-fire* dynamics observed in biological neurons. According to this mechanism, the perceptron emits a signal when the weighted input given by the inner product $\mathbf{w}^T \mathbf{x}$ exceeds a certain threshold. A graphical representation of the perceptron is illustrated in Figure 1.4.

The perceptron model was developed with a specific training algorithm that updates the weights iterating through the data as follows:

$$w_i^{(t+1)} = w_i^{(t)} + \eta \left(y^m - \varphi(\mathbf{x}^T \mathbf{w}^{(t)}) \right) x_i^m. \quad (1.22)$$

With this learning rule, the perceptron can achieve a perfect classification task, converging to the optimal weights in a finite number of steps [37, 6]. However, this only holds for linearly separable datasets, that is, when there exists a line (or hyperplane) that perfectly divides the data into two classes in the input space [49].

To overcome this limitation, more complex models have been developed that introduce *hidden layers*, allowing the learning of even complex relationships between input data.

Feed-forward neural networks

Modern ANNs can be interpreted as sets of hierarchically organized perceptrons where the output of each perceptron serves as input for the subsequent ones. This structure is also commonly referred to as Multilayer Perceptron (MLP), and it forms the conceptual foundation of many DL systems. One of the most important architectures is that of so-called *feed-forward* networks. In this configuration, the perceptrons are organized in successive acyclic *layers*, so that the information flows only forward from the input to the output. Each layer receives data from the previous one, processes it using an activation function, and transmits it to the next.

In its simplest form, a feed-forward network is described by a deterministic model composed of a sequence of linear and nonlinear transformations. The input $\mathbf{x} \in \mathbb{R}^n$ is multiplied by a weight matrix $\mathbf{W}_l \in \mathbb{R}^{J_l \times J_{l-1}}$ for each layer $l = 1, \dots, L$, and the result is transformed by an activation function φ_l , applied element-wise. These functions φ_l are vectorized versions of the perceptron activation function, mapping the result of a matrix-vector multiplication to a vector of the same size, with each element transformed by the same non-linear function. Formally, ANN with a feed-forward structure can be expressed as a function of the form:

$$f_{\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_L}(\mathbf{x}) = \varphi_L(\mathbf{W}_L \cdots \varphi_2(\mathbf{W}_2 \varphi_1(\mathbf{W}_1 \mathbf{x})) \cdots). \quad (1.23)$$

After L layers, the final activation function, φ_L , maps the output to the output space $\mathcal{Y}$. One compelling aspect of these models is that they combine the flexibility of non-linear dynamics with the computational power of linear algebra. Graphically, a model with a feed-forward dynamics consists of:

- A single **input layer** with input features $x_1, \dots, x_i$.
- L **hidden layers** with units $h_1^l, \dots, h_j^l$ for $l = 1, \dots, L$
- A single **output layer** with units $y_1, \dots, y_k$.

An example of ANN with a feed-forward structure is illustrated in Figure 1.5. A foundational result known as the *universal approximation theorem*, proved by Hornik, Stinchcombe, and White, shows that a feed-forward network with at least one hidden layer and a non-linear activation function can approximate any continuous function on a compact domain with arbitrary precision [50].

Various differentiable activation functions can be used in the hidden layers depending on the problem task (for a more complete overview, see [51]). Here, we only define the necessary for this manuscript:

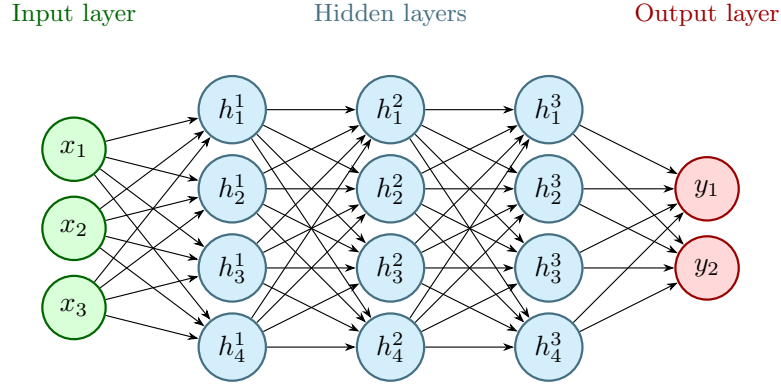


Figure 1.5. Architecture of an ANN. The network consists of interconnected nodes organized into layers: an input layer with three inputs, three hidden layers with four units, and an output layer with two units. Each node processes incoming signals using activation functions and transmits the result to subsequent layers.

- **Sigmoid:** a nonlinear function mapping input values to the range between 0 and 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

It is commonly adopted to model the probability of a binary output.

- **Rectified Linear Unit (ReLU):** a simple and computationally efficient function that returns zero for negative inputs and the input itself for positive values:

$$\text{ReLU}(x) = \max(0, x)$$

- **Softmax:** converts a vector of K real-valued inputs into a K -discrete probability distribution:

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}},$$

with $i = 1, \dots, K$. It is often used in the output layer for multi-class classification tasks.

When using non-linear and differentiable activation functions, the resulting optimization problem becomes non-convex, and therefore computationally demanding. In training, parameter optimization is made possible by efficient implementations of the *backpropagation* algorithm [7]. The model parameters are typically initialized randomly and then optimized using gradient-based techniques. More precisely, the process involves computing the derivatives of the loss function with respect to the single weights, efficiently performed through error backpropagation and the *chain rule* of the derivation, which enables the error signal to flow backward across the network layers and allows effective learning even in deep architectures.

Name	Kernel	Hyperparameters
Linear	$\mathbf{x}^T \mathbf{x}'$	–
Polynomial	$(\mathbf{x}^T \mathbf{x}' + c)^p$	$p \in \mathbb{N}, c \in \mathbb{R}$
Gaussian / RBF	$e^{-\gamma \ \mathbf{x} - \mathbf{x}'\ ^2}$	$\gamma \in \mathbb{R}^+$
Exponential	$e^{-\gamma \ \mathbf{x} - \mathbf{x}'\ }$	$\gamma \in \mathbb{R}^+$
Sigmoid	$\tanh(\mathbf{x}^T \mathbf{x}' + c)$	$c \in \mathbb{R}$

Table 1.1. Example of most common classical kernel functions as distance measures between a pair of data points $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$

1.4.3 Kernel methods

Kernel methods solve machine learning tasks based on the idea of a measure of similarity between data points. For example, the similarity measure can be used to compare a new data point with training samples in a classification task. This similarity measure is expressed by a kernel function, which is defined as follows:

Definition 6. Given a data domain $\mathcal{X}$, a kernel is a positive semi-definite bivariate function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{C}$. Positive semi-definiteness means that for any set $\mathcal{D} = \{x^1, x^2, \dots, x^M\} \subset \mathcal{X}$, the Gram matrix K with entries

$$K_{m,m'} = \kappa(x^m, x^{m'}), \quad x^m, x^{m'} \in \mathcal{D} \quad (1.24)$$

is positive semi-definite. Consequently, $\kappa(x^m, x^{m'}) \geq 0$ and $\kappa(x^m, x^{m'}) = \kappa(x^{m'}, x^m)^*$.

In the definition, the codomain is taken in its most general form to be the space of complex numbers. However, in practical scenarios, it is usually restricted to the space of real numbers. Examples of the commonly used classical kernel functions are provided in Table 1.1.

An important characteristic of kernel functions is that they can be written as an inner product of data mapped to a suitable *feature space* $\mathcal{F}$ by a *feature map*:

$$\phi : \mathcal{X} \rightarrow \mathcal{F}, \quad (1.25)$$

such that

$$\kappa(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}. \quad (1.26)$$

The feature map allows us to transform a non-linearly separable dataset into a linearly separable form, enabling the use of linear models, as shown in Figure 1.6. Furthermore, the representation as an inner product in the feature space enables one to operate in a high-dimensional feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between all pairs of data in the feature space. This operation is often computationally less intensive than the explicit computation of the coordinates and is known as the *kernel trick* [52]. Another advantage of expressing a model in terms of a kernel function is that it allows us to turn one model into another by simply replacing a kernel κ with another kernel κ' .

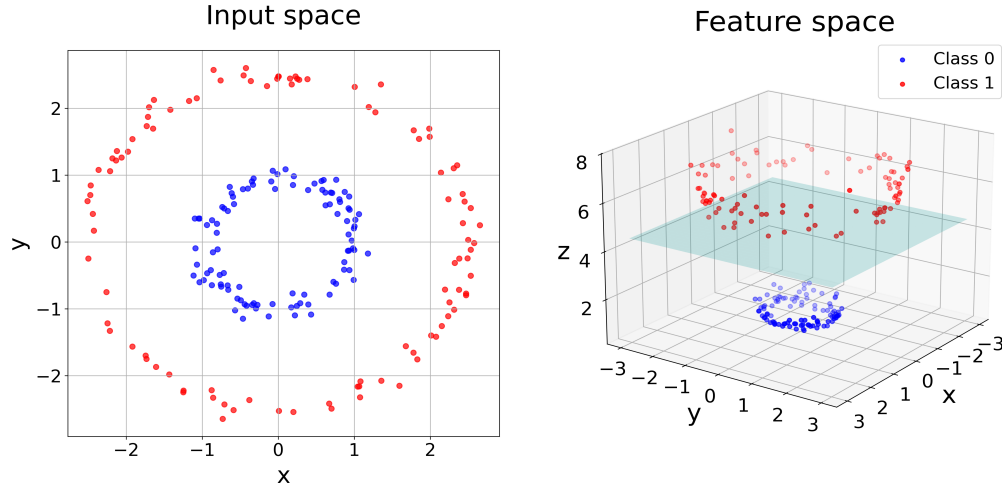


Figure 1.6. A non-linearly separable dataset in the two-dimensional input space $\mathcal{X}$ (on the left) is made linearly separable in the three-dimensional feature space $\mathcal{F}$ (on the right) with the feature map $\varphi(x, y) = (x, y, x^2 + y^2)$, which adds one dimension to the input data.

Support Vector Machines

Support Vector Machines (SVMs) are one of the most well-known kernel methods, typically used for classification and regression tasks. The modern formulation was introduced in the 1990s by Vladimir Vapnik and Corinna Cortes [53]. SVMs aim to solve classification problems based on a simple geometric principle of finding the optimal hyperplane (also called a *decision boundary*) that separates data belonging to different classes. Once trained, to classify new data points, it is sufficient to verify which area of the decision boundary they fall within.

For a more formal prospective, consider a binary classification problem with input domain $\mathcal{X} = \mathbb{R}^n$ and output domain $\mathcal{Y} = \{-1, 1\}$, and a linear model:

$$f_{\theta}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b, \quad (1.27)$$

with $\theta = \{\mathbf{w}, b\}$ the set of model parameters, where $b \in \mathbb{R}$ is the bias test. Assume linearly separable data, that is, there exists a set of parameters θ that defines a hyperplane separating the training data such that the points of the two classes lie on opposite sides. Mathematically, linear separation can be expressed by the following condition:

$$\exists \theta \text{ s.t. } \forall m \in \{1, \dots, M\}, f_{\theta}(\mathbf{x}^m) y^m > 0, \quad (1.28)$$

where $\mathbf{x}^m$ and y^m are, respectively, the m -th training data point and the corresponding class label. The goal is to determine the decision boundary that maximizes the *margin*, defined as the distance between the separating hyperplane and the closest training points $\mathbf{x}^s$. These points are known as *support vectors*.

The margin can be geometrically derived by decomposing one of the support vectors $\mathbf{x}^s$ in its orthogonal projection $\mathbf{x}^{\text{proj}}$ onto the separating hyperplane and a component of length Δ (i.e., the margin) orthogonal to the hyperplane, as shown in Figure 1.7. Since the orthogonal vector is for construction parallel to the weight

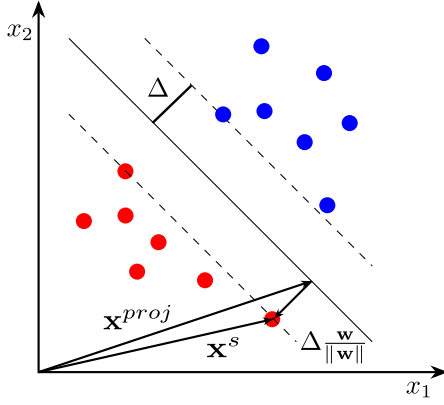


Figure 1.7. Illustration of the geometric construction of the expression for the distance Δ of the margin.

vector $\mathbf{w}$, we can write:

$$\mathbf{x}^s = \mathbf{x}^{\text{proj}} + \Delta \frac{\mathbf{w}}{\|\mathbf{w}\|}. \quad (1.29)$$

where $\|\cdot\|$ is the Euclidean norm. Calculating the output of the model for the support vector, with a proper shift of $\mathbf{w}$ and b , it reads:

$$f_{\theta}(\mathbf{x}^s) = \mathbf{w}^T \mathbf{x}^{\text{proj}} + \Delta \frac{\mathbf{w}^2}{\|\mathbf{w}\|}. \quad (1.30)$$

Given the orthogonality $\mathbf{w} \perp \mathbf{x}^{\text{proj}}$, the inner product $\mathbf{w}^T \mathbf{x}^{\text{proj}}$ is zero and since $\mathbf{w}^T \mathbf{w} = \|\mathbf{w}\|^2$, we obtain:

$$\Delta = \frac{f_{\theta}(\mathbf{x}^s)}{\|\mathbf{w}\|}. \quad (1.31)$$

The support vector machine defines the decision boundary so that the margin is maximized while ensuring condition 1.28. The weight vector $\mathbf{w}$ and the bias term b can be rescaled without affecting the position of the margin. Since this scaling freedom leads to infinitely many equivalent hyperplanes, we fix the scale by setting $\Delta \|\mathbf{w}\| = 1$. Given this condition, it can be demonstrated [54] that a way to find the decision boundary is by solving the problem:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y^m (\mathbf{w}^T \mathbf{x}^m + b) \geq 1. \quad (1.32)$$

This constrained quadratic optimization problem can be turned into an unconstrained problem by using Lagrange multipliers $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)$ and minimizing the Lagrange function:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{m=1}^M \alpha_m \left[y^m (\mathbf{w}^T \mathbf{x}^m + b) - 1 \right]. \quad (1.33)$$

Although this is a n -dimensional convex optimization problem, it can be intractable for high-dimensional feature spaces. In contrast, its *dual* form does not depend on $\mathbf{w}$ or N , which makes it computationally more efficient. The dual Lagrangian function is given by:

$$\mathcal{L}_{\text{dual}}(\boldsymbol{\alpha}) = \sum_{m=1}^M \alpha_m - \frac{1}{2} \sum_{m, m'=1}^M \alpha_m \alpha_{m'} y^m y^{m'} (\mathbf{x}^m)^T \mathbf{x}^{m'}, \quad (1.34)$$

which is minimized with respect to the Lagrange multipliers α_m , subject to constraints $\alpha_m \geq 0 \forall m$ and $\sum_{m=1}^M \alpha_m y^m = 0$. Eq. 1.34 defines a convex optimization problem in a M -dimensional space, where M is the number of training samples. Once the Lagrangian multipliers are found, the weights are determined by means of relation:

$$\mathbf{w} = \sum_{m=1}^M \alpha_m y^m \mathbf{x}^m, \quad (1.35)$$

and the predictions can be computed with the linear model in Eq. 1.27.

Finally, we observe that the dual function in Eq. 1.34 is expressed in terms of inner products $\mathbf{x}^T \mathbf{x}'$ between two data samples $\mathbf{x}$ and $\mathbf{x}'$, which is the linear kernel reported in Table 1.1. Therefore, the form of Eq. 1.34 enables the use of the kernel trick, allowing SVMs to construct nonlinear decision boundaries by implicitly mapping data to a higher-dimensional feature space. Crucially, this transformation preserves the convexity of the optimization problem and maintains its dependence on the number of training samples.

Chapter 2

Generative AI and Diffusion Models

Generative artificial intelligence is a subfield of artificial intelligence that uses the flexibility of deep neural networks to produce new data. Here, we provide the mathematical framework that describes deep generative modeling. This chapter is structured as follows. In Section 2.1, we begin by presenting a historical overview of the major milestones in the development of approaches and platforms for generative AI. Next, Section 2.2 discusses the role and importance of *synthetic data* generation within the deep learning paradigm. In Section 2.3, we formally describe deep generative modeling. We then describe the role of latent space as an essential element in data generation within the most widely used approaches, namely *generative adversarial networks*, *variational autoencoders*, and *diffusion models*. The chapter concludes with Section 2.4, which provides a detailed description of the generative diffusion models that are the starting point of this work. Indeed, this thesis is based on, to the best of our knowledge, the first work that generalizes diffusion models to the quantum domain [29]. This extension will be explored in detail in Chapter 7, where we introduce the novel paradigm of quantum diffusion models.

2.1 Historical overview

Generative Artificial Intelligence (GenAI) represents one of the most fascinating technologies of our century that enables machines not only to classify, recognize patterns, or cluster data, but also to generate content by emulating human abilities and creativity. GenAI systems rely on sophisticated deep learning models designed to identify and learn the underlying relationships from a finite set of samples to create new original and meaningful content such as images, text, video, or other forms of data.

Although the term “generative AI” exploded into public and academic spheres in the 2020s, the origins of GenAI can be traced back to the mid-20th century. Between 1964 and 1967, Joseph Weizenbaum, computer scientist at what was then the MIT Artificial Intelligence Laboratory, developed ELIZA, a pioneering text-based natural language processing program designed to simulate human-machine interaction by mimicking the conversational style of a psychotherapist. Considered the first chatbot

in history, ELIZA operated using a pattern matching and substitution methodology that gave the user the illusion of a meaningful dialogue [55].

In the mid-2010s, the emergence of generative models such as Generative Adversarial Networks (GANs) [9], Variational Autoencoders (VAEs) [10], and Diffusion Models (DMs) [11] marked a significant breakthrough in the landscape of deep learning. For the first time, these models exploited the flexibility of deep ANNs not only to classify the visual content of an image but also to reproduce it, paving the way for the generation of realistic images. Subsequently, the introduction of the transformer architecture in 2017 marked a paradigm shift in the field of generative modeling [12]. Unlike previous ANN architectures that relied on recurrence or convolution, transformers leveraged self-attention mechanisms proposed by Bahdanau [56] to capture long-range dependencies and contextual relationships within data. In 2018, based on the transformer framework, OpenAI [57] launched the first pre-trained generative model known as GPT-1 [13]. Later in 2022, the public release of ChatGPT [58], a chatbot created by OpenAI, revolutionized the accessibility and application of generative AI for general-purpose text-based tasks, such as engaging in natural conversations.

Since the early 2020s, the rapid advancement and widespread adoption of GenAI systems by a growing number of users have contributed to what is commonly known as the *AI boom*. This has been driven by significant improvements in transformer-based deep ANNs and the emergence of the so-called Denoising Diffusion Probabilistic Models [14]. In particular, diffusion-based generative models have achieved state-of-the-art performance, outperforming previous models such as VAEs and GANs in image and audio synthesis [15, 16]. This was followed by the release of several platforms based on the integration of DMs and transformers, such as DALL·E [17], a text-image model developed by OpenAI, Stable Diffusion [18] by Stability AI [59] and Midjourney [19] that marked a significant advance in image generation, demonstrating unprecedented capabilities in the generation of 3D images, artwork, and photorealistic drawings based on text descriptions. Recently, Google DeepMind [20] announced the releases of Gemini Diffusion [21], a diffusion-based Large Language Model.

Today, GenAI systems are widely adopted in academia and industry, and they are redefining how we learn, create, and make decisions with a profound impact on both social and economic dynamics. In finance, for example, they can process and synthesize large volumes of data to identify patterns and anomalies, supporting fraud prevention and risk management [60]. In healthcare, generative models are being applied to drug discovery and healthcare [61, 62]. In the entertainment industry, they are used to generate original content such as music and video [63]. However, easy access to these technologies to generate content has raised numerous ethical questions and governance challenges, as they can be used to commit cybercrimes by deceiving and manipulating people through fake news [64]. This can pose risks to privacy, misinformation, and intellectual property. Consequently, researchers and political leaders are actively developing regulatory and technical frameworks to ensure responsible use [65]. Another challenge concerns biases in generative models. Because these models learn from existing data, they can reproduce and amplify social biases. Addressing these issues requires diverse training datasets, transparent model design, and ongoing evaluation processes.

Despite these challenges, the integration of GenAI into daily life continues to

increase and represents a technological revolution in human-machine interaction, with profound implications for the future of society.

2.2 The importance of synthetic data

In the landscape of deep learning, data samples serve as a critical foundation for the construction of accurate predictive models. However, real-world data sets are often incomplete, lacking labels, or inaccessible due to privacy regulations [66]. These limitations are particularly pronounced in sensitive domains such as healthcare, finance, and public services, where legal and ethical concerns restrict the sharing of personal information. As a result, the availability of high-quality data for training robust models remains a persistent challenge.

To address these constraints, *synthetic data* has emerged as a promising alternative [67, 68, 69]. Unlike data obtained through direct observation or experimentation, synthetic data is artificially generated using mathematical models and advanced machine learning techniques. Its objective is to replicate the statistical properties and structural relationships of real datasets while offering enhanced flexibility and privacy protection. Beyond mere replication, synthetic data can simulate hypothetical scenarios or novel combinations of variables, enabling applications such as visual recognition training, molecular design, and economic modeling.

The adoption of synthetic data brings several advantages to data-driven research and development. When produced using robust methodologies, it can mitigate the biases inherent in real-world datasets, allowing models to be evaluated under more controlled and neutral conditions. It also offers a cost-effective solution, particularly in resource-intensive applications such as medical imaging, eliminating the need for expensive equipment and manual collection processes. Moreover, synthetic data inherently supports privacy preservation, as it is not linked to identifiable individuals, thereby facilitating data sharing without compromising confidentiality. Another key benefit is scalability. Once a synthetic data generator is established, it can produce datasets of virtually any size, tailored to specific analytical or computational requirements [70]. This capability enables experimentation and model development without the data labeling costs and risks associated with handling sensitive real-world data.

However, the effectiveness of synthetic data is closely linked to the quality of the underlying generation process. If the models used do not adequately reflect the relevant characteristics of the intended domain, the resulting data may lead to inconsistencies or unintended distortions. This highlights the importance of employing suitable generation methods and applying careful validation procedures to ensure that synthetic datasets are both reliable and fit for purpose.

In conclusion, the use of synthetic data represents a powerful tool for overcoming the limitations of real data in machine learning and deep learning. Their potential to improve privacy, reduce costs, and support scalable experimentation makes them an increasingly crucial resource. However, their effectiveness depends on careful design and validation, which are essential to ensure their suitability for specific application contexts [71].

2.3 Deep generative modeling

Deep generative modeling is an approach that combines probabilistic modeling with the representational power of deep ANNs to learn complex data distributions. In the following, we first define the distinction between probabilistic discriminative and generative models. Then, we introduce the concept of latent space, a fundamental and crucial component in architectures such as VAEs, GANs, and DMs.

2.3.1 Discriminative vs generative models

Probabilistic models (see Definition 4) are divided into two main classes: *discriminative* and *generative*. This distinction depends on the distribution the model is designed to learn and is influenced by the application objectives and training techniques.

Discriminative models are probabilistic models trained to estimate the conditional distribution $p_\theta(y|x)$, that is, the probability that, given a feature vector $x \in \mathcal{X}$, the observation belongs to class $y \in \mathcal{Y}$. In other words, the main objective of discriminative models is to infer the correct label for a feature vector x . This approach is typical of supervised learning, in which data samples are labeled. The most paradigmatic example is the use of ANNs for classification tasks.

On the other hand, generative models aim to learn an approximate form of the distribution of the data itself, i.e., $p_\theta(x) \approx p_{\text{data}}(x)$. This allows them to generate new samples that resemble those in the training dataset. This approach is typical of unsupervised learning, in which data samples are not labeled. There are two main ways [43] of modeling $p_\theta(x)$: either by evaluating the likelihood function of the observations or by synthesizing realistic samples that reflect the statistical properties of the training data set. The former are called *explicit generative models*, and examples include VAEs and DMs. The latter are called *implicit generative models*, and an example is the class of models known as GANs.

A subclass is represented by conditional generative models. Unlike generative models that learn $p_\theta(x)$, conditional models learn to generate data guided by external information. In mathematical terms, they are trained to learn the distribution $p_\theta(x|y)$, generating a data sample x consistent with a conditioning variable y (for example, a class or a textual description). They allow direct control over generation, such as image synthesis from text prompts. For example, text-to-image platforms such as Midjourney and Stable Diffusion rely on conditional probabilistic models of the form $p(\text{image}|\text{text})$.

The main work of this thesis relies on deep generative models, which are generative models that leverage the flexibility of deep ANNs combined with the concept of latent space to learn complex high-dimensional distributions to generate novel data.

2.3.2 Latent space in data generation

In machine learning, the latent space is a lower-dimensional representation in which a model captures the underlying structure of data samples in a compact and tractable form. In deep generative models, the latent space plays a crucial role not only in representation learning but also as a sampling domain for data generation.

Specifically, these models aim to approximate the unknown data distribution $p_{\text{data}}(x)$ by transforming a simple and tractable distribution $p(z)$, defined over the latent space. In the context of GenAI, the data distribution is often called *posterior* and the latent distribution is called *prior*. Typically, the prior $p(z)$ is chosen as a standard Gaussian or a uniform distribution. More precisely, by sampling latent variables $z \sim p(z)$, the generative model applies a learned transformation to produce new instances x' that mimic the statistical characteristics of the real data distribution $p_{\text{data}}(x)$, making the model effective in synthesizing new data. This principle underlies prominent generative architectures such as VAEs, GANs, and DMs. This principle is realized for each architecture in a distinct way:

- **Variational Autoencoders:** learn an explicit mapping between data and latent variables through an encoder-decoder framework, maximizing a variational lower bound. More precisely, a first neural network, known as the *encoder*, maps the input data to a smaller latent representation, typically modeled as a Gaussian distribution. The latent variables z are then sampled from this distribution and passed to a second network, the *decoder*, which reconstructs the original data from the latent space. Once trained, the decoder can generate new data samples by receiving latent variables as input.
- **Generative Adversarial Networks:** employ a generator-discriminator setup. In particular, a latent vector z is typically sampled from a uniform or Gaussian distribution. This vector is then passed through a neural network called the *generator*, which is trained to produce realistic samples. Training is implemented through an adversarial process, involving a second network known as a *discriminator*, which is trained to distinguish between real and generated data.
- **Diffusion Models:** they are inspired by the thermodynamic diffusion process and consist of two phases. In a forward diffusion process, data samples are gradually corrupted by Gaussian noise over time, transforming them into increasingly noisy latent representations. Next, in a reverse denoising process, a neural network learns to reconstruct the data by progressively removing the noise. After training, the model generates new samples starting from random latent input and reversing the diffusion dynamics.

The different architectures described above are illustrated in Figure 2.1. Despite their differences, all of these models rely on the concept of latent space as a generative substrate, enabling the synthesis of high-dimensional data from low-dimensional and structured representations.

Furthermore, the latent space plays a key role as a source of *stochasticity*, which is essential to introduce diversity into the generated samples. Once models are trained, sampling different latent variables from the prior distribution leads to distinct output, enabling the generation of diverse and novel data instances. In conclusion, latent space is a fundamental component in generative modeling, serving both as a mechanism to make learning tractable and as a source of statistical variability in the generation process.

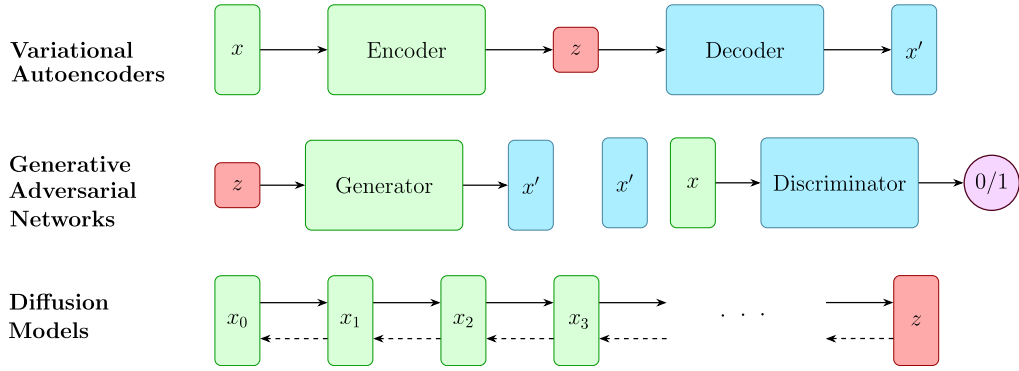


Figure 2.1. Representation of three deep generative model architectures, arranged vertically from top to bottom: Variational Autoencoder, Generative Adversarial Network, and Diffusion Model. The core components and the corresponding generative mechanism are depicted in each diagram, providing a comparative overview of their structural differences.

2.4 Diffusion models

Recently, DMs have emerged as the state-of-the-art in the landscape of deep generative modeling. In particular, they have shown superior performance in image and audio synthesis, outperforming earlier approaches such as VAEs and GANs in terms of fidelity and diversity [15, 16].

Originally introduced by Sohl-Dickstein et al. [11], DMs are deep generative models inspired by the physical phenomenon of *diffusion*. In physical diffusion dynamics, a set of particles gradually lose information about their initial state under the influence of stochastic processes. Analogously, data samples are iteratively perturbed by adding increasing amounts of Gaussian noise until they become pure, structureless noise. Sohl-Dickstein et al. proposed a generative model based on the idea of learning to reverse this process: starting from noise, the model is trained to progressively remove noise and reconstruct the initial samples. This approach allows for the modeling of complex probability distributions while maintaining computational tractability for both training and sampling.

The general pipeline of DMs consists of two processes formally described by two Markov chains¹. The forward process begins with a data point x_0 sampled from an unknown complex data distribution $q(x_0)$, and progressively corrupts it through successive additions of Gaussian noise until it produces x_T , which represents pure noise. Intermediate variables $x_1, \dots, x_{T-1}$ represent increasingly noisy versions of the data, generated by the forward diffusion process through successive additions of Gaussian noise. This transformation is governed by a fixed transition distribution $q(x_t|x_{t-1})$, which contains no learnable parameters and simply defines the amount of noise introduced at each time step. From the perspective of the model, these variables serve as latent representations used during training to learn the inverse denoising process. The reverse process is represented by the distribution $q(x_{t-1}|x_t)$, which corresponds to the removal of noise starting from x_T . However, this distribution

¹A Markov chain is a stochastic process $\{X_n\}_{n \in \mathbb{N}}$ with a state space S , where X_n denotes the state at time n . It satisfies the Markov property (*memoryless*): $P(X_{n+1} = x_{n+1} | X_n, X_{n-1}, \dots, X_0) = P(X_{n+1} = x_{n+1} | X_n)$, namely the distribution of the next state depends only on the current state.

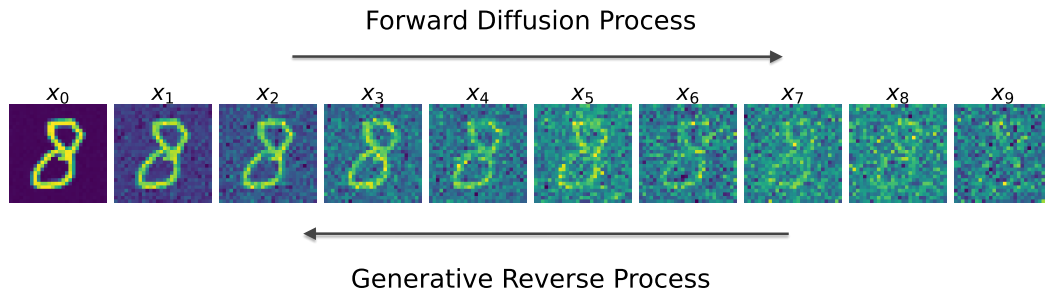


Figure 2.2. Starting from a image x_0 sampled from MNIST dataset [30], the forward diffusion process progressively corrupts the data from left to right, eventually producing pure noise sample x_9 after $T = 9$ time steps. The reverse trajectory from right to left describes the generative process in which noise is removed to generate a realistic sample.

is generally intractable and cannot be computed in a closed form. To address this problem, an auxiliary distribution $p_\theta(x_{t-1}|x_t)$ is adopted, which is implemented by a neural network with learnable parameters θ . This network approximates the dynamics backward and allows for gradual reconstruction of the data from the noise. Figure 2.2 illustrates an example of forward diffusion and reverse denoising trajectories.

A distinctive feature of DMs is that all latent variables share the same dimensionality as the original data. Unlike VAEs, these models do not impose a bottleneck that compresses information into a lower-dimensional latent space. As a result, the notion of latent differs significantly: rather than forming a compact, explicitly defined subset of the architecture, in DMs the latent space emerges implicitly through the stochastic dynamics of the forward and reverse processes.

Diffusion-based generative models can be formulated in various ways depending on the type of data (continuous or discrete values) and the nature of the time intervals (discrete or continuous). In the following, we begin with the discrete-time formulation for continuous data [14], detailing the forward and reverse processes, as well as the training procedure and the associated loss function. We then move on to the continuous-time formulation, based on stochastic differential equations [72]. Finally, we present the discrete-time diffusion model designed for categorical variables [11, 73, 74].

2.4.1 Denoising diffusion probabilistic models

The Denoising Diffusion Probabilistic Models (DDPMs) were introduced in 2020 by Ho et al. [14], based on the previous work by Sohl-Dickstein et al. [11]. Due to its ability to generate samples with high fidelity and diversity through an efficient training process, the DDPM framework has become a foundational reference within the family of diffusion-based models, particularly in discrete-time formulations. Here, it is adopted as a methodological basis for introducing and analyzing the fundamental mechanisms underlying diffusion-based generative models. The following sections provide an in-depth discussion of the forward diffusion process, the inverse denoising trajectory, and the objective function in the training procedure.

Forward Process

The forward process is formally described as a Markov chain over a finite number of discrete time steps T . It begins with an initial data sample $x_0 \sim q(x_0)$, which is progressively corrupted by Gaussian noise until it becomes indistinguishable from pure noise at the final time step x_T . Here, $q(x_0)$ is the complex and unknown underlying distribution that describes the generative process of the observed data. In forward dynamics, each intermediate latent variable x_t is a mixture of the initial data sample and noise. This expresses the gradual degradation of structured data into pure noise. At each time step t , a scheduled hyperparameter $\beta_t \in (0, 1)$ controls the amount of noise added to the previous latent state.

Formally, the transition from x_{t-1} to x_t is described by the Gaussian transition distribution:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I), \quad (2.1)$$

where $\mathcal{N}(x; \mu, \sigma)$ is the Gaussian distribution of the stochastic variable x with mean μ and standard deviation σ , and I denotes the identity matrix. The latent variable x_t is sampled by Eq. 2.1 or can be equivalently computed as follows:

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon, \quad (2.2)$$

where $\epsilon \sim \mathcal{N}(0, I)$ corresponds to the noise used to corrupt the data. Due to the Markov property, the joint distribution over the entire trajectory conditioned on the initial data point can be written as follows:

$$q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}). \quad (2.3)$$

As all forward transition kernels are Gaussian, it is useful to reparameterize the process by defining $\alpha_t = 1 - \beta_t$, which yields an equivalent but more tractable formulation. This reparameterization simplifies the training procedure and enables efficient sampling, as it allows one to obtain the latent variable x_t directly from the original data x_0 and a standard Gaussian noise variable $\epsilon \sim \mathcal{N}(0, I)$. Specifically, the closed-form expression is given by:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad (2.4)$$

where $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ denotes the cumulative product of the noise coefficients. In addition, the marginal distribution of x_t conditioned on x_0 is also Gaussian:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)I). \quad (2.5)$$

As the number of time steps T increases, the cumulative product $\bar{\alpha}_T$ tends to zero: $\lim_{T \rightarrow \infty} \bar{\alpha}_T = 0$. Therefore, for sufficiently large T , the final latent variable x_T becomes independent of the conditioning variable x_0 :

$$\begin{aligned} q(x_T|x_0) &\approx \mathcal{N}(x_T; \sqrt{0} x_0, (1 - 0)I) \\ &= \mathcal{N}(x_T; 0, I) \\ &= q(x_T). \end{aligned} \quad (2.6)$$

It demonstrates that the forward process ultimately transforms structured data into pure Gaussian noise. From another perspective, this means that the initial unknown data distribution $q(x_0)$ is transformed into a well-defined, tractable, and easily sampled distribution, in this case an isotropic Gaussian, as discussed in Section 2.3.2.

Finally, it is important to emphasize that the forward process does not rely on the use of neural networks and does not involve any learnable parameters.

Reverse Process

The reverse process constitutes the generative phase of diffusion models. As discussed previously, the forward process progressively corrupts a data sample x_0 by adding Gaussian noise at each step, until the final distribution $q(x_T)$ converges to a standard Gaussian $\mathcal{N}(0, I)$. This property is crucial because it guaranties that the final state of the forward chain is analytically tractable and easy to sample from. The objective of the reverse process is to invert this corruption, reconstructing structured data starting from noisy samples.

Formally, this requires modeling the conditional distribution $q(x_{t-1}|x_t)$, which describes the probability of obtaining a less noisy representation x_{t-1} given a corrupted sample x_t . However, this distribution is analytically intractable. To address this limitation, it is approximated by a parameterized distribution $p_\theta(x_{t-1}|x_t)$ that is implemented through a neural network. The generative process begins with the following prior distribution:

$$p(x_T) = \mathcal{N}(0, I), \quad (2.7)$$

which coincides with the limiting distribution of the forward process. Next, starting from a sample $x_T \sim p(x_T)$, the model progressively reconstructs the data by iteratively applying the learned reverse transitions $p_\theta(x_{t-1}|x_t)$, thus denoising the latent variables x_t step by step until the sample x_0 is obtained. The reverse process can be expressed as the joint distribution given by:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t), \quad (2.8)$$

which defines a Markov chain that evolves from x_T to x_0 . This formulation highlights that the Markov property remains valid in the reverse direction, allowing the generative model to be represented as a sequence of conditional transitions that collectively implement the denoising process.

The training objective

The training objective is derived from a variational principle similar to the Evidence Lower Bound (ELBO) for VAEs, but adapted to the specific structure of the diffusion process. The core idea is to optimize the lower bound on the log-likelihood of the data. Formally, starting from the marginal likelihood, the following derivation holds:

$$\begin{aligned}
\log L(\theta|x_0) &= \log p_\theta(x_0) \\
&= \int q(x_{1:T}|x_0) \cdot (\log p_\theta(x_0)) dx_{1:T} \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} [\log p_\theta(x_0)] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{p_\theta(x_{1:T}|x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T}) q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0) q(x_{1:T}|x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] + \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] + D_{\text{KL}}(q(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0)).
\end{aligned}$$

The Kullback-Leibler divergence, D_{KL} , quantifies the discrepancy between the true forward dynamics and the inverse dynamics learned by the model. However, since the D_{KL} term is always nonnegative, we obtain the inequality:

$$\log p_\theta(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right], \quad (2.9)$$

where the term on the right-hand side represents the ELBO in diffusion-based generative models. Thus, the log-likelihood of the data can be bounded below by a computable expectation involving the generative and forward processes. Maximization of the ELBO term of Eq. 2.9 guides the reverse process toward the true posterior distribution, underpinning the training objective.

2.4.2 Score-Based diffusion model

Introduced by Song et al. [72], continuous-time DMs represent a powerful class of generative models that simulate the gradual transformation of a complex data distribution into a known prior, typically a Gaussian distribution. This process occurs through the progressive injection of noise over time and is formally described by a forward stochastic differential equation (SDE). The forward SDE defines the dynamics of the diffusion process, starting from real data and slowly degrading it until a simple, tractable distribution is reached. To generate new realistic samples, an inverse SDE is used, which operates in the opposite direction: starting from the prior distribution, it gradually removes noise to reconstruct the original data distribution. Fundamentally, the inverse SDE depends on the temporal gradient field of the perturbed data distribution, known as the *score*. This score represents the logarithm gradient of the density distribution and can be learned through score matching techniques. Once estimated, it allows the simulation of the reverse process to generate data consistent with the original distribution.

From a formal perspective, the goal is to construct a diffusion process $\{x(t)\}_{t=0}^T$ indexed by a continuous time variable $t \in [0, T]$, such that $x(0) \sim p_0(x)$, for which we have a dataset of i.i.d. samples, and $x(T) \sim p_T(x)$, for which we have a tractable

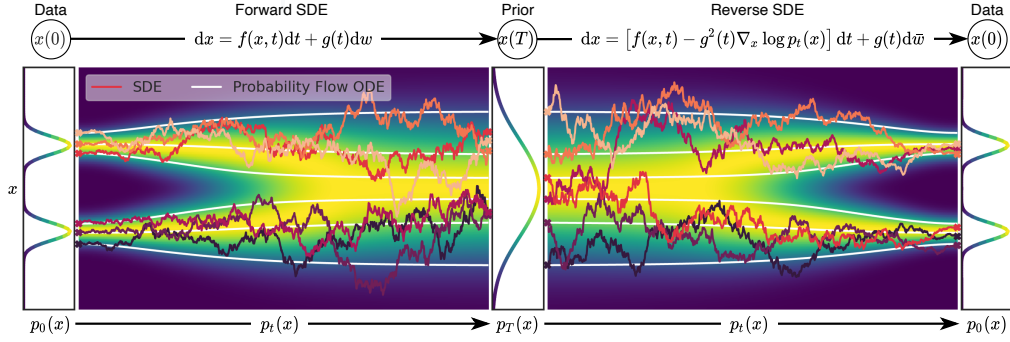


Figure 2.3. Illustration of forward and reverse diffusion processes. The forward process gradually transforms the data distribution $p_0(x)$ into a simple prior $p_T(x)$ by injecting noise over time, modeled by a continuous-time SDE. This process can then be reversed to recover and generate samples from $p_0(x)$, provided that the score of the distribution, $\nabla_x \log p_t(x)$, is known at each intermediate time step (Reproduced from [72]).

form to efficiently generate samples. In this description, we denote $p_0(x)$ as the unknown data distribution and $p_T(x)$ as the prior distribution. This diffusion process can be modeled as a solution of Itô SDE:

$$dx = f(x, t)dt + g(t)dw \quad (2.10)$$

where w denotes the standard Wiener process, also known as Brownian motion, the function $f(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a vector-valued function called the *drift coefficient*, which governs the deterministic part of the dynamics, and $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is a scalar function known as the *diffusion coefficient*, responsible for the stochastic component of the process. Typically, the distribution $p_T(x)$ is chosen as a simple prior that contains no information about the data distribution $p_0(x)$, such as a Gaussian distribution with fixed mean and variance.

The generative process begins with samples drawn from the prior distribution $x(T) \sim p_T(x)$. A remarkable result from Anderson [75] states that the time-reversal of a diffusion process is itself a diffusion process, which evolves according to the equation:

$$dx = \left[f(x, t) - g(t)^2 \nabla_x \log p_t(x) \right] dt + g(t)d\bar{w}. \quad (2.11)$$

Here, $\bar{w}$ denotes a standard Wiener process when time flows backward from T to 0 , and dt represents an infinitesimal negative time step. Once the score distribution $\nabla_x \log p_t(x)$ is known for all t , the reverse SDE can be simulated to generate samples from the original data distribution p_0 . We illustrate the forward and reverse dynamics in Figure 2.3.

The estimation of the score distribution is obtained by training a score-based model on data samples using score matching [76, 77]. More precisely, to estimate $\nabla_x \log p_t(x)$, we train a time-dependent score-based model $s_\theta(x, t)$ by minimizing the following objective:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)|x(0)} \left[\left\| s_\theta(x(t), t) - \nabla_{x(t)} \log p_{0t}(x(t)|x(0)) \right\|_2^2 \right] \right\}, \quad (2.12)$$

where $\lambda : [0, T] \rightarrow \mathbb{R}_{>0}$ is a positive weighting function and t is uniformly sampled in the interval $[0, T]$. In addition, x_0 is drawn from the data distribution $p_0(x)$, and x_t is sampled from the conditional distribution $p_{0t}(x_t|x_0)$, which describes the perturbed data at time t given the original sample x_0 . Under the assumption of sufficient data and model capacity, score matching guaranties that the output of the model parameterized with the optimal solution of Eq. 2.12, denoted by $s_{\theta^*}(x, t)$, is equal to the score function $\nabla_x \log p_t(x)$ for almost all x and t .

2.4.3 Discrete denoising diffusion probabilistic models

Discrete Denoising Diffusion Probabilistic Models (3DPMs) constitute a class of DMs specifically designed for discrete-valued random variables. The first formulation of such models was proposed by Sohl-Dickstein et al. [11] in the context of binary variables. This approach was subsequently extended by Hooeboom et al. [73] to categorical variables, employing transition matrices with uniform transition probabilities. In what follows, we introduce the generalized framework by Austin et al. [74] for diffusion over categorical random variables that encompasses the earlier models as particular instances.

Let $x_t, x_{t-1} \in \{1, \dots, K\}$ be discrete random variables with K categories. The probabilities of forward transitions are represented by a matrix $Q_t \in \mathbb{R}^{K \times K}$, where each entry $[Q_t]_{ij} = q(x_t = j | x_{t-1} = i)$ denotes the probability of transitioning from category i to category j at time t . To express this transition in terms of categorical distributions, we use the one-hot encoding of x_{t-1} , denoted as a row vector. For example, if $K = 4$ and $x_{t-1} = 2$, then the one-hot representation is $x_{t-1} = [0, 1, 0, 0]$. Using this representation, the forward transition can be written as

$$q(x_t | x_{t-1}) = \text{Cat}(x_t; p = x_{t-1} Q_t), \quad (2.13)$$

where $\text{Cat}(x; p)$ denotes a categorical distribution on the one-hot row vector x with discrete probabilities given by the row vector p , and $x_{t-1} Q_t$ is a product of a row vector and a matrix. Assuming that the transition matrix Q_t is applied independently to each feature of the data sample, the forward diffusion process can be compactly expressed using categorical distributions. In particular, starting from the initial state x_0 , the marginal distribution at time t is given by:

$$q(x_t | x_0) = \text{Cat}(x_t; p = x_0 \bar{Q}_t), \quad \text{with } \bar{Q}_t = Q_1 Q_2 \dots Q_t. \quad (2.14)$$

The design of the transition matrices Q_t plays a central role in modeling the behavior of the diffusion process. In addition, each matrix must mathematically satisfy two essential conditions. First, every row of Q_t must sum up to one to ensure conservation of the probability mass over time. Second, the cumulative product $\bar{Q}_t$ must converge to a known stationary distribution when t becomes large. An example that satisfies these conditions is the transition matrix for uniform noise, defined as follows:

$$Q_t = (1 - \beta_t)I + \frac{\beta_t}{K} \mathbf{1}\mathbf{1}^T \quad (2.15)$$

where $\beta_t \in [0, 1]$, I is the identity matrix, and $\mathbf{1}$ is a column vector consisting of K ones. In this form, Q_t ensures that the diffusion process converges to a uniform

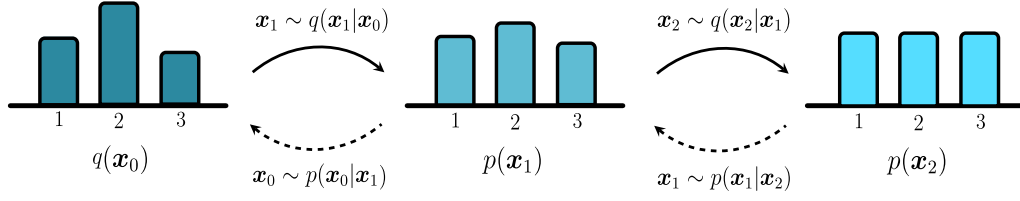


Figure 2.4. Example of 3DPMs for a 3-categorical random variable in the uniform noise scenario. An initial data distribution $q(x_0)$ is transformed into a uniform categorical distribution $p(x_2)$ after two time steps. The forward transition kernel is $q(x_t|x_{t-1})$, while $p(x_{t-1}|x_t)$ is the transition kernel of the backward process obtained by training an ANN.

distribution over categories when t becomes large. An example of 3DPMs for a 3-categorical random variable in the uniform noise scenario is illustrated in Figure 2.4. The reverse process can be implemented directly by predicting the logits² of the distribution $p_\theta(x_{t-1}|x_t)$ using a neural network. The training and optimization procedures are the same as those used in the DDPMs framework.

²In statistics, the term *logit* refers to the logarithm of the odds of a probability $p \in (0, 1)$. Formally, it is defined as $\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$. The inverse of the logit is the sigmoid function $\sigma(z) = 1/(1 + e^{-z})$. In the machine learning framework, logits are the raw, not normalized output scores of a final layer of a neural network for each possible class or category before they are converted into probabilities by an activation function such as softmax to produce a probability distribution where all values sum up to 1.

Chapter 3

Quantum Computation

Quantum computing is an alternative computational paradigm that aims to harness the unique properties of quantum systems, such as *superposition*, *coherence*, *interference*, and *entanglement*, to solve complex and computationally intensive problems that go beyond the capabilities even for the most powerful and modern classical (super)computers available today. In this chapter, we provide an introduction to the main concepts of quantum computing and quantum information theory [78]. We begin in Section 3.1 by introducing the basic unit of information at the quantum level, i.e. the *quantum bit*, and its peculiar quantum-mechanical properties. In Section 3.2 we describe the composition of a quantum bit in more complex systems. Next, in Section 3.3 we will see how to process and manipulate quantum information stored in a multi-qubit system using *quantum logic gates*. Section 3.4 describes how to extract classical information from a quantum system by *measurement operations* to perform computational tasks. This chapter concludes in Section 3.5, where we present the necessary criteria for the physical implementation of a quantum computer.

3.1 Quantum bit

In quantum computing, the fundamental unit of information is the *quantum bit*¹, or *qubit* for short. A qubit is a physical object that can be realized by any two-level quantum-mechanical system. Currently, the most popular candidates for implementation are superconducting qubits [80], photonic setups [81], ion traps [82], and the topological properties of quasi-particles [83].

Independently of its physical realization, a qubit can be treated as an abstract model. In this framework, a qubit q is a mathematical object associated with a complex two-dimensional Hilbert space $\mathcal{H}_q \simeq \mathbb{C}^2$. A widely adopted basis for this space is the *computational basis*, consisting of the orthonormal vectors $|0\rangle$ and $|1\rangle$, which correspond to the classical bit values 0 and 1. Here, pure² quantum states are represented using the *bra-ket notation*, which was introduced by P. Dirac [84]. Unlike the state of a classical bit, which can assume only one of the two values 0 or 1 at a time, a generic pure state $|\psi\rangle$ of a qubit is a complex linear combination of

¹The term was coined by Benjamin Schumacher during a conversation with William Wootters in 1995 [79].

²The terminology of pure and mixed states will be clarified and explained in Section 4.2.

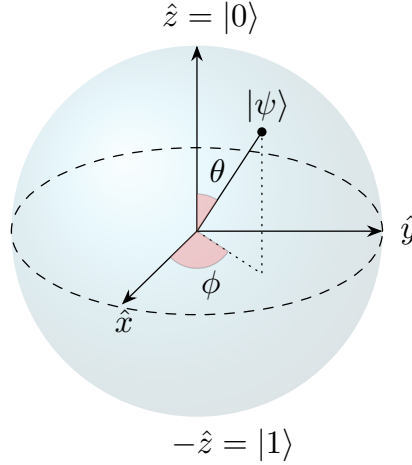


Figure 3.1. Bloch sphere representation of a single qubit. A pure state $|\psi\rangle$ is represented as a point of coordinates (θ, ϕ) on the surface of a unit sphere, whose poles correspond to the basis states $|0\rangle$ and $|1\rangle$.

the basis states as follows:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad \alpha, \beta \in \mathbb{C}. \quad (3.1)$$

The complex numbers α and β are known as *probability amplitudes* and satisfy the *normalization* condition $|\alpha|^2 + |\beta|^2 = 1$. The state in Eq. 3.1 is commonly referred to as a (coherent) *superposition*, and the *relative phase* between α and β is responsible for quantum *interference* phenomena. According to Born's rule, when we perform a measurement on the computational basis of this state, we obtain the result 0 with probability $|\alpha|^2$ and the result 1 with probability $|\beta|^2$. Sometimes, it can be useful to represent the basis states $|0\rangle$ and $|1\rangle$ as column vectors. In this representation, the pure state given in Eq. 3.1 is written as:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \text{with} \quad |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad (3.2)$$

The pure states of a qubit are characterized by four real parameters (two complex numbers). However, it is possible to reduce the number of independent real parameters required to define a pure qubit state to two. This reduction follows from the normalization condition and the fact that states differing only by a global phase, such as $|\psi\rangle$ and $e^{i\delta} |\psi\rangle$ with $\delta \in \mathbb{R}$, represent the same physical state, since an overall phase does not have an observable effect. A useful way to geometrically visualize the state (and its evolution) of a qubit is via the Bloch sphere³: a three-dimensional unit radius sphere in which the basis states $|0\rangle$ and $|1\rangle$ correspond to the north and south poles, respectively. The Bloch sphere representation of a qubit state is illustrated in Figure 3.1. In this representation, a pure state of a qubit can be rewritten (up to an

³Ettore Majorana was actually the first to introduce this sphere to represent spinors as points on the surface of a sphere in 1932 [85]. Later, Bloch and Rabi extended and spread Majorana's results.

overall global phase) in the following form:

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle, \quad (3.3)$$

where the real numbers θ and ϕ are the polar and azimuthal angles, respectively, defining a unique point on the surface of the sphere, with $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$.

Although a qubit can exist in a *continuum* of pure states between the computational basis states $|0\rangle$ and $|1\rangle$, the accessible information that can be extracted from a qubit through measurement is strictly classical in nature. This was formally established by the Holevo theorem (or bound) [86], which states that although a single qubit can exist in a superposition of states and thereby encode more information than a classical bit, the amount of classical information that can be recovered through a single measurement is limited to at most one classical bit. Furthermore, after the measurement operation, the initial state of the qubit *collapses* to one of its two basis states, destroying the quantum property of superposition.

Despite the intrinsic dichotomy between the properties of a qubit state before and after the measurement, it is possible to manipulate quantum systems by performing different operations on the quantum states. Furthermore, the peculiar properties of quantum states can be exploited during the intermediate steps of quantum algorithms before measurement operations (for example, the superposition allows the ability to process multiple inputs simultaneously [87, 25, 24]). This enables quantum computing to obtain potential speedups over its classical counterpart in solving complex intractable problems.

3.2 Multi-qubit systems

A classical computer operates using billions of transistors, which act as binary switches representing bits: 0 (off) and 1 (on). By integrating these transistors into complex circuits within a chip, a computer can store and process data information to solve computational tasks. Similarly, the core of a quantum computer is the Quantum Processing Unit (QPU) [88] that is composed of and operates on systems of n qubits. In the following, we introduce the mathematical formalism used to describe a system of n qubits and then compare it to a system of n classical bits. Successively, we describe how quantum systems composed of two or more qubits can give rise to the phenomenon of *quantum entanglement*, which has no analogue in classical physics.

According to quantum theory, the Hilbert space $\mathcal{H}_Q$ of a quantum system composed of n qubits $q_1, \dots, q_n$ is given by the n -fold tensor product of the single-qubit Hilbert spaces:

$$\mathcal{H}_Q = \bigotimes_{i=1}^n \mathcal{H}_{q_i} \simeq \mathbb{C}^{2^n}. \quad (3.4)$$

This space has dimension 2^n and is spanned by 2^n orthonormal basis vectors $|i\rangle$, where the index $i \in \{0, 1\}^n$ denotes a binary string of length n . A general pure state $|\Psi\rangle$ of the composite system S is an arbitrary complex linear combination of the basis states:

$$|\Psi\rangle = \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle, \quad (3.5)$$

where the coefficients $\alpha_i \in \mathbb{C}$ satisfy the normalization constraint $\sum_i |\alpha_i|^2 = 1$. We emphasize that a system of n classical bits can occupy only one of the 2^n possible states at a time. In contrast, a general quantum state of n qubits is described by 2^n complex amplitudes. This means that at least 2^n classical bits may be required to represent the state of a quantum system. This explains why it is difficult for classical computers to simulate some quantum systems.

If a quantum system consists of two or more subsystems, it can give rise to an intrinsic property of their collective states known as the *entanglement*⁴, which has no counterpart in classical mechanics. The quantum phenomenon of entanglement enables qubits to exhibit measurement correlations that do not exist at the classical level. In particular, when we measure the state of one qubit in an entangled pair, this instantaneously determines the state of the others, regardless of the physical distance between them. From a formal description, a generic pure state in the Hilbert space of the composite system S is said to be *separable* if it can be written as:

$$|\Psi_{\text{sep}}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle, \quad |\Psi_{\text{sep}}\rangle \in \mathcal{H}_S, \quad (3.6)$$

where $|\psi_i\rangle$ is the state of the i -th subsystem associated with the Hilbert space $\mathcal{H}_i$. The states present in $\mathcal{H}_S$ that cannot be expressed in this form are called *entangled states*. A paradigmatic example of two-qubit entangled states is given by the four Bell states:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \quad (3.7)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \quad |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle). \quad (3.8)$$

Entanglement is a powerful resource that allows quantum algorithms and protocols in certain cases to overcome the limitations of classical computation and classical information theory.

Remarkable examples are the two complementary quantum communication protocols: *quantum teleportation* [90] and *superdense coding* [91]. In particular, in the former, entanglement enables the transferring of quantum information (states) between distant locations, while, in the latter, entanglement allows for the transmission of two classical bits using a single qubit.

3.3 Quantum logic gates

So far we have introduced and described how quantum information is stored in quantum systems and their properties. How can we manipulate and process this information using a quantum computer? In quantum algorithms, the first step is to prepare a set of n qubits, called a *quantum register*, conventionally in the pure state $|0\rangle^{\otimes n}$. Next, the qubit states are manipulated and processed by *unitary* transformations U that are in the jargon called *quantum logic gates* (or simply quantum gates), analogous to the terminology used in classical computation where logic operations are performed on bits. The sequence of operations in a quantum

⁴The term *entanglement* was introduced for the first time by E. Schrödinger in 1935 [89].

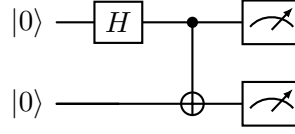


Figure 3.2. Quantum circuit to realize the Bell state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. The quantum register consists of two single qubits that are represented by distinct wires. Both qubits are initially in the state $|0\rangle$. Next, a single-qubit gate (H) is applied to the first qubit, followed by a two-qubit gate (CNOT). Time evolves from left to right. Measurements are performed on the qubits at the end of the circuit and are indicated by the meter symbols.

register can be represented graphically using a set of circuit symbols which will be introduced in Tables 3.1 and 3.2. An illustrative example of a quantum circuit diagram is shown in Figure 3.2.

The unitary evolution of *closed* quantum systems implies that quantum computation is *reversible*, since for any operation U also the inverse operation $U^\dagger$ is valid and such that $UU^\dagger = I$ (where the *adjoint* operation $U^\dagger$ is the transpose and complex conjugate of U). Moreover, the number of inputs to a quantum gate is equal to the number of outputs, which is not always the case for classical gates (e.g., the AND gate has two inputs and one output). This implies that, in principle, no information is lost during the operation of quantum gates. Quantum gates are grouped according to the number of qubits on which they act, such as single-qubit gates, two-qubit gates, and so on. Any n -qubit gate can be represented as $2^n \times 2^n$ complex unitary matrices. In the following, we describe the most important and commonly used single-qubit and two-qubit gates. Next, we present the concept of a *universal gate set* for quantum computation, that is, the fact that any n -qubit gate can be *efficiently* approximated using only single-qubit and two-qubit gates.

3.3.1 Single-qubit gates

In principle, the set of single-qubit gates is infinite. In fact, any unitary transformation acting on a qubit in the form $U = e^{iH}$, with H Hermitian ($H = H^\dagger$), corresponds to a single-qubit gate. However, in practice, only a finite subset is used as elementary gates in the sense that their composition enables the implementation of any arbitrary single-qubit unitary operation. In the following, we introduce the most commonly used single-qubit gates in the literature. In Table 3.1, we report them together with their circuit representation, as depicted in quantum circuits, and their matrix representation on computational basis $\{|0\rangle, |1\rangle\}$.

A trivial gate is the identity gate I , which is represented by the identity matrix,

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad (3.9)$$

and whose action does not modify the quantum state:

$$I|\psi\rangle = |\psi\rangle. \quad (3.10)$$

Next, a particularly important set of single-qubit gates is composed of the Pauli

gates $\{X, Y, Z\}$, whose matrix representations in the computational basis are:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad (3.11)$$

and their actions on the two basis states are as follows:

$$X |0\rangle = |1\rangle, \quad X |1\rangle = |0\rangle \quad (3.12)$$

$$Y |0\rangle = i |1\rangle, \quad Y |1\rangle = -i |0\rangle \quad (3.13)$$

$$Z |0\rangle = |0\rangle, \quad Z |1\rangle = -|1\rangle. \quad (3.14)$$

The Pauli gates X , Y and Z correspond, respectively, to a rotation of the state around the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes of the Bloch sphere by an angle of π radians. The Pauli- X gate is the quantum equivalent of the NOT gate for classical computation, as it acts by flipping the basis states $|0\rangle$ and $|1\rangle$ of the qubit. The Pauli matrices together with the identity matrix form a basis in the space of complex 2×2 matrices. Furthermore, Pauli matrices are the generators of three useful classes of unitary matrices when they are exponentiated, known as *Pauli rotations gates*. They are defined by the following equations:

$$R_X(\theta) \equiv e^{-iX\theta/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X = \begin{pmatrix} \cos\frac{\theta}{2} & -i \sin\frac{\theta}{2} \\ -i \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad (3.15)$$

$$R_Y(\theta) \equiv e^{-iY\theta/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad (3.16)$$

$$R_Z(\theta) \equiv e^{-iZ\theta/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}, \quad (3.17)$$

where the passage from exponential to trigonometric form is obtained using the definition of the exponential function of a matrix and the involutory property ($A^2 = I$) of Pauli matrices:

$$e^{-i\omega A} \equiv \sum_{k=0}^{\infty} \frac{(-i\omega A)^k}{k!} = \sum_{k \in \text{even}} \frac{\omega^k}{k!} I - \sum_{k \in \text{odd}} \frac{\omega^k}{k!} A = \cos \omega I - i \sin \omega A. \quad (3.18)$$

The Pauli rotations act on a qubit state rotating it around the respective axis in the Bloch sphere by an amount indicated by the angle parameter. As we will see later, these operations are fundamental gates of variational quantum circuits, because they are used to introduce free and learnable parameters into quantum machine learning algorithms.

In addition to the aforementioned gates, other fundamental single-qubit quantum gates are the Hadamard gate (denoted by H) and the phase gate (denoted by $P(\delta)$). The Hadamard gate is usually used at the beginning of the quantum algorithms routines to create *superposition* states. Its matrix representation in the computational basis state is as follows:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad (3.19)$$

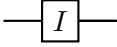
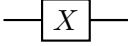
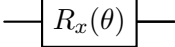
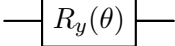
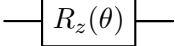
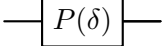
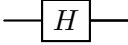
Name	Symbol/Circuit rep.	Matrix representation
Identity		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Pauli-X		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Pauli rotation-X		$\begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$
Pauli rotation-Y		$\begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$
Pauli rotation-Z		$\begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$
Phase $P(\delta)$		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{pmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Table 3.1. A summary of common single-qubit gates. Their name, circuit representation, and matrix form in the computational basis are illustrated.

and its action on the basis states is:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (3.20)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (3.21)$$

The Hadamard gate can also be written as a linear combination of the Pauli gates: $H = \frac{X+Z}{\sqrt{2}}$.

Another essential single-qubit operation is the phase gate $P(\delta)$, which corresponds to a rotation of the state $|\psi\rangle$ by an angle δ around the $\hat{z}$ -axis of the Bloch sphere. Its matrix representation in the computational basis is given by:

$$P(\delta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{pmatrix}, \quad (3.22)$$

and it acts on the basis state as follows:

$$P(\delta) |0\rangle = |0\rangle, \quad (3.23)$$

$$P(\delta) |1\rangle = e^{i\delta} |1\rangle. \quad (3.24)$$

Some particular instances are the S gate ($\delta = \pi/2$), the T gate ($\delta = \pi/4$), and the Z gate ($\delta = \pi$). As we shall see successively, phase gates play a fundamental role in the quantum Fourier transform circuit.

All other single-qubit gates can be expressed in terms of the aforementioned gates. In fact, it can be demonstrated [78] that any arbitrary single-qubit unitary gate U can be decomposed into a product of three rotation gates as follows:

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta), \quad (3.25)$$

where the global phase factor $e^{i\alpha}$ is often omitted in the decomposition.

3.3.2 Two-qubit gates

To implement useful quantum algorithms, we must enable interactions between two or more qubits via multi-qubit gates. However, as we will discuss in Section 3.3.3, any multi-qubit gate can be decomposed into a sequence of single-qubit and two-qubit gates. Here, we introduce the most commonly used two-qubit gates.

Among two-qubit gates, controlled gates are particularly important, as they enable the generation of entanglement. The prototypical example is the controlled-NOT (CNOT) gate that acts on two qubits referred to as *control* and *target*. Its definition and matrix representation on the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ are given by:

$$\text{CNOT} \equiv |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.26)$$

and its action is as follows:

$$\text{CNOT} = |00\rangle = |00\rangle \quad (3.27)$$

$$\text{CNOT} = |01\rangle = |01\rangle \quad (3.28)$$

$$\text{CNOT} = |10\rangle = |11\rangle \quad (3.29)$$

$$\text{CNOT} = |11\rangle = |10\rangle. \quad (3.30)$$

The CNOT gate applies the Pauli X gate to the target (the second qubit) if the control (the first qubit) is in the state $|1\rangle$; otherwise, it applies the identity gate. More generally, one can define the controlled- U quantum gate, which applies a unitary transformation to the state of the target qubit if the control qubit is in the state $|1\rangle$; otherwise, it applies the identity gate. Its definition and matrix representation are:

$$CU \equiv |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix} \quad (3.31)$$

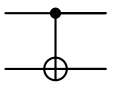
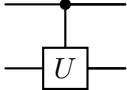
Name (abbr.)	Symbol/Circuit rep.	Matrix representation
Controlled-NOT (CNOT)		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Controlled- U (CU)		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix}$
SWAP		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Table 3.2. Summary of the most important two-qubit operations. Each row shows the gate name (with abbreviation), its circuit representation, and its matrix form in the computational basis.

Furthermore, a k -controlled unitary gate $C^k(U)$ can be realized, where the operator U is applied if all k controllers are in quantum state $|1\rangle$; otherwise, the identity operation is applied. This family of gates can be decomposed into single-qubit and CNOT gates [92]. An example is the Toffoli gate, which implements a CCNOT gate.

Another very important quantum gate is the SWAP gate⁵ whose action is to exchange the state of two qubits:

$$\text{SWAP}(|\psi_1\rangle \otimes |\phi_2\rangle) = |\phi_2\rangle \otimes |\psi_2\rangle, \quad (3.32)$$

and it can be represented in matrix form as:

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.33)$$

The SWAP gate is useful in hardware QPU settings to allow operations between qubits that are not adjacency. In fact, if two qubits are not physically connected, we can simply swap one of those qubits with another that is physically connected to the other qubit. In Figure 3.2 we report the most important two-qubit gates, showing their circuit symbol and matrix representation on the computational basis.

⁵The SWAP gate is not an elementary operation: it is equivalent to performing three alternating CNOT gates.

3.3.3 Universal gate sets

In classical computation, a small and finite set of gates (such as {AND, OR, NOT}, or even the classical Toffoli gate alone) can be used to compute any Boolean function $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$. Such a set of gates is said to be *universal for classical computation*. Similarly, in quantum computation, a set of gates is said to be *universal* if any unitary operation U can be approximated to arbitrary precision by a quantum circuit composed of a finite number of gates from that set. Moreover, a universal quantum gate set must be capable of generating superposition states, entanglement, and complex amplitudes.

The first example of a universal set of quantum gates is given by CNOT together with all single-qubit gates, since it can be shown that any unitary operation U on an arbitrary system of n qubits can be decomposed into single-qubit gates and CNOT gates [92]. Other sets are: {CNOT, H , T } [93]. {CNOT, $R_y(\pi/4)$, S } [94], and also {Toffoli, H } [95]. Thus, there exist multiple universal sets of quantum gates for quantum computation.

Universal sets can be employed to design real quantum devices, raising the issue of determining the most suitable set of gates in practical implementations. Consequently, an important question concerns the efficiency of the approximations required, namely: how efficiently can a given universal set approximate an arbitrary unitary operation? In general, the answer depends on how easily certain quantum operations can be implemented on different QPU architectures. However, a theoretical foundation for this question is provided by the Solovay–Kitaev theorem [96], which guarantees that any unitary operation can be *efficiently* approximated with an accuracy ϵ using a finite sequence of gates whose length grows only polylogarithmically with the desired precision, namely $O(\log^c(1/\epsilon))$, where c is a small constant. Next, we present the theorem for the group $SU(2)$, which consists of the set of all unitary matrices of size 2×2 with determinant equal to 1.

Theorem 3.3.1 (Solovay-Kitaev). *Let $\mathcal{G}$ be a finite set of elements in $SU(2)$ containing its own inverses (so $g \in \mathcal{G}$ implies $g^{-1} \in \mathcal{G}$), and such that the group $\langle \mathcal{G} \rangle$ they generate is dense in $SU(2)$. Consider some desired accuracy $\epsilon > 0$. Then there is a constant c such that for any $U \in SU(2)$, there is a finite sequence S of gates from $\mathcal{G}$ of length $O(\log^c(1/\epsilon))$, where $c \approx 4$, such that $\|S - U\| \leq \epsilon$.*

More generally, the theorem also holds in $SU(d)$ for any fixed d [97]. The demonstration and a more detailed discussion can be found in [78, 97], and are not the subject of this manuscript.

3.4 Measurements and expectation values

At the end of a quantum circuit (or sometimes during it, depending on the task), we need to extract classical information from quantum systems through measurements to perform computational tasks. In quantum theory, the measurement process is a delicate topic that involves the transition from quantum to classical behavior. Here we formally introduce the concepts of quantum measurements and how they can be used to infer physical properties of quantum systems.

Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators* acting on the state space of the system being measured, according to quantum theory postulates. The index m refers to the possible outcomes of the measurement operation. If the state of the quantum system immediately before the measurement is $|\psi\rangle$, then the probability that the outcome m occurs is given by

$$p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle. \quad (3.34)$$

The state of the system after the measurement process is as follows:

$$\frac{M_m |\psi\rangle}{\sqrt{\langle \psi | M_m^\dagger M_m | \psi \rangle}} \quad (3.35)$$

The measurement operators are such that they satisfy the *completeness equation*:

$$\sum_m M_m^\dagger M_m = I, \quad (3.36)$$

which ensures that the total probability sums to one:

$$\sum_m p(m) = \sum_m \langle \psi | M_m^\dagger M_m | \psi \rangle = 1. \quad (3.37)$$

Projective measurements

A special class of quantum measurements, known as projective measurements, plays a central role in numerous applications of quantum computing and quantum machine learning. A projective measurement corresponds to a physical property, called *observable*, and is described by a Hermitian operator O acting on the Hilbert space of the system. By the spectral decomposition theorem, an observable O can be expressed as:

$$O = \sum_m m P_m, \quad (3.38)$$

where $P_m = |m\rangle\langle m|$ is the projector onto the eigenspace of O corresponding to the eigenvalue m . The possible results of the measurement are the *real* eigenvalues m of the observable. The probability of obtaining the result m after the measurement operation is given by:

$$p(m) = \langle \psi | P_m | \psi \rangle. \quad (3.39)$$

After the measurement, the system is in the state:

$$\frac{P_m |\psi\rangle}{\sqrt{p(m)}}. \quad (3.40)$$

Projective measurements ($\sqrt{P_m} \equiv M_m$) in addition to satisfying the completeness relation also satisfy the conditions that are orthogonal projectors, that is, M_m are Hermitian and $M_m M_{m'} = \delta_{m,m'} M_m$. In quantum circuits, projective measurements are represented by a meter symbol which is shown in Figure 3.3.

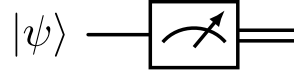


Figure 3.3. Meter symbol. The single line on the left side represents a qubit, while the two lines on the right represent a bit of classical information.

Expectation values of an observables

In addition, projective measurements can be used to infer the physical properties of the system. In particular, the *expectation value* of an observable O for a quantum system in state $|\psi\rangle$ is defined as:

$$\langle O \rangle = \sum_m m p(m) \quad (3.41)$$

$$= \sum_m m \langle \psi | P_m | \psi \rangle \quad (3.42)$$

$$= \langle \psi | O | \psi \rangle. \quad (3.43)$$

It represents the average result one would obtain by repeatedly measuring the observable O on identically prepared copies of the system in the pure state $|\psi\rangle$. A useful result relating the expectation value and the trace of an operator is the following. Suppose $|\psi\rangle$ is a unit vector and A is an arbitrary operator, then it holds:

$$\langle \psi | A | \psi \rangle = \text{tr}(A |\psi\rangle\langle\psi|) \quad (3.44)$$

The standard deviation associated with observations of the observable O is given by:

$$\Delta(O) = \sqrt{\langle O^2 \rangle - \langle O \rangle^2}. \quad (3.45)$$

The standard deviation is a measure that quantifies the spread of observed values when a large number of experiments are performed in which the state $|\psi\rangle$ is prepared and the observable O is measured.

3.5 DiVincenzo criteria for quantum computation

In 2000, physicist David DiVincenzo proposed a set of necessary conditions, known as the DiVincenzo criteria, that any physical system must satisfy to effectively implement a quantum computer. The seven DiVincenzo criteria are divided into two categories: five concern quantum computation, while the remaining two refer to quantum communication. We introduce and briefly describe them here (for a more detailed discussion, see [98]). The criteria are the following:

1. *A scalable physical system with well characterized qubits.*

Independently of the actual physical implementation, a quantum computer must have a well-defined and scalable system of qubits. Scalability is an essential requirement for the development of quantum computing as it ensures

that the physical system containing a set of qubits can be expanded to handle increasingly complex calculations. Furthermore, a "well-characterized" qubit means that the physical parameters of a qubit should be accurately known, such as the internal Hamiltonian and its interactions with other qubits and the external environment.

2. *The ability to initialize the state of the qubits to a simple fiducial state.*
For the execution of any quantum algorithm to be reliable and reproducible, the quantum system must be initialized in a well-defined and controllable state, typically a pure state such as $|000 \dots 0\rangle$.
3. *Long relevant decoherence times, much longer than the gate operation time.*
Quantum coherence is the property of a quantum system in which the different states of qubits maintain well-defined phase relationships, and it is related to the superposition property. When the phenomenon of decoherence occurs due to interactions with an external environment, the system will exhibit a classical behavior, and therefore the capabilities of quantum algorithms will be no different from the classical ones. Consequently, the coherence time of a qubit is an essential characteristic and must be significantly longer than the time required to perform quantum gate operations on it.
4. *A universal set of quantum gates.*
A quantum algorithm is typically specified as a sequence of unitary transformations that act on a set (or subset) of n qubits. A universal gate set refers to a collection of elementary quantum gates that can be used to approximate (decompose) any unitary operation on a system of n qubits. This set consists of single-qubit gates and at least an entangling two-qubit gate, such as the Controlled-NOT (CNOT) gate. Moreover, universality is also fundamental to connect the theoretical aspects with the physical implementation of quantum computing.
5. *A qubit-specific measurement capability.*
Extracting computational results from a quantum system is crucial to performing calculations. This requires the ability to accurately and repeatably measure individual qubits. Although, today, quantum measurements are not 100% efficient, there is a trade-off between efficiency and other available resources that results in reliable quantum computation. In fact, if the efficiency of a measurement is q , preparing and measuring more than $1/q$ identical copies of the same qubit will produce reliable results.

The other two criteria are related to quantum communication and the transmission of qubits across different places. They are reported below:

1. *The ability to interconvert stationary and flying qubits;*
2. *The ability faithfully to transmit flying qubits between specified locations.*

Chapter 4

Quantum Computation in the NISQ Era

In the previous chapter, we described the paradigm of quantum computing in the ideal scenario, that is, when computation occurs in the absence of noise. In particular, we considered a qubit as a *closed* quantum system, that is, a system completely isolated from its surrounding environment. In this framework, the states of a qubit are always exactly known, represented as vectors in a Hilbert space, and evolve by unitary transformations. However, currently available quantum processors consist of *physical* qubits that are highly sensitive to unwanted interactions with the external environment. These interactions show up as *quantum noise* and lead to decoherence and errors in quantum computation. Consequently, it is essential to develop a mathematical framework that allows us to describe quantum noise and the behavior of *open* quantum systems [78, 99]. This chapter is structured as follows. We begin in Section 4.1 with a description of the state of the art of current quantum processors, which are called Noisy Intermediate-Scale Quantum (NISQ) devices. In Section 4.2 we introduce the formalism of the *density operator* to represent quantum states, which is a crucial tool when dealing with systems whose states are not known exactly. In Section 4.3 we introduce the formalism of *quantum operations* that allows us to describe both the unitary and non-unitary dynamics of open quantum systems and to mathematically model quantum noise. The chapter concludes in Section 4.5 with a description of the continuous-time dynamics of the open quantum system via the *Kossakowski-Lindblad-Gorini master equation*.

4.1 NISQ devices

The quantum circuit model provides a pictorial and intuitive representation of the logic action of quantum gates on qubits. However, while quantum circuits are the standard way for submitting instructions to a real quantum device, they do not faithfully represent the physical processes occurring within the quantum hardware. In fact, quantum computers are engineered and realized with different quantum physical systems. For example, a commonly used technology employs superconductive circuit-based platforms [100, 101] realized with transmon qubits [102, 103]. This technology is used by companies such as IBM [104], Rigetti Computing [105] and Google [106].

Xanadu [107] instead uses photons as information units within the linear optical quantum computing paradigm [108] to realize their devices. Furthermore, quantum processors can also be implemented by manipulating the properties of single atoms. In this context, IonQ [109] realizes quantum devices with trapped ions [110, 111], while Pasqal [112] and QuEra [113] realize analog quantum computers with neutral rubidium Rydberg atoms held in optical tweezers [114]. Therefore, each quantum computer has a different finite set of gates that can *physically* run on the hardware. This set of gates is called the *native basic gates* and consists of single-qubit and two-qubit gates that differ depending on the physical architecture and the technology used to realize QPUs. Consequently, every gate of the original circuit must be expressed in terms of these native basis gates. The process of decomposing the circuit into a hardware-specific form is called *transpilation* and produces a new quantum circuit that is (approximately) equivalent to the original but can be executed on the real machine. Furthermore, although enormous technological progress has been made in building the first generations of quantum computers, these devices are far from the realization of an universal *fault-tolerant* quantum computer [115], that is, quantum hardware with a large and scalable number of error-corrected qubits. The current era of quantum computing is that of the so-called Noisy Intermediate-Scale Quantum (NISQ) devices [26]. These processors are composed of tens or a few hundred noisy qubits. Quantum noise is the major challenge in the development of fault-tolerant quantum devices. It can arise from various sources, such as interactions with the environment, imperfections in the physical implementation of quantum gates (i.e., not ideal unitary operations), and hardware constraints such as the connectivity of the qubits in the QPU. The error rate of quantum gates depends on the technology used, but is typically of the order of 0.01–1% and 0.1–1% for single and two-qubit quantum gates, respectively. In order to reduce the effects of noise, quantum error correction techniques [116, 117] can be applied, which, however, require an elevated number of *physical* qubits. In this context, Google Quantum AI has recently developed a new quantum device called Willow that seems to allow an exponential reduction of errors while increasing the number of qubits [118].

Despite their limitations, NISQ devices are a powerful tool to explore quantum algorithms and their applications. Recently, in fact, a new class of algorithms has emerged that involves hybrid quantum–classical approaches. The idea is to harness the power of quantum computing together with classical computing resources to compensate for the limitations of NISQ devices. These algorithms are known as *hybrid quantum–classical algorithms* or also as NISQ algorithms [119, 120]. A plethora of these algorithms have been proposed in different applications, for example: image processing [121, 122], quantum chemistry [123, 124], combinatorial and optimization problems [125], search algorithms [126], machine learning tasks [127], linear system solvers [128, 129], and experimental data analysis [130, 131]. Thus, NISQ devices represent an exciting and rapidly evolving stage in the development of quantum computing. The insights gained from computational models based on NISQ algorithms can contribute to the advancement of future generations of quantum computers, ultimately paving the way for fault-tolerant quantum computing.

4.2 Density operator formalism

The density operator provides a suitable means for describing quantum systems whose state is not known exactly. More precisely, suppose that we only know that a quantum system can be in one of a number of states $|\psi_i\rangle$, indexed by i , with respective probability $p_i \in [0, 1]$. To describe this statistical uncertainty on the state of the system, one can consider a weighted mixture of the states in the set $\{p_i, |\psi_i\rangle\}$. This set is called an *ensemble of pure states*. The state of the system is represented by the *density operator* defined as follows:

$$\rho \equiv \sum_i p_i |\psi_i\rangle\langle\psi_i|, \quad (4.1)$$

with $\sum_i p_i = 1$. The density operator is also known as the *density matrix*, and in this work we will use the two terms interchangeably. A quantum state of the form shown in Eq. 4.1 is called a *mixed* state and is used to describe the statistical uncertainty about the state of a quantum system. Mixed states are in contrast to pure states, which are instead used to describe quantum systems whose state is known exactly.

In Eq. 4.1 we have introduced the density operator as a tool to describe ensembles of quantum states. However, it is also possible to provide an intrinsic characterization of density operators that does not rely on the ensemble interpretation. In this approach, the state of a quantum system is an operator ρ that lives in the space of bounded linear operators $\mathcal{B}(\mathcal{H})$ acting on $\mathcal{H}$ and that satisfies the following conditions:

- **Trace condition:** ρ has trace equal to one $\text{tr}(\rho) = 1$.
- **Positivity condition:** ρ is a semidefinite positive operator $\rho \geq 0$.

From these properties follows that the density operator ρ is Hermitian: $\rho = \rho^\dagger$. It can be shown that any operator ρ is a density operator associated with some ensemble of pure state $\{p_i, |\psi_i\rangle\}$ if and only if it satisfies the above conditions.

The density matrix formalism is a powerful tool that includes both pure and mixed states. In fact, a *pure state* corresponds to a situation in which the system is in a well-definite state $|\psi_i\rangle$ with probability $p_i = 1$ and all other probabilities $p_j = 0$ for $j \neq i$. In this case, the density operator becomes:

$$\rho = |\psi_i\rangle\langle\psi_i|. \quad (4.2)$$

In quantum information theory, one can define a measure on a quantum state giving information on how much a state is mixed. This scalar quantity is called *purity* of a quantum state and is defined as:

$$\text{purity} \equiv \text{tr}(\rho^2). \quad (4.3)$$

For pure states $\text{tr}(\rho^2) = 1$, while for mixed states $\text{tr}(\rho^2) < 1$. A special case is the *completely mixed* state, in which the system is equally likely to occupy any of the d orthonormal basis states (i.e., $p_i = 1/d$ for $i = 1, 2, \dots, d$). Its density matrix is given by:

$$\rho = \frac{I}{d}, \quad (4.4)$$

and the purity reaches its minimum value:

$$\text{tr}(\rho^2) = \frac{1}{d}. \quad (4.5)$$

4.2.1 Bloch sphere representation

The Pauli matrices $\{\sigma_x, \sigma_y, \sigma_z\}$ together with the identity matrix I is a basis for the space of 2×2 Hermitian matrices. Thus, any qubit state ρ can be expanded on this basis as follows:

$$\rho = \frac{I + \vec{r} \cdot \vec{\sigma}}{2} = \frac{1}{2} \begin{pmatrix} 1 + r_z & r_x - ir_y \\ r_x + ir_y & 1 - r_z \end{pmatrix}, \quad (4.6)$$

where $\vec{r} = (r_x, r_y, r_z)$ is a real three-dimensional vector such that $\|\vec{r}\| \leq 1$. The vector $\vec{r}$ is known as the *Bloch vector* of the state ρ . If ρ is a pure state, then it holds:

$$\text{tr}(\rho^2) = \text{tr} \left[\left(\frac{I + \vec{r} \cdot \vec{\sigma}}{2} \right)^2 \right] \quad (4.7)$$

$$= \frac{1 + r_x^2 + r_y^2 + r_z^2}{2} \quad (4.8)$$

$$= 1 \quad (4.9)$$

which implies:

$$r_x^2 + r_y^2 + r_z^2 = 1. \quad (4.10)$$

This is the equation of a unit sphere in $\mathbb{R}^3$, and it is another way to derive that pure states of a qubit lie on the surface of the Bloch sphere. For mixed states, the purity satisfies:

$$\text{tr}[\rho^2] \leq 1 \quad \Rightarrow \quad r_x^2 + r_y^2 + r_z^2 \leq 1. \quad (4.11)$$

Thus, mixed states correspond to points inside the Bloch sphere. From Eq. 4.6, the center of the Bloch sphere, ($\vec{r} = (0, 0, 0)$), corresponds to the completely mixed state:

$$\rho = \frac{I}{2}. \quad (4.12)$$

4.2.2 Evolution

According to quantum physics, the evolution of a closed quantum system in the state $|\psi\rangle$ is described by the unitary operator U , and after the evolution has occurred, the system will be in the state $U|\psi\rangle$. Thus, the evolution of the density operator is described by the following equation:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U |\psi_i\rangle\langle\psi_i| U^\dagger = U \rho U^\dagger \quad (4.13)$$

The time evolution of the density operator ρ in a closed quantum system is governed by the von Neumann equation:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar} [H, \rho]. \quad (4.14)$$

where $[A, B] = AB - BA$ denotes the commutator between A and B . In this expression, H is the Hamiltonian of the system, and the equation describes the unitary evolution. In Section 4.3, we will see that the unitary evolution of the density operator represents a special case of dynamics. Indeed, in the theory of open quantum systems, the dynamics of a density operator is described by quantum operations, which provide a mathematical tool that allows us to model even non-unitary dynamics such as those of noisy processes.

4.2.3 Measurements

In density operator language, measurements are described as follows. Suppose that we perform a measurement described by the measurement operators M_m and the initial state of the system is $|\psi_i\rangle$. Then the probability of getting the result m is given by:

$$p(m|i) = \langle \psi_i | M_m^\dagger M_m | \psi_i \rangle = \text{tr}(M_m^\dagger M_m |\psi_i\rangle\langle\psi_i|), \quad (4.15)$$

where in the last equality, we have used Eq. 3.44. Therefore, the probability of obtaining the result m by performing a measurement on a system described by the density operator as in Eq. 4.1 is as follows:

$$p(m) = \sum_i p(m|i) p_i \quad (4.16)$$

$$= \sum_i p_i \text{tr}(M_m^\dagger M_m |\psi_i\rangle\langle\psi_i|) \quad (4.17)$$

$$= \text{tr}(M_m^\dagger M_m \rho). \quad (4.18)$$

The state of the system after the measurement is

$$\rho_m = \sum_i p_i \frac{M_m |\psi_i\rangle\langle\psi_i| M_m^\dagger}{\text{tr}(M_m^\dagger M_m \rho)} \quad (4.19)$$

$$= \frac{M_m \rho M_m^\dagger}{\text{tr}(M_m^\dagger M_m \rho)}. \quad (4.20)$$

In this framework, the expectation value of an operator O follows directly from Eq. 3.44 and the linearity of the trace operation:

$$\langle O \rangle = \text{tr}(O\rho). \quad (4.21)$$

where ρ is the quantum state of the system.

4.2.4 The reduced density operator

One of the most important applications of the density operator is the description of the *subsystems* of a composite quantum system. This is provided by the *reduced density operator*. Suppose that we have a physical system composed of two subsystems A and B , whose state is described by the density operator ρ_{AB} . The reduced density operator for system A is defined by the *partial trace* over subsystem B as follows:

$$\rho_A \equiv \text{tr}_B(\rho_{AB}). \quad (4.22)$$

The partial trace $\text{tr}_B(\cdot)$ is a map of operators defined by

$$\text{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \equiv |a_1\rangle\langle a_2| \text{tr}(|b_1\rangle\langle b_2|), \quad (4.23)$$

where $|a_1\rangle, |a_2\rangle$ are any two vectors of the system A , and $|b_1\rangle, |b_2\rangle$ are any two vectors of the system B . The trace on the right-hand side is the usual trace over the subsystem B :

$$\text{tr}(|b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle. \quad (4.24)$$

The partial trace operation is extended to any arbitrary operators in the composite system AB by requiring, in addition to Eq. 4.23, that it acts linearly on its input.

4.2.5 Fidelity and von Neumann entropy

An important question in quantum information theory is: how can we quantify the similarity between two quantum states? This is addressed by the concept of *quantum fidelity*, which measures the "closeness" between two density operators. Formally, the Uhlmann fidelity [132] between two quantum states ρ and σ is defined as:

$$F(\rho, \sigma) = \left(\text{tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right)^2. \quad (4.25)$$

The square roots in this expression are mathematically well defined because both ρ and $\sqrt{\rho} \sigma \sqrt{\rho}$ are positive semidefinite operators. Furthermore, the square root is defined via the spectral theorem. If (at least) one of the two quantum states is pure, for example $\rho = |\psi_\rho\rangle\langle\psi_\rho|$, then the fidelity simplifies to:

$$F(\rho, \sigma) = (\text{tr}(\sqrt{\rho \sigma \rho}))^2 = \langle\psi_\rho|\sigma|\psi_\rho\rangle, \quad (4.26)$$

where we used the fact that if ρ is pure, then $\sqrt{\rho} = \rho$. Instead, if both states are pure, $\rho = |\psi_\rho\rangle\langle\psi_\rho|$ and $\sigma = |\psi_\sigma\rangle\langle\psi_\sigma|$, then the fidelity becomes:

$$F(\rho, \sigma) = |\langle\psi_\rho|\psi_\sigma\rangle|^2. \quad (4.27)$$

The fidelity between pure quantum states has a straightforward interpretation, i.e., it is the overlap between two quantum states.

Some of the important properties of quantum state fidelity are:

- **Symmetry:** the quantum fidelity is symmetric with respect to its arguments: $F(\rho, \sigma) = F(\sigma, \rho)$.
- **Bounded Values:** for any two density operators ρ and σ , $0 \leq F(\rho, \sigma) \leq 1$. It reaches its maximum, $F = 1$, when the states are identical $\rho = \sigma$.
- **Unitary invariance:** the quantum fidelity is invariant under unitary transformations. Given two density operators ρ and σ evolving under a unitary operator U , it holds: $F(\rho, \sigma) = F(U\rho U^\dagger, U\sigma U^\dagger)$.

Another important quantity is the von Neumann entropy, which is a quantum counterpart of the Shannon entropy of classical information theory. The von Neumann entropy measures the statistical uncertainty associated with the description of a quantum system. For a quantum-mechanical system described by a density matrix ρ , the von Neumann entropy is defined as:

$$S(\rho) \equiv -\text{tr}(\rho \log \rho), \quad (4.28)$$

where the logarithm is taken to base two. Some properties of von Neumann entropy relevant to the concepts discussed in this thesis are the following:

1. The entropy is always non-negative:

$$S(\rho) \geq 0 \quad \forall \rho \in \mathcal{B}(H). \quad (4.29)$$

2. The entropy is zero if and only if ρ is a pure state:

$$S(\rho) = 0 \iff \rho = |\psi\rangle\langle\psi|. \quad (4.30)$$

3. In a d -dimensional Hilbert space, the entropy is at most $\log d$. The entropy reaches its maximum value if and only if the system is in the completely mixed state:

$$S(\rho) = \log d \iff \rho = I/d. \quad (4.31)$$

In Chapter 7, Fidelity and von Neumann entropy will be used, respectively, to evaluate the information loss of a quantum state during the forward diffusion process and as a loss function to train classical and quantum artificial neural networks in the protocols of quantum generative diffusion models.

4.3 Quantum operations

In quantum theory, the term *quantum noise* refers to unwanted interactions that affect quantum systems. It can arise from various sources such as thermal fluctuations, electromagnetic interference, and interactions with a surrounding environment. Furthermore, quantum noise is also generated by *quantum fluctuations* that are typical and *inherent* of quantum systems, hence going much beyond the classical noise sources. In quantum computing, quantum noise represents a major challenge in the development of large-scale, fault-tolerant quantum computers. Different types of quantum noise can affect qubits. For instance, the depolarizing noise destroys the coherences of a quantum state by replacing it with a completely mixed state; the amplitude damping noise drives the qubit to its ground state; phase damping alters the relative phase between the basis states of a qubit. Therefore, understanding and characterizing different types of noise is crucial for designing quantum error correction or mitigation techniques. In the framework of open quantum systems, quantum noise is described by the formalism of *quantum operations*. This formalism turns out to be a key tool for describing the dynamics of open quantum systems, allowing us to address a wide range of physical scenarios: from nearly closed systems that are weakly coupled to their environments to systems that are strongly coupled

to their environments. There exists a large literature on the subject of quantum operation. Here, we report the main references used during the writing of this thesis: [133, 134, 135, 136, 137, 138, 139, 140].

Mathematically, a quantum operation is a map $\mathcal{E}$ that transforms an initial quantum state ρ as:

$$\rho' = \mathcal{E}(\rho), \quad (4.32)$$

where ρ' is the quantum state after transformation. Quantum operations are also known as quantum dynamical maps or quantum channels, and in this manuscript the two terms will be used interchangeably. We have already encountered two simple examples of quantum operations, that is, unitary transformations $\mathcal{E}(\rho) = U\rho U^\dagger$ and measurement operations $\mathcal{E}_m(\rho) = M_m\rho M_m^\dagger$. As mentioned above, quantum operations allow us to describe the evolution of quantum systems in different scenarios. In particular, in quantum computing and quantum information, they turn out to be crucial tools for describing discrete state changes, that is, transformations between an initial state ρ and a final state ρ' , without explicit reference to the passage of time. There exist three different ways of understanding and formalizing quantum operations.

- The first is based on the idea of studying the dynamics of an open quantum system as interaction between a *principal* quantum system and its surrounding *environment*. This approach is known as the *Stinespring representation*.
- The second is known as the *Kraus representation* and represents quantum operations as sum-operation operators on the Hilbert space of the principal system alone.
- The third is via a set of physically motivated axioms that a dynamical map in quantum mechanics must satisfy.

All of these three approaches turn out to be equivalent and offer a powerful tool to understand quantum noise and its effects.

4.3.1 Quantum system coupled to environment

In quantum mechanics, the evolution of a state of a closed quantum system occurs through unitary operators generated by the Hamiltonian of the system. A natural way to describe the evolution of an open quantum system involves modeling its dynamics as emerging from the interaction between a system of interest S and an external environment E . More precisely, system S is considered part of a larger system that also includes its environment E . The combined system $S + E$ forms a closed quantum system, and therefore its evolution is still determined by unitary transformations U_{SE} generated by the Hamiltonian of form $H_{SE} = H_S + H_E + H_{\text{int}}$. The term H_{int} describes the interactions between the system and the environment. Consider the quantum system S prepared in a state ρ_S and the environment initialized in a state ρ_{env} . In addition, we assume that the combined system is in a product state¹

¹Although the system and environment generally do not begin in a product state, assuming such separability is often reasonable in many practical scenarios.

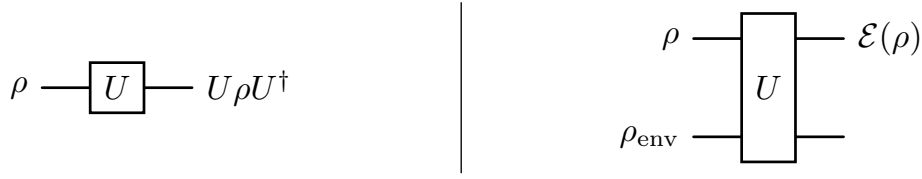


Figure 4.1. Models of closed (left) and open (right) quantum systems in a quantum circuit representation. An open quantum system consists of two parts, the system and an environment. After applying the unitary transformation U , the system no longer interacts with the environment and, in general, the final state of the system $\mathcal{E}(\rho)$ is not related by a unitary transformation to the initial state ρ . The subscript S for the system has been omitted for simplicity.

$\rho_{SE} = \rho_S \otimes \rho_E$. The evolution of state of the combined system is as follows:

$$\rho_{SE} \rightarrow U_{SE} (\rho_S \otimes \rho_E) U_{SE}^\dagger. \quad (4.33)$$

Since we are only interested in the dynamics of S , we can trace out the environmental degrees of freedom:

$$\mathcal{E}(\rho_{SE}) = \text{tr}_E [U_{SE} (\rho_S \otimes \rho_E) U_{SE}^\dagger]. \quad (4.34)$$

This expression constitutes the first of three equivalent formulations by which a quantum operation can be formalized, and it is also known as the *Stinespring representation*. If $H_{\text{int}} \equiv 0$, then $U_{SE} = U_S \otimes U_E$ and $\rho_S \rightarrow U_S \rho_S U_S^\dagger$. In this definition, a natural question that might arise is: how can U_{SE} be specified if the environment has nearly infinite degrees of freedom? It turns out that to properly describe any possible transformation by this approach, if the system of interest has a Hilbert space of dimensions d , it suffices to model the environment as a system with a Hilbert space of dimension no more than d^2 . Finally, the evolution of closed and open quantum systems is depicted through a quantum circuit model as shown in Figure 4.1.

4.3.2 Kraus representation

A more abstract and elegant form for describing quantum operations is the *Kraus representation*. This approach reformulates Eq. 4.34 in terms of operators acting only on the Hilbert space of the principal system. In fact, let $\{|e_k\rangle\}$ be an orthonormal basis for the (finite-dimensional) state space of the environment, and let the

environment be in the initial state² $\rho_E = |e_0\rangle\langle e_0|$. Eq. 4.34 can be rewritten as:

$$\mathcal{E}(\rho_{SE}) = \text{tr}_E \left[U_{SE} (\rho_S \otimes |e_0\rangle\langle e_0|) U_{SE}^\dagger \right] \quad (4.35)$$

$$= \sum_k \left[\langle e_k | U_{SE} | e_0 \rangle \right] \rho_S \left[\langle e_0 | U_{SE}^\dagger | e_k \rangle \right] \quad (4.36)$$

$$= \sum_k E_k \rho_S E_k^\dagger, \quad (4.37)$$

where $E_k \equiv \langle e_k | U_{SE} | e_0 \rangle$ is an operator on the state space of the principal system. The operators $\{E_k\}$ are known as *Kraus operators* and are the operation elements for the quantum operation $\mathcal{E}$. From the requirement that quantum operations be *trace-preserving* (the trace of the state $\mathcal{E}(\rho)$ is equal to one) the *completeness relation* follows:

$$1 = \text{tr}(\mathcal{E}(\rho)) \quad (4.38)$$

$$= \text{tr} \left(\sum_k E_k \rho E_k^\dagger \right) \quad (4.39)$$

$$= \text{tr} \left(\sum_k E_k^\dagger E_k \rho \right), \quad (4.40)$$

and since this relationship is true for all ρ , we have:

$$\sum_k E_k^\dagger E_k = I. \quad (4.41)$$

For completeness, note that there also exist non-trace-preserving quantum operations, characterized by the condition $\sum_k E_k^\dagger E_k \leq I$. These maps describe processes in which additional information about the system is obtained through measurement. However, such operations are not the subject of this thesis.

Finally, the composition $\mathcal{F} \circ \mathcal{E}$ of two quantum operations $\mathcal{E}$ and $\mathcal{F}$, acting on the same quantum system, is itself a quantum operation, as it admits a Kraus representation.

4.3.3 Quantum operation as CPTP map

Quantum operations can also be defined according to a set of physically motivated axioms that ensure that the evolution of quantum states remains consistent with the principles of quantum mechanics. These axioms include trace preservation, linearity, and complete positivity and ensure that quantum operations are mathematically and physically well defined. In this framework, quantum operations are described by Completely Positive and Trace Preserving (CPTP) maps.

A CPTP map $\mathcal{E}$ is a map from the set of density operators of the input space Q_1 to the set of density operators of the output space Q_2 , with the following three axiomatic properties:

²There is no loss of generality in assuming that the environment being in a pure state, since any mixed state can be purified by introducing an auxiliary system.

1. **Trace-preserving:** if $\text{tr}[\rho] = 1$, then $\text{tr}[\mathcal{E}(\rho)] = 1$. This condition ensures that the sum of probabilities remains one after the quantum operation.
2. **Linearity:** the operation $\mathcal{E}$ is a convex-linear map on the set of density matrices, that is, for a set of probabilities $\{p_i\}$ holds:

$$\mathcal{E}\left(\sum_i p_i \rho_i\right) = \sum_i p_i \mathcal{E}(\rho_i). \quad (4.42)$$

3. **Completely positivity:** the quantum operation $\mathcal{E}$ is a completely positive map. That is, if $\mathcal{E}$ maps the density operators of system Q_1 to the density operators of system Q_2 , then $\mathcal{E}(A)$ must be positive for any positive operator $A \in Q_1$. Furthermore, if we introduce an extra system R of arbitrary dimensionality, it must be true that $(I \otimes \mathcal{E})(A)$ is positive for any positive operator A acting on the combined system RQ_1 , where I denotes the identity map on the extra system R .

Finally, we note that a map $\mathcal{E}$ that satisfies the above three axioms admits a Kraus representation if and only if there exists a set of Kraus operators $\{E_k\}$ such that $\sum_k E_k^\dagger E_k = I$. Moreover, it is possible to show that any map in the Kraus representation can be expressed in the Stinespring representation [78].

4.4 Examples of quantum operations on a single qubit

In this section, we examine some concrete examples of quantum noise described by quantum operations on a single qubit. In particular, we provide a mathematical model using Kraus operators for the following types of quantum noise: *bit flip*, *phase flip*, *depolarizing*, *amplitude damping*, and *phase damping*. These noise models are important for understanding and capturing many of the important characteristics and effects of noise that occur in real-world quantum mechanical systems.

4.4.1 Bit flip channels

The single-qubit *bit flip* channel flips the state of a qubit from $|0\rangle$ to $|1\rangle$ (and vice versa) with probability $1 - p$, where $p \in [0, 1]$. In Kraus representation, it is modeled by the following operators:

$$E_0 = \sqrt{p}I = \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad E_1 = \sqrt{1-p}X = \sqrt{1-p} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (4.43)$$

The effect of the bit flip channel on a single qubit can be geometrically visualized by means of the Bloch representation of states of a qubit, and is illustrated in Figure 4.2.

4.4.2 Phase flip channel

The *phase flip* error of a single qubit changes the relative phase of the $|1\rangle$ state while leaving the $|0\rangle$ state unchanged. More precisely, it maps the state $|1\rangle$ to $e^{i\pi}|1\rangle$ with

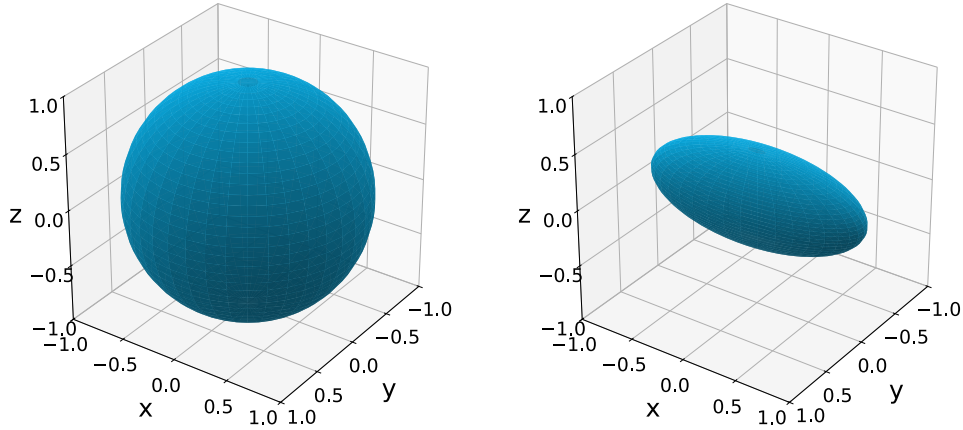


Figure 4.2. The effect of the bit-flip channel on the Bloch sphere for $p = 0.3$. The sphere on the left represents the set of all pure states, while the deformed sphere on the right shows the states after passing through the bit flip channel. Note that the states along the $\hat{x}$ -axis remain unchanged, whereas the $\hat{y}$ - $\hat{z}$ plane is uniformly contracted by a factor of $1 - 2p$.

probability $1 - p$, where $p \in [0, 1]$. It is described by the following Kraus operations:

$$E_0 = \sqrt{p}I = \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad E_1 = \sqrt{1-p}Z = \sqrt{1-p} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (4.44)$$

The effect of the phase flip channel on the Bloch sphere is illustrated in Figure 4.3.

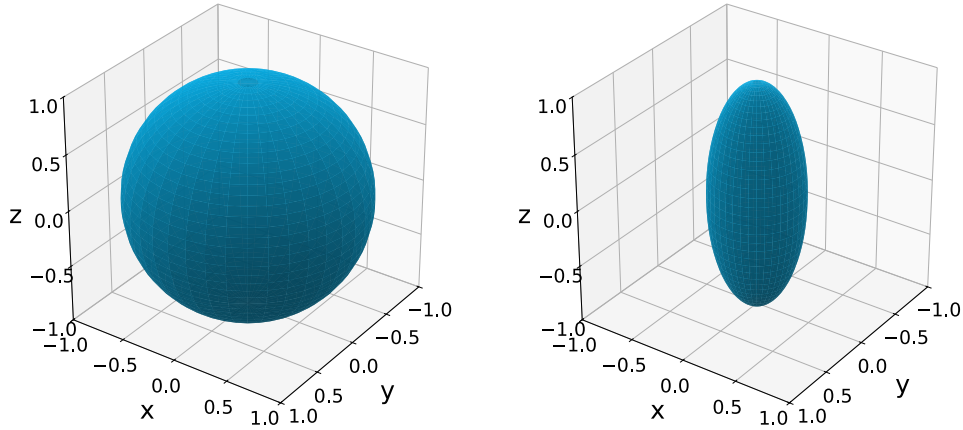


Figure 4.3. The effect of the phase flip channel on the Bloch sphere, for $p = 0.3$. Note that the states on the $\hat{z}$ -axis remain unchanged, whereas the $\hat{x}$ - $\hat{y}$ plane is uniformly contracted by a factor of $1 - 2p$.

4.4.3 Depolarizing channel

One of the main sources of quantum noise that affects quantum computers is *depolarization*, which is responsible for the decoherence phenomena. It is modeled by a depolarizing channel where a qubit state ρ remains unchanged with probability $1 - p$

or becomes a maximally mixed state $I/2$ with probability p . The maximally mixed state represents complete uncertainty about the state of a qubit. Mathematically, the depolarizing channel is described by the following quantum maps:

$$\mathcal{E}(\rho) = \frac{pI}{2} + (1-p)\rho. \quad (4.45)$$

The effect of the depolarizing channel on the Bloch sphere is illustrated in Figure 4.4. The depolarizing channel can also be written using the following set of Kraus operators:

$$E_0 = \sqrt{1 - \frac{3p}{4}} I, \quad E_1 = \sqrt{\frac{p}{4}} X \quad (4.46)$$

$$E_2 = \sqrt{\frac{p}{4}} Y, \quad E_3 = \sqrt{\frac{p}{4}} Z, \quad (4.47)$$

and in a more compact form as:

$$\mathcal{E}(\rho) = \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z). \quad (4.48)$$

The depolarizing channel can alternatively be parameterized as:

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z). \quad (4.49)$$

This expression tells us that the action of the depolarizing channel can be understood as follows: the state ρ is left unchanged with probability $1-p$, and each of the Pauli operators X, Y, Z is applied with probability $\frac{p}{3}$.

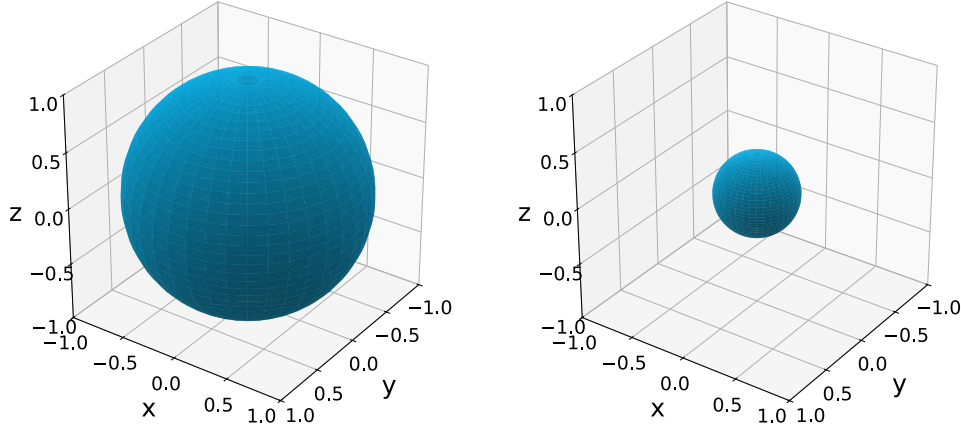


Figure 4.4. The effect of the depolarizing channel on the Bloch sphere, for $p = 0.6$. Note how the entire sphere contracts uniformly as a function of p .

4.4.4 Amplitude damping

Another important process that occurs in quantum systems is *energy dissipation*, such as an atom that spontaneously emits a photon. This type of quantum noise is

characterized by a quantum operation known as *amplitude damping* and is formally represented by the Kraus operators:

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \quad E_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix} \quad (4.50)$$

where $\gamma \in [0, 1]$ is the probability of amplitude damping. The effect of this quantum noise causes a qubit to decay toward its ground state $|0\rangle$, and is shown in Figure 4.5.

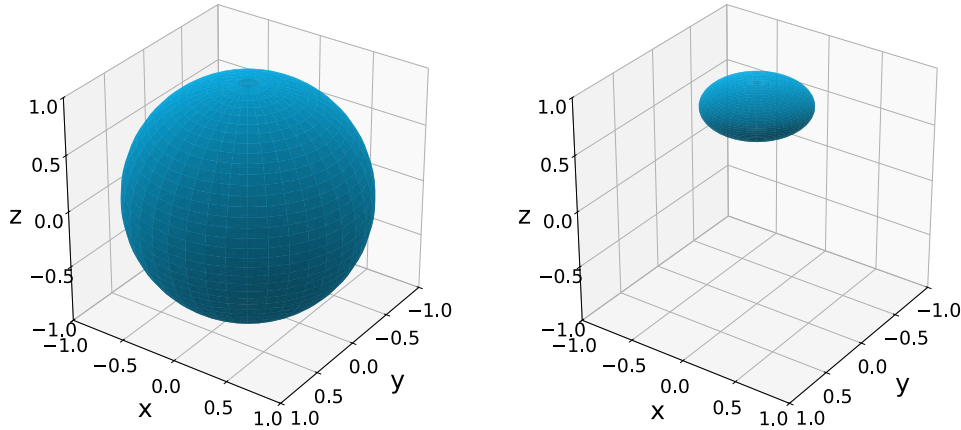


Figure 4.5. The effect of the amplitude damping channel on the Bloch sphere, for $p = 0.8$. Note how the entire sphere shrinks towards the north pole.

4.4.5 Phase damping

Finally, a peculiar noise of the quantum process is *phase damping*. This noise describes the loss of quantum information without loss of energy. Physically it can describe, for example, the dynamics that occurs when a photon randomly scatters as it travels through a waveguide. This quantum noise can be modeled by the *phase damping* channel with the following Kraus operators:

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda} \end{bmatrix}, \quad E_1 = \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{bmatrix}, \quad (4.51)$$

where $\lambda \in [0, 1]$ is the probability of phase damping. The effect of phase damping is the following: the operator E_0 leaves $|0\rangle$ unchanged but reduces the amplitude of the state $|1\rangle$ of a factor $\sqrt{1-\lambda}$; while the operator E_1 destroys the state $|0\rangle$ and reduces the amplitude of the state $|1\rangle$ of a factor $\sqrt{\lambda}$ and does not change it into the state $|0\rangle$.

4.5 Kossakowski-Lindblad-Gorini master equation

So far, we have studied the dynamics of open quantum systems using the quantum operations formalism, which describes discrete state changes without explicit reference to the passage of time. An alternative approach used in physics to model the

dynamics of open quantum systems is via the *continuous-time* description provided by the *master equation*, which can be written more generally in the *Kossakowski-Lindblad-Gorini* form [133, 134, 135] as follows:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar}[H, \rho] + \sum_j \gamma_j \left[L_j \rho L_j^\dagger - \frac{1}{2} \{L_j L_j^\dagger, \rho\} \right], \quad (4.52)$$

where $\{A, B\} = AB + BA$ denotes an anticommutator, H is the Hamiltonian of the system, a Hermitian operator representing the *coherent* part of the dynamics, while $\{L_j\}$ are the *Lindblad operators* which represent the *incoherent* part of the dynamics. The coefficients $\{\gamma_j\}$ are a set of non-negative real numbers. If all $\gamma_i = 0$ one recovers the von Neumann equation:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar}[H, \rho]. \quad (4.53)$$

The derivation and physical approximations to obtain the Lindblad equation from microscopic dynamics are detailed in [99], and will not be reproduced here.

We finally note that solving a master equation provides the time evolution of the density operator of the system. This solution can be expressed in the form of a quantum operation using the Kraus representation:

$$\rho(t) = \sum_k E_k(t) \rho(0) E_k^\dagger(t), \quad (4.54)$$

where $E_k(t)$ are the time-dependent Kraus operator derived from the solution of the master equation. However, not every quantum operation described in this form corresponds to a master equation. This is because the Kraus representation captures state transformations without explicitly modeling the continuous-time dynamics. In particular, it can also describe non-Markovian processes, which involve memory effects and cannot be captured by the Markovian assumptions underlying the Lindblad master equation.

Chapter 5

Quantum Machine Learning

Quantum machine learning is one of the most promising applications of quantum computing. Here, we provide a description of this discipline [28]. This chapter is structured as follows. In Section 5.1, we begin by presenting an overview of quantum machine learning and its four approaches. Next, Section 5.2 describes the main strategies for encoding data into quantum states: *basis encoding*, *amplitude encoding*, and *angle encoding*. In Section 5.3, we introduce *quantum variational circuits*, which represent the quantum analogue of neural networks. In particular, we show how quantum circuits can be interpreted as both deterministic and probabilistic quantum machine learning models. Section 5.4 discusses the training of variational circuits and the techniques for computing partial derivatives by evaluating the circuit at shifted parameter values using the *parameter-shift rule*. Finally, the chapter concludes in Section 5.5 with a discussion on quantum encoding as feature maps and a formal definition of *quantum kernels*.

5.1 Overview and the four approaches

Quantum Machine Learning (QML) is an interdisciplinary research area that arises from the intersection of quantum computing and machine learning. Its objective is to explore how the peculiar properties of quantum theory can improve and enhance the performance of traditional learning algorithms. For example, this synergy is considered particularly promising for the analysis of complex data structures and the solution of high-dimensional problems, such as molecular simulation, optimization problems, and modeling probability distributions, where classical computational methods often face high computational demands or are constrained by limited resources.

The history of QML begins in the early 2000s, with seminal contributions such as the quantum associative memory proposed by Ventura and Martinez [141] and the QBoost algorithm introduced by Hartmut Neven and collaborators [142], implemented on the first commercial quantum annealer developed by D-Wave [143]. However, the term “*quantum machine learning*” came into use only in the mid-2010s, with the manuscript by Lloyd et al. [144], and successively the monograph by Peter Wittek [27], who was instrumental in disseminating and critically challenging the field.

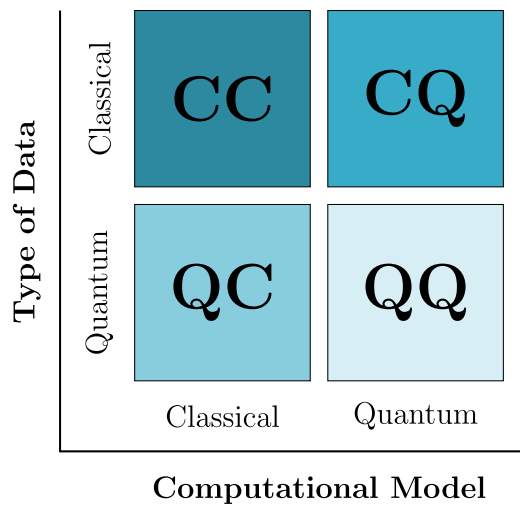


Figure 5.1. The four quantum machine learning approaches that combine the nature of data (classical or quantum) and the type of the computational models (classical or quantum).

Since 2014, the field has attracted increasing interest from both academic and industrial communities, covering different topics [145, 146, 147, 148, 149]. Today, QML is considered a sub-discipline of quantum computing research and is supported by the development of various open-source software frameworks such as PennyLane [150], Qiskit [151], Cirq [152], TensorFlow Quantum [153], and many others.

QML encompasses different approaches. A useful way to describe them is to adopt a classification based on the nature of the data (classical or quantum) and the computational model (classical or quantum). This framework yields four approaches, as shown in Figure 5.1. The first, referred to as CC (Classical-Classical), involves classical models applied to classical data, but includes the use and design of quantum-inspired techniques such as *tensor networks* developed in the context of many-body quantum physics. The second, QC (Quantum-Classical), describes the application of classical algorithms to quantum data, such as quantum states of interest, often derived from experimental measurements, with the aim of extracting information or estimating system properties. The third case, CQ (Classical-Quantum), involves quantum algorithms used to process classical data, such as images, video, or text, with the goal of achieving performance improvements over conventional machine learning methods. Finally, the QQ (Quantum-Quantum) configuration refers to quantum models applied to quantum data, such as quantum states generated by simulations or quantum sensors. This approach allows quantum properties to be exploited in both data and algorithms. Although research in the field of QML is in continuous development, the discipline still faces several challenges [154]. For instance, the limitations of current quantum hardware, in particular the intrinsic noise and the limited number of qubits available, represent a significant obstacle to the reliability and scalability of quantum algorithms. Furthermore, the efficient encoding of classical data into quantum states remains an open challenge, as the resulting computational cost, in some cases, can easily predominate the complexity of quantum algorithms, reducing their advantages.

Nevertheless, quantum machine learning represents one of the most promising frontiers for progress in both artificial intelligence and quantum computing, with potential applicability across diverse domains [155].

5.2 Quantum encoding for classical data

As in the case of classical learning algorithms, data represents the main ingredient in QML routines [156, 157]. In particular, in the CQ approach, representing data using quantum states is the first step in using quantum models to learn from classical data. Quantum encoding plays a crucial role and can often be the bottleneck in the routine of a quantum algorithm. There are several ways to embed the data information in a system of n qubits described by a quantum state. Here we present the most popular strategies: *basis encoding*, *amplitude encoding*, and *angle encoding*.

5.2.1 Basis encoding

This is the most direct encoding strategy as each classical bit is replaced by a qubit. Suppose that we want to encode an n -bit binary string of the form $\mathbf{b} = (b_0, b_1, \dots, b_{n-1})^T \in \{0, 1\}^n$. Then, n qubits are required for the encoding: each bit b_i is mapped to the computational basis state $|b_i\rangle$, where 0 and 1, respectively, correspond to the states $|0\rangle$ and $|1\rangle$. The resulting quantum state of the binary string $\mathbf{b}$ is given by:

$$|\mathbf{b}\rangle = |b_0 b_1 \dots b_{n-1}\rangle, \quad (5.1)$$

where each $b_i \in \{0, 1\}$ for $i = 0, 1, \dots, n-1$. This state is obtained by applying the following transformation to the initial state $|0\rangle^{\otimes n}$:

$$U(\mathbf{b}) = \prod_{i=1}^n X_i^{b_i}, \quad (5.2)$$

where $X_i^{b_i}$ denotes the application of the Pauli- X gate to the i -th qubit if $b_i = 1$, and the identity otherwise.

This strategy has the advantage of allowing a deterministic recovery of the original classical information. In fact, by performing measurements on the computational basis of the states, one can obtain the classical bit string $\mathbf{b}$. However, its drawback is that it requires a large number of qubits, and for large n , it can be unfeasible to implement it on current QPUs.

5.2.2 Amplitude encoding

Amplitude encoding is the basis for several QML algorithms. This strategy incorporates classical data into quantum states by mapping the components of a vector to the amplitudes of a quantum state. Specifically, given a classical vector of real values $\mathbf{x} = (x_0, x_1, \dots, x_{2^n-1})^T \in \mathbb{R}^{2^n}$, it is encoded in a quantum state of the form:

$$|\mathbf{x}\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle, \quad (5.3)$$

where $|j\rangle$ are the computational basis states of a system of n qubits, and the normalization condition $\sum_{j=0}^{2^n-1} |x_j|^2 = 1$ is satisfied.

The main advantage of this strategy is that it allows one to represent 2^n real (or complex) values by using only n qubits, thus offering an *exponential* reduction

in computational resources in terms of qubits with respect to the data dimension. However, the main disadvantage of amplitude encoding is its circuit complexity. Indeed, preparing a general quantum state with arbitrary amplitudes typically requires a number of quantum gates that grows exponentially with the number of qubits. Consequently, amplitude encoding circuits often have a depth too high to be reliably executed on current quantum devices. Furthermore, encoding information in the probabilistic amplitudes of a quantum system entails limitations in information recovery. In practice, a large number of measurements are required to faithfully reconstruct the encoded data.

5.2.3 Angle encoding

Angle encoding is a strategy for embedding classical data into quantum states using rotation quantum gates to encode scalar values into the rotation angles of qubits. Given a classical vector $\mathbf{x} = (x_0, x_1, \dots, x_{n-1})^T \in \mathbb{R}^n$, each component x_i is associated with a rotation angle applied to the i -th qubit. A common approach is to use Pauli rotation gates R_x , R_y , or R_z , which rotate the qubit around the X , Y , or Z axis of the Bloch sphere, respectively. For example, using R_y encoding, the corresponding quantum state of $\mathbf{x}$ is given by:

$$|\psi\rangle = \bigotimes_{i=0}^{n-1} R_y(x_i) |0\rangle. \quad (5.4)$$

This results in a quantum state where the amplitude of each qubit depends on the corresponding input feature. This strategy is commonly used in variational quantum algorithms and hybrid quantum-classical learning models.

5.3 Variational quantum circuits as learning models

Near-term quantum computing represents a key resource for the development of QML, as it allows us to address real-world problems with currently available resources. However, limitations imposed by simulators and quantum hardware, such as the use of a small number of qubits and logic gates due to the presence of noise, impose constraints on the design of efficient algorithms. To address these challenges, research has focused on two complementary approaches. The first involves the design of *hybrid quantum-classical algorithms*, in which the machine learning model is implemented as a quantum circuit, while optimization and training are performed by a classical coprocessor. The second approach aims to exploit the topology of quantum hardware to reduce or mitigate the effects of intrinsic noise, thus improving the robustness and efficiency. In both cases, Variational Quantum Circuits (VQCs) [158] (also known as Parameterized Quantum Circuits (PQCs) [159] or, in the context of QML, Quantum Neural Networks (QNNs) [160]) are particularly well-suited for implementation on current quantum devices in different applications [161, 162, 163, 164]. Similarly to classical neural networks, where the architecture is defined by the arrangement of layers, a variational circuit is composed of an *ansatz* of fixed and parameterized quantum gates. The free parameters are optimized by minimizing a cost function, thus adapting the behavior of the circuit to the specific problem. In the following,

we will present these circuits as deterministic and probabilistic quantum learning models.

5.3.1 Deterministic quantum learning models

In Chapter 1, we have defined a deterministic machine learning model as a function f_θ that maps elements from the input data space $\mathcal{X}$ to the output domain $\mathcal{Y}$. In the context of QML, a similar definition can be formulated using VQCs.

Definition 7 (Deterministic quantum learning model). *Let $\mathcal{X}$ be an input data domain and let $U(x, \theta)$ be a quantum circuit that depends on both the input x and a set of trainable parameters $\theta \in \mathbb{R}^K$, and let M be a Hermitian operator representing a quantum observable. Furthermore, let us consider a quantum state prepared by applying the circuit to the initial state $|0\rangle$ is denoted by $|\psi(x, \theta)\rangle = U(x, \theta)|0\rangle$. The model output is defined as the expectation value of the observable M as follows:*

$$f_\theta(x) = \langle \psi(x, \theta) | M | \psi(x, \theta) \rangle. \quad (5.5)$$

Alternatively, using the density matrix formalism, we define

$$f_\theta(x) = \text{tr} \{ M \rho(x, \theta) \}, \quad (5.6)$$

where $\rho(x, \theta) = U(x, \theta) |0\rangle \langle 0| U(x, \theta)^\dagger$.

This construction defines a deterministic quantum learning model. For brevity, we will sometimes abbreviate the model output as $f_\theta(x) = \langle M \rangle_{x, \theta}$. A commonly adopted structure for the unitary circuit $U(x, \theta)$ is illustrated in Figure 5.2. In this scheme, the circuit is divided into two main components: a data embedding block $S(x)$, which encodes the input $x \in \mathbb{R}^N$ in the quantum state, and a parametric block $W(\theta)$, which depends on the trainable parameters $\theta \in \mathbb{R}^K$. The composition of these two blocks defines the complete circuit as follows:

$$U(x, \theta) = W(\theta)S(x). \quad (5.7)$$

Both blocks can be further decomposed into sequences of quantum gates that act on individual components of the input or parameters. In particular, the embedding block can be expressed as:

$$S(x) = T_{N+1} \prod_{i=1}^N S_i(x_i) T_i, \quad (5.8)$$

and the parametric block as:

$$W(\theta) = V_{K+1} \prod_{k=1}^K W_k(\theta_k) V_k, \quad (5.9)$$

where $S_i(x_i)$ and $W_k(\theta_k)$ are gates encoding the i -th input element and the k -th parameter, respectively, while T_i and V_k are fixed unitary operators that do not depend on data or parameters.

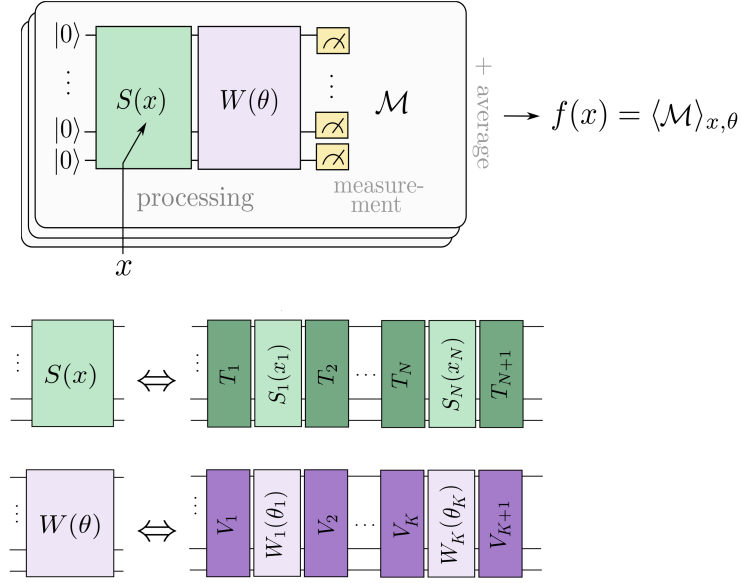


Figure 5.2. On the top, illustration of a quantum variational circuit $U(x, \theta) = W(\theta)S(x)$ as a deterministic learning model. On the bottom, decomposition of the data encoding $S(x)$ and the parametrized block $W(\theta)$ (Adapted from [28]).

5.3.2 Probabilistic quantum learning models

A probabilistic machine learning model is formally described as a parameterized probability distribution $p_\theta(x)$ (see Chapter 1). Since quantum theory is inherently probabilistic, it offers a natural framework for building probabilistic learning models based on quantum circuits. Next, we provide a formal formulation in the context of unsupervised learning. For a definition of probabilistic models in supervised learning scenarios, see [28]. A quantum variational circuit defines a quantum parametric model as follows:

Definition 8 (Unsupervised probabilistic quantum learning model). *Let $\mathcal{X}$ be an input domain, $W(\theta)$ be a parametrized unitary transformation such that $|\psi(\theta)\rangle = W(\theta)|0\rangle$, and $M = \sum_{x \in X} x|x\rangle\langle x|$ be a diagonal measurement with outcomes corresponding to the inputs x . An unsupervised probabilistic quantum model is defined by the distribution:*

$$p(x) = |\langle x | \psi(\theta) \rangle|^2. \quad (5.10)$$

Note that the construction of probabilistic quantum learning models differs from that of deterministic quantum learning models in that the sampled data are associated with the measurement results. Furthermore, these models are naturally generative, as their implementation on a quantum computer produces samples directly from the $p(x)$ distribution through measurement operations. An example of a probabilistic quantum model is shown in Figure 5.3.

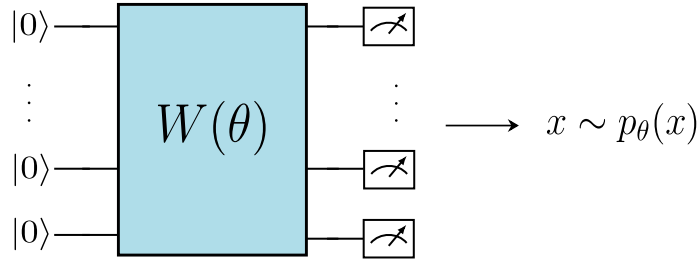


Figure 5.3. A variational quantum circuit as a probabilistic learning model. The data samples x from a distribution $p_\theta(x)$ are obtained performing single measurements on the input data basis $\{|x\rangle\}$, with $x \in \mathcal{X}$.

5.4 Training

As described in Section 1.3.3, training a neural network means finding the set of model parameters θ that minimize a data-dependent cost function. This is realized through an optimization procedure that typically makes use of gradient-based techniques such as stochastic gradient descent and its variants.

To apply these techniques in a multi-layer neural network, it is essential to calculate the gradients of the parameters with respect to the cost function. This is made possible by error backpropagation, known in its most general form as *automatic differentiation*. From a formal perspective, automatic differentiation is an application of the chain rule. Consider a cost function $C(f_\theta)$ depending on a model f that in turn depends on a set of parameters θ . The partial derivative of the cost function with respect to a parameter $\mu \in \theta$ can be written as follows:

$$\frac{\partial C(f_\theta)}{\partial \mu} = \frac{\partial C}{\partial f_\theta} \frac{\partial f_\theta}{\partial \mu}. \quad (5.11)$$

The calculation of these partial derivatives is usually performed using machine learning libraries such as *PyTorch* and *TensorFlow*.

Variational quantum models also aim to minimize a cost function with respect to the model parameters θ , thus the same optimization procedure can be adopted. However, because f_θ results from a quantum computation, the term $\frac{\partial f_\theta}{\partial \mu}$ cannot be directly computed using classical libraries. Therefore, we need to find a way to compute gradients in the quantum context, which will be described in the following paragraph. Once such partial derivatives are available, quantum computing can be integrated into hybrid machine learning pipelines.

Finite difference

A first simple approach to numerically approximate the partial derivative of a quantum model $\frac{\partial f_\theta}{\partial \mu}$ is to use the finite-difference method for which:

$$\frac{\partial f_\theta}{\partial \mu} \approx \frac{f_{\theta+\Delta\theta} - f_\theta}{\Delta\mu}, \quad (5.12)$$

where $\Delta\theta$ denotes the set of parameters where μ has been replaced by $\mu + \Delta\mu$, with $\Delta\mu$ representing an infinitesimal shift. This yields a linear approximation of the model between two distinct points.

However, finite-difference methods can be problematic in contexts where each function evaluation can only be computed with an error. In particular, the smaller the gradient, the greater the precision required to accurately estimate the cost function, which in turn requires multiple repetitions of the algorithm [165]. Moreover, numerical finite-difference techniques are particularly challenging in scenarios where the optimization landscape contains many saddle points and where the algorithm produces measurements with high variance. These limitations make finite-difference methods not suitable for near-term quantum learning algorithms.

Parameter shift rule

The parameter shift rule is a technique that allows the partial derivative of a quantum expectation value with respect to a gate parameter to be calculated, expressed as a linear combination of the same expectation value evaluated with shifted parameter values:

$$\frac{\partial f_\mu}{\partial \mu} = \sum_i a_i f_{\mu+s_i} \quad (5.13)$$

where $f_\mu = \langle M \rangle_\mu$ and $\{a_i\}, \{s_i\}$ are real scalars.

Unlike finite-difference methods, this approach produces *exact gradients* rather than approximations, and the shifts s_i are not necessarily infinitesimal. In practice, however, since quantum computers can only estimate expectation values, parameter-shift rules provide an *estimation of the analytic gradient*, while finite differences provide an *estimate of an approximate gradient*. The breakthrough in quantum gradient computation was achieved in the paper by Mitarai et al. [166], which introduced the most prominent form of the parameter-shift rule. In its concise formulation, the rule applies to gates $\mathcal{G}(\mu)$ generated by a Hamiltonian G with $G^2 = I$ such that:

$$\mathcal{G}(\mu) = e^{-i\mu G} = \cos(\mu)I - i\sin(\mu)G. \quad (5.14)$$

Examples are Pauli rotations and their tensor products, which are the building blocks of parametrized circuits. Let $f_\mu = \langle \psi | \mathcal{G}^\dagger(\mu) \mathcal{B} \mathcal{G}(\mu) | \psi \rangle$, with $\mathcal{B} = V^\dagger \mathcal{M} V$, be a deterministic quantum model, after a bit of algebra and using trigonometric identities [28, 167], the partial derivative of f_μ can be expressed as follows:

$$\frac{\partial f_\mu}{\partial \mu} = \frac{1}{2\sin(s)} (f_{\mu+s} - f_{\mu-s}), \quad (5.15)$$

which is the standard two-term parameter shift rule.

In practice, s is often chosen as $\pi/2$, such that the multiplicative factor is equal to $1/2$. The finite-difference rule emerges as the limit $s \rightarrow 0$. In addition, the parameter shift rule can be extended to compute higher-order derivatives [167], and it is still possible to derive a formulation of the parameter shift rule, even when gates do not satisfy the condition $G^2 = I$ [168].

From a computational resource perspective, for example, the standard two-term parameter shift formulation requires a number of circuit evaluations equal to $2KS$, where K is the number of parameters and S is the number of shots needed to estimate a single expectation value. During the early stages of training, S can be chosen very small, even equal to 1, since quantum expectation provides an unbiased estimator of the gradient. This ensures that repeated low-shot evaluations can still drive convergence to the desired minimum [169]. The main challenge concerns the scalability with respect to K . In fact, each of the K gradient components needs to be computed independently, which quickly becomes computationally expensive. In contrast, classical neural networks leverage automatic differentiation and backpropagation, reusing intermediate results to obtain the full gradient in approximately a single model evaluation.

Finally, although in principle it is possible to optimize variational quantum models using parameter-shift methods and deep learning tools, the cost function landscape may not be favorable for the training process. In fact, the landscape of the cost function may contain regions known as *barren plateaus* in which all the gradient components are nearly zero, making optimization extremely challenging. For a more detailed discussion, see [28, 170, 171].

5.5 Quantum kernels

As discussed in Section 1.4.3, feature maps play a central role in the kernel method approach to solving machine learning tasks. The basic idea of kernel methods is to embed data into high-dimensional feature spaces, where it may become linearly separable, and to compute the inner products of the resulting feature vectors. Then, these inner products can be interpreted as similarity measures between data points. In Section 5.2, we introduced quantum embedding strategies for classical data. Here, we show how these strategies can be interpreted as feature maps. For a more complete overview and description on the quantum kernels, see [172, 173, 28].

Consider a data encoding circuit $S(x)$ of a classic data sample $x \in \mathcal{X}$ from a domain $\mathcal{X}$, such that $|\phi(x)\rangle = S(x)|0\rangle$. The data-encoding feature map is defined as follows:

Definition 9 (Data-encoding feature map). *Given an n -qubit quantum system with states $|\psi\rangle$, let $\mathcal{F}$ denote the space of complex-valued (density) matrices of dimension $2^n \times 2^n$, equipped with the Hilbert–Schmidt inner product $\langle \rho, \sigma \rangle_F = \text{tr}\{\rho^\dagger \sigma\}$ with $\rho, \sigma \in \mathcal{F}$. The data-encoding feature map is defined as the transformation*

$$\phi : \mathcal{X} \rightarrow \mathcal{F}, \quad \phi(x) = |\phi(x)\rangle\langle\phi(x)| = \rho(x), \quad (5.16)$$

which is implemented by a quantum data-encoding circuit $S(x)$.

We recall that kernels are defined as positive semi-definite real-valued functions $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. For every such function, there exists at least one feature map $\phi : \mathcal{X} \rightarrow \mathcal{F}$ in a Hilbert space of features $\mathcal{F}$ such that the kernel can be expressed as the inner product of feature vectors:

$$\kappa(x, x') = \langle \phi(x'), \phi(x) \rangle_{\mathcal{F}}. \quad (5.17)$$

In addition, each feature map induces a kernel through the inner product in its corresponding feature space. Thus, from Definition 9 we easily obtain the kernel associated with the data-encoding feature map that we refer to as the *quantum kernel*.

Definition 10 (Quantum Kernel). *Let ϕ be a data-encoding feature map over the domain $\mathcal{X}$. The quantum kernel is defined as*

$$\kappa(x, x') = \text{tr}\{\rho(x')\rho(x)\} = |\langle \phi(x') | \phi(x) \rangle|^2. \quad (5.18)$$

To show that it is a kernel, it is necessary to show that the quantum kernel is positively definite. Since the product of two kernels is itself a kernel, this can be proven by showing that the complex conjugate of a kernel is also a kernel. For any $x_m \in \mathcal{X}$, $m = 1, \dots, M$, and for any $c_m \in \mathbb{C}$, we have

$$\sum_{m, m'} c_m c_{m'}^* \kappa_c(x^m, x^{m'})^* = \sum_{m, m'} c_m c_{m'}^* \langle \phi(x^m) | \phi(x^{m'}) \rangle \langle \phi(x^m) | \phi(x^{m'}) \rangle \quad (5.19)$$

$$= \left(\sum_m c_m \langle \phi(x^m) | \right) \left(\sum_{m'} c_{m'} \phi(x^{m'}) \right) \quad (5.20)$$

$$= \left\| \sum_m c_m |\phi(x_m)\rangle \right\|^2 \quad (5.21)$$

$$\geq 0, \quad (5.22)$$

which means that the complex conjugate of a kernel is also positive definite.

Chapter 6

Quantum Encoding for Image Classification

This chapter is an adapted version of the article “Analysis of Quantum Image Representations for Supervised Classification” published in AVS Quantum Science and is part of the research conducted during my doctoral studies [174]. In this study, we analyze the compression properties and investigate the trade-off between accuracy and storage cost in binary classification, considering four different quantum embeddings for classical 2D images. The chapter is structured as follows. We begin in Section 6.1 by describing the motivations of our work. Next, Section 6.2 describes the four quantum representations for grayscale images used in our study: *Tensor Network* [175], *Flexible Representation of Quantum Images* [176], *Novel Enhanced Quantum Representation* [177], and *Quantum Probability Image Encoding* [178]. Finally, in Section 6.3 we present and discuss our findings.

6.1 Background and motivation

We live in the era of big data and artificial intelligence. The growing volume of data and the need to solve increasingly complex computational problems are two driving forces to improve the efficiency of data storage, processing, and analysis. Quantum Image Processing (QIP) is an interdisciplinary field at the intersection of quantum information science and image processing with the potential to alleviate some of these challenges by exploiting the peculiar properties of quantum systems, such as superposition and entanglement [179, 180, 181].

On a classical computer, digital images are represented as matrices whose elements encode discrete color (or intensity) values for each pixel. Therefore, an image contains two types of information for each pixel: position and color. More precisely, position is encoded implicitly via matrix indices, while color is encoded explicitly as matrix values. Quantum Image Representation (QImR) is a subfield of QIP that studies how to efficiently encode image information in quantum states. In general, given a classical image I of size $N = 2^n \times 2^n$, it can be encoded in a quantum state as follows:

$$|I\rangle = \frac{1}{\sqrt{N}} \sum_{P=0}^{N-1} |C_P\rangle \otimes |P\rangle, \quad (6.1)$$

where the two quantum states $|C_P\rangle$ and $|P\rangle$ encode, respectively, the color and position of a pixel. This superposition state enables a reduction in storage cost. For example, a black and white image with N pixels requires N bits to be stored on a classical computer, while on a quantum device it requires only $\log N$ qubits. This means an exponential reduction in storage resources [178]. The two quantum states $|C_P\rangle$ and $|P\rangle$ in Eq. 6.1 can encode the intensity and position of a pixel using the quantum embeddings described in Section 5.2.

Compact representations are often associated with improved generalization performance in classical machine learning. In fact, there exists a connection between learning and compression, which is explored in classical deep learning [182] and quantum learning [183, 184]. In addition, recent studies have demonstrated that hybrid quantum–classical architectures can leverage these compression principles to improve performance in image classification tasks [185].

Quantum image representations have recently been studied and tested on NISQ devices [186]. For example, Das et al. [121] implemented a quantum pattern recognition protocol on a real QPU using the Quantum Probability Image Encoding (QPIE) representation [178]. In parallel, Geng et al. [187] examined the feasibility of implementing QImRs on superconducting and trapped-ion quantum platforms, successfully implementing a 2×2 Flexible Representation of Quantum Images (FRQI) [176] image. In addition to these hardware-based implementations, quantum image representations based on tensor networks have emerged as a promising alternative. Such methods enable efficient compression and encoding of image data in a form naturally suited to quantum circuit architectures. For example, Stoudenmire et al. [188] demonstrated that quantum-inspired tensor networks can be effectively applied to supervised learning tasks, particularly image classification, by reducing dimensionality while conserving essential data features.

We showed in Section 5.5 that quantum embeddings can be interpreted as feature maps, allowing the application of kernel theory in a quantum framework. In particular, kernels constructed through quantum circuits have the potential to enhance and improve computational performance in classification tasks [189, 190, 191, 192, 193, 194].

Accordingly, our study aims to compare and analyze the compression properties of four different quantum image representations that incorporate pixel information (color and position) in different ways. We also investigate the tradeoff between accuracy and storage costs in binary classification by evaluating QImR-based kernels for the four different quantum representations against a classical linear kernel.

6.2 Quantum encoding for images

In this section, we introduce four types of quantum image representations that encode the color and position of pixels, discussing their advantages and disadvantages.

6.2.1 Tensor network representation

The Tensor Network Representation (TNR) is based on the *real-ket* encoding proposed by José I. Latorre [175] in 2005. In real-ket encoding, the pixel values of an image are mapped to the real amplitudes of a pure quantum state using a 4-block

$i_1 = 1$	$i_1 = 2$	$i_1 = 1$	$i_1 = 2$
$i_2 = 1$		$i_2 = 2$	
$i_1 = 3$	$i_1 = 4$	$i_1 = 3$	$i_1 = 4$
$i_2 = 3$		$i_2 = 4$	
$i_1 = 3$	$i_1 = 4$	$i_1 = 3$	$i_1 = 4$

Figure 6.1. An example of hierarchical indexing for real-ket representation of an image with $2^2 \times 2^2$ pixels.

structured scheme. More precisely, given an image of size $2^n \times 2^n$, it can be expressed as a real-amplitude ket as follows:

$$|\psi_{2^n \times 2^n}\rangle = \sum_{i_1=1}^4 \cdots \sum_{i_n=1}^4 c_{i_n, \dots, i_1} |i_n \cdots i_1\rangle, \quad (6.2)$$

where $c_{i_n, \dots, i_1}$ encodes the value of the pixel indexed by the coordinates $i_n, \dots, i_1$ in the block structure. Specifically, an image is iteratively reduced into four blocks, each indexed with an integer $i = 1, \dots, 4$. The position of each pixel can be expressed using n integers with values between 1 and 4. In particular, each sub-block is assigned an index based on its position: $i_n = 1$ corresponds to the top-left pixel, $i_n = 2$ to the top-right pixel, $i_n = 3$ to the bottom-left pixel, and $i_n = 4$ to the bottom-right pixel. To specify which sub-block is addressed within the larger structure, we introduce an additional label, i_{n-1} , following the same convention within the internal sub-blocks. This procedure is performed n times until each sub-block consists of a single pixel. This hierarchical labeling ensures a consistent and scalable representation of the pixel organization. An example of TNR encoding for an image with $n = 2$ is shown in Figure 6.1.

This formulation allows the use of mathematical expressions to describe an image as a quantum state, where the tensors encode the relationships between pixels. Indeed, once the image is mapped into a quantum state, a particularly useful framework for compressing it is via Tensor Networks (TNs) methods or, specifically, Matrix Product States (MPSs) [195]. In particular, MPS provides a compact way to encode an image by organizing pixel information into an array of tensors that efficiently capture the entanglement between different regions. With this approach, the MPS state can be expressed mathematically as follows:

$$|\psi_\Gamma\rangle = \sum_{\{i_j\}, \{\lambda_j\}} \Gamma_{\lambda_1 \lambda_2}^{(1) i_1} \Gamma_{\lambda_2 \lambda_3}^{(2) i_2} \cdots \Gamma_{\lambda_n \lambda_1}^{(n) i_n} |i_n, \dots, i_2, i_1\rangle, \quad (6.3)$$

where $\Gamma^{(j)}$ are tensors with real elements. The advantage of MPS lies in its ability to represent images using fewer parameters, making it an efficient tool for compression

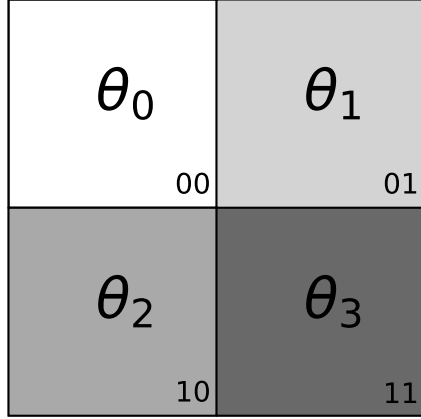


Figure 6.2. A 2×2 FRQI image. The angles θ_i , with $i = 0, 1, 2, 3$, at the center of each pixel encode the gray intensity of the corresponding pixels in the angular representation. The 2-bit strings in the lower-right corner identify the positions in the image in binary representation.

and structured encoding. Finally, from the tensors Γ in Eq. 6.3, it is possible to obtain a quantum circuit of $2n + 1$ qubits to create the same state on a NISQ device [196].

6.2.2 Flexible Representation of Quantum Images

The Flexible Representation of Quantum Images (FRQI) was proposed by Le et al. [176] in 2010. This encoding takes inspiration from the pixel representation of images in classical computers and uses quantum superposition to encode a classical image in the form of a normalized quantum state. Formally, FRQI encodes a $2^n \times 2^n$ grayscale image in a quantum state as follows:

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} |c(\theta_i)\rangle \otimes |i\rangle, \quad (6.4)$$

where the color-state,

$$|c(\theta_i)\rangle = \cos \theta_i |0\rangle + \sin \theta_i |1\rangle. \quad (6.5)$$

Here, $|0\rangle$ and $|1\rangle$ are the two-dimensional quantum states of the computational basis, $|i\rangle$ is a sequence of $2n$ qubits storing the position of the pixel, and $\theta_i \in [0, \frac{\pi}{2}]$ is the angle that encodes the color of the i -th pixel. The implementation [176] of a FRQI circuit begins by applying Hadamard gates, H , to the position qubits to create a uniform superposition. Then, for each pixel, a multi-controlled rotation gate, $C^{2n}(R_y(\theta))$, is applied to the color qubit, where the rotation angle θ depends on the intensity of the pixel. An example of a 2×2 FRQI image is illustrated in Figure 6.2. Its main advantage is that FRQI requires only $2n + 1$ qubits to represent a grayscale image on a quantum device, exponentially reducing memory resources compared to the classical case ($\mathcal{O}(2^{2n})$). In particular, $2n$ qubits are used to encode the positions of pixels, while a single qubit stores the grayscale information of all pixels. Furthermore, this model provides efficient color operations [197] and fast geometric transformations [198]. The two main FRQI drawbacks are image retrieval and the depth of the quantum circuit. The first is due to the color encoding of each

pixel that is in the probability amplitudes of a qubit, and consequently an infinite number of measurements are necessary to exactly reconstruct the initial image. Thus, the original image can only be retrieved probabilistically. The second drawback is that the preparation of the FRQI states is based on the polynomial preparation theorem [176] and has a computational complexity of $\mathcal{O}(2^{4n})$. However, this last drawback has recently been overcome by Nasr et al. [186] who introduced Enhanced Flexible Representation of Quantum Images (EFRQI) reducing the computational complexity to $\mathcal{O}(2n2^{2n})$.

6.2.3 Novel Enhanced Quantum Representation

To overcome the image retrieval limitation of FRQI, Zhang et al. [177] proposed Novel Enhanced Quantum Representation (NEQR). More precisely, NEQR encodes the digital image into a superposition of two entangled sequences of qubits: one stores the grayscale and the other the position information. Formally, given a $2^n \times 2^n$ image with 2^q gray intensities, it is embedded in a quantum state of the form:

$$|I\rangle = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} |f(y, x)\rangle \otimes |yx\rangle. \quad (6.6)$$

Here, $f(y, x) \in [0, 2^q - 1]$ is a q -bit sequence $C_{yx}^0 \dots C_{yx}^{q-2} C_{yx}^{q-1}$ encoding the grayscale value of the yx pixel:

$$f(y, x) = C_{yx}^0, C_{yx}^1, \dots, C_{yx}^{q-2}, C_{yx}^{q-1}, \quad C_{yx}^k \in [0, 1],$$

and the state,

$$|yx\rangle = |y_0 y_1 \dots y_{n-1}\rangle \otimes |x_0 x_1 \dots x_{n-1}\rangle,$$

encodes the position of the pixel yx . NEQR requires $2n + q$ qubits to store an image on a quantum computer. Specifically, $2n$ qubits are used to encode pixel positions and q qubits are used to store grayscale intensity values. An example of a 2×2 NEQR image is shown in Figure 6.3. The implementation [177] of a NEQR

11110000 00	01000100 01
10010100 10	01001001 11

Figure 6.3. A 2×2 NEQR image. The 8-bit strings in the center of each pixel encode the gray intensity, and the 2-bit strings in the lower right corner encode the pixel positions in the image.

Quantum Encoding	Color Encoding	Qubit Resource	Complexity	Retrieval
TNR	Tensor	$2n + 1$	–	Probabilistic
FRQI	1 Angle	$2n + 1$	$\mathcal{O}(2^{4n})$	Probabilistic
NEQR	Basis States	$2n + q$	$\mathcal{O}(qn2^{2n})$	Deterministic
QPIE	Amplitudes	$2n$	$\mathcal{O}(2^n)$	Probabilistic

Table 6.1. Comparison of key parameters and characteristics of four quantum image representations, TNR, FRQI, NEQR, and QPIE, for encoding images of size $2^n \times 2^n$ with grayscale values in the range $[0, 2^q - 1]$.

circuit begins by applying Hadamard gates H to the position qubits, creating a uniform superposition of all pixel positions. Then, multi-controlled NOT quantum gates $C^{2n}(\text{NOT})$ are applied to encode the grayscale intensity information into the q color qubits, conditioned on the corresponding pixel positions. The original classical image can be accurately retrieved through quantum measurements on the basis states. Moreover, NEQR provides a quadratic speedup in quantum image preparation compared to FRQI, with a computational complexity of $\mathcal{O}(2qn2^{2n})$. Recently, Nasr et al. [186] introduced Enhanced Novel Enhanced Quantum Representation (ENEQR), which further reduces the complexity to $\mathcal{O}(2n2^{2n})$.

6.2.4 Quantum Probability Image Encoding

Quantum Probability Image Encoding (QPIE) was proposed by Yao et al. [178] in 2017. This encoding uses the probability amplitudes of pure quantum states to store the grayscale values of pixels. Mathematically, given a grayscale image of size $2^n \times 2^n$, the corresponding quantum state is as follows:

$$|I\rangle = \sum_{i=0}^{2^{2n}-1} c_i |i\rangle, \quad c_i = \frac{I_i}{\sqrt{\sum_{j=0}^{2^{2n}-1} I_j^2}}, \quad (6.7)$$

where I_i denotes the color intensity of the pixel in the position i , and c_i is the corresponding normalized intensity. The QPIE only uses $2n$ qubits to be represented on a quantum computer. This model thus requires fewer qubits than the previous quantum encoding schemes, namely FRQI and NEQR. However, because pixel values are stored in the amplitude, QPIE has the same drawback as FRQI about the retrieval of the original image from a finite number of measurements. Implementing a QPIE circuit is equivalent to implementing an amplitude embedding. The preparation of a pure quantum state of n qubits with a specified probability distribution requires computational complexity $\mathcal{O}(2^n)$ [199]. The method proposed by Möttönen et al. [200] provides an efficient construction of such circuits through uniformly controlled rotations. An example of a 2×2 QPIE image is illustrated in Figure 6.4.

Finally, the main properties and characteristics of the quantum encoding described above are summarized in Table 6.1.

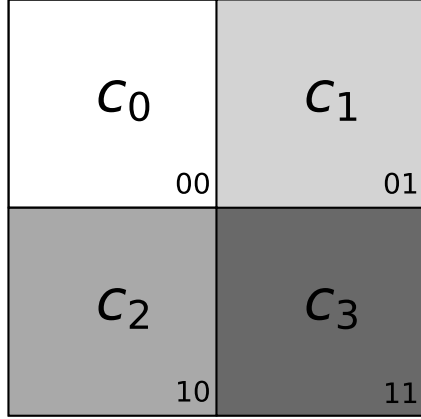


Figure 6.4. A 2×2 QPIE image. The amplitudes c_i , with $i = 0, 1, 2, 3$, at the center of each pixel encode the gray intensity of the corresponding pixels, while the 2-bit strings in the lower-right corner label the pixel positions in the image in binary representation.

6.3 Our results

In this section, we present our original study. We start by describing the simulation setup. We then derive the analytical expression for the quantum kernels for the four representations that we use for compression analysis and SVM-based classification. Finally, we show and discuss our new results.

Our simulations are performed using PennyLane [150] and Scikit-Learn [201], which are open-source software for quantum and classical machine learning, respectively. PennyLane is used for implementing quantum circuits that embed classical images into quantum states, while Scikit-Learn is used for performing the SVM-based classification. In our analysis, we adopt two popular datasets: MNIST [30] and Fashion MNIST [31], whose samples are illustrated in Figure 6.5. All image samples are pre-processed before being encoded in quantum states. In particular, the size of each image is reduced to 16×16 using the module *cv2.INTER_AREA* of OpenCV [202] and the color of each pixel is rescaled to $[0, 255]$. Binary classification is performed using a SVM with a training set and a test set consisting of 10^3 and 10^2 images, respectively. A summary describing the simulation parameters is reported in Table 6.2.

Before proceeding with the analytical calculation of the QImRs-based quantum kernels, which will be used in the simulations of the SVM model, we recall the definition in the noiseless scenario and its relationship to quantum fidelity (see Section 4.2.5). Given two classical images I and I' , encoded in two pure quantum states $|I\rangle$ and $|I'\rangle$, respectively, the quantum kernel is defined as:

$$K_{I,I'} = |\langle I|I'\rangle|^2, \quad (6.8)$$

where $\langle \cdot | \cdot \rangle$ denotes the scalar product in the Hilbert space. This quantity corresponds to the quantum fidelity between the two quantum states and measures the similarity between the encoded images. In particular, it holds:

- $K_{I,I'} = 1$ if and only if $|I\rangle = |I'\rangle$ that means the images are identical;
- $K_{I,I'} = 0$ if the states are orthogonal, meaning completely distinguishable;

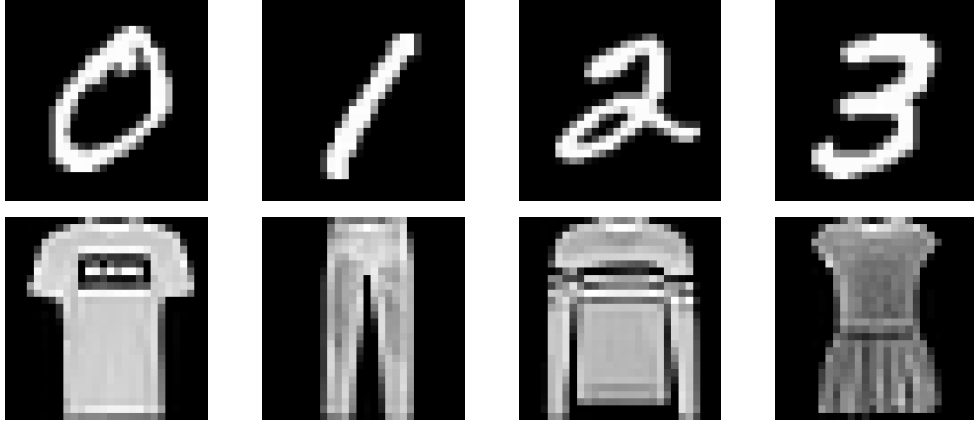


Figure 6.5. Image samples of MNIST (first row) and Fashion MNIST (second row) datasets.

Component	Description
Datasets	10^3 (training) and 10^2 (test) images from MNIST and Fashion-MNIST datasets.
Task	Binary classification: class 0 vs. classes 1–9.
Preprocessing	Grayscale images resized to 16×16 pixels.
Model	Support Vector Machine (SVM) with a pre-computed kernel, implemented using the <code>Scikit-learn</code> library with default parameters.
Validation	K-fold cross-validation.

Table 6.2. Implementation details for SVM-based binary classification using quantum image representations kernel and classical linear kernel [174].

- intermediate values $0 < K_{I,I'} < 1$ quantify the degree of overlap between the two quantum states.

The inner product between two MPS states as in Eq. 6.3 is given by:

$$\langle \psi_{\Gamma} | \psi_{\Gamma'} \rangle = \sum_{\{i_k\}, \{\lambda_k\}, \{\beta_k\}} \prod_k \Gamma_{\lambda_k \lambda_{k+1}}^{(k) i_k *} \Gamma'_{\beta_k \beta_{k+1}}^{(k) i_k}, \quad (6.9)$$

and can be efficiently calculated using standard contraction techniques [203]. By defining:

$$E_k \equiv \sum_{i_k} \Gamma_{\lambda_k \lambda_{k+1}}^{(k) i_k *} \Gamma'_{\beta_k \beta_{k+1}}^{(k) i_k}, \quad k = 1, 2, \dots, n, \quad (6.10)$$

the TNR-kernel between two MPS states can be written in compact form as:

$$|\text{Tr}(E_1 E_2 \cdots E_n)|^2. \quad (6.11)$$

For the FRQI case, given two quantum states as in Eq. 6.4, the inner product between them is as follows:

$$\begin{aligned}\langle I(\theta)|I(\theta')\rangle &= \frac{1}{2^{2n}} \sum_{i,j=0}^{2^{2n}-1} \langle c(\theta_i)|c(\theta_j)\rangle \otimes \langle i|j\rangle \\ &= \frac{1}{2^{2n}} \sum_{i,j=0}^{2^{2n}-1} (\cos \theta_i \cos \theta'_i + \sin \theta_i \sin \theta'_i) \\ &= \frac{1}{2^{2n}} \sum_{i=0}^{2^{2n}-1} \cos(\theta_i - \theta'_i),\end{aligned}$$

where we use the orthonormal property of the basis state $\langle i|j\rangle = \delta_{i,j}$. Thus, the FRQI-kernel is given by:

$$K_{I,I'} = \left| \frac{1}{2^{2n}} \sum_{i=0}^{2^{2n}-1} \cos(\theta_i - \theta'_i) \right|^2. \quad (6.12)$$

For the NEQR encoding, the inner product between two states as in Eq. 6.6 reads:

$$\begin{aligned}\langle I|I'\rangle &= \frac{1}{2^{2n}} \sum_{y,y',x,x'=0}^{2^n-1} \langle f'(y,x)|f'(y',x')\rangle \otimes \langle yx|y'x'\rangle \\ &= \frac{1}{2^{2n}} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} \delta_{f,f'} \delta_{yx,y'x'},\end{aligned}$$

where in the last equality, we use the orthonormal property of the basis states of the same state space. Then, a NEQR-kernel is equal to:

$$K_{I,I'} = \left| \frac{1}{2^{2n}} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} \delta_{f,f'} \delta_{yx,y'x'} \right|^2. \quad (6.13)$$

Finally, let us consider the inner product between two QPIE states as in Eq. 6.7 is the following:

$$\begin{aligned}\langle I|I'\rangle &= \sum_{i,j=0}^{2^{2n}-1} c_i c'_j \langle i|j\rangle \\ &= \sum_{i=0}^{2^{2n}-1} c_i c'_i,\end{aligned}$$

and, the QPIE-kernel is given by:

$$K_{I,I'} = \left| \sum_{i=0}^{2^{2n}-1} c_i c'_i \right|^2. \quad (6.14)$$

A summary of the QImR-based quantum kernels used in the simulations by the SVM model is reported in Table 6.3.

QImRs	Quantum Kernel
TNR	$ \text{Tr}(E_1 E_2 \cdots E_n) ^2$
FRQI	$\left \frac{1}{2^{2n}} \sum_{i=0}^{2^{2n}-1} \cos(\theta_i - \theta'_i) \right ^2$
NEQR	$\left \frac{1}{2^{2n}} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} \delta_{f,f'} \delta_{yx,y'x'} \right ^2$
QPIE	$\left \sum_{i=0}^{2^{2n}-1} c_i c'_i \right ^2$

Table 6.3. Summary of the four quantum image representations and their kernels for $2^n \times 2^n$ grayscale images where the gray intensity values of each pixel are in $[0, 2^q - 1]$ [174].

To investigate the compression performance of the four QImRs, we compute the Gram matrices of 100 fixed training images for two classes, 0 and 1, and show them in Figure 6.6. The elements of the Gram matrix are the quantum kernels between two images, which indicate the overlap between two quantum states that encode the information of two classical images. Specifically, these elements take values in the range $[0, 1]$: a value of 1 indicates that the two quantum states completely overlap, while a value of 0 indicates no overlap. The different off-diagonal colorings of the four Gram matrices describe the degree of overlap between the quantum states associated with distinct images. These off-diagonal elements can be related to the level of compression. In fact, if two images that are distinct in their classical representation are encoded into quantum states whose overlap is close to 1, this indicates that the encoding procedure has (highly) compressed their information in the quantum space such that the resulting states become almost indistinguishable. In other words, the two images become almost identical in their quantum representation. Conversely, if the overlap between the two quantum states is close to 0, the two images maintain their distinction even after quantum encoding.

From Figure 6.6, we can see that FRQI and QPIE perform a high compression of the classical image information with respect to NEQR in the encoding procedure. In fact, most of the off-diagonal elements of FRQI and QPIE Gram matrices are in the range $[0.8, 1]$, while for NEQR they are in the interval $[0, 0.4]$. The TNR Gram matrix shows an intermediate level of compression with respect to FRQI, QPIE, and NEQR, and its elements are in the range $[0.4, 0.7]$. This can be explained by the different ways in which the 8-bit string representing the color of the pixels is encoded. FRQI and QPIE encode the 8-bit color string in probability amplitudes using angle and amplitude encoding, respectively. In particular, the FRQI encoding maps the 8-bit string to the angle of a single qubit, while QPIE is equivalent to implementing an amplitude embedding, which allows 2^n values to be represented using only n qubits. In contrast, NEQR encodes the color using a basic encoding in which each bit is associated to a qubit. Therefore, different inputs are mapped to orthogonal vectors on a Hilbert space, and consequently no information is lost in the quantum embedding. In the case of TNR, color information is encoded in the real amplitudes of the states, however, the hierarchical encoding structure leads to an intermediate scenario between the other three representations.

Next, we want to evaluate the trade-off between accuracy and memory in binary classification problems. This is performed by calculating the mean accuracy and the

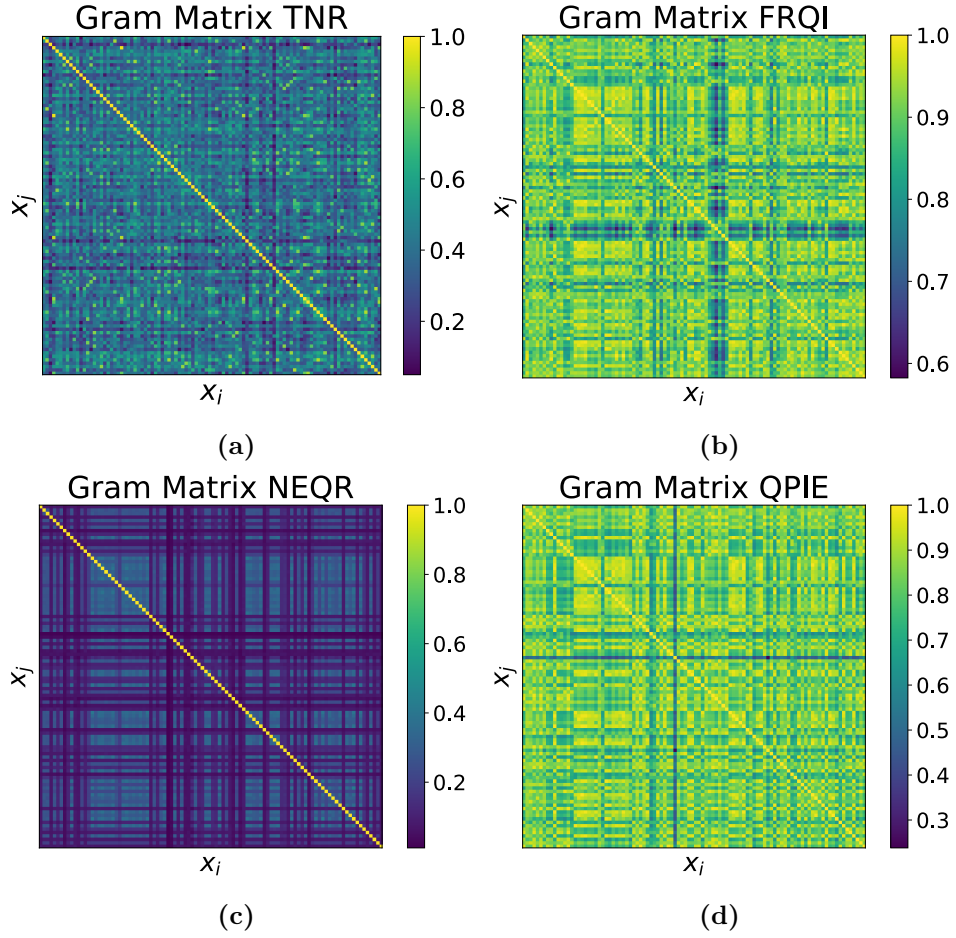


Figure 6.6. The Gram matrices of 100 training images x_i from Fashion MNIST for the quantum kernels: TNR (a), FRQI (b), NEQR (c) and QPIE (d). Samples from classes 0 and 1 have been selected. The elements on the diagonal are the kernel of a quantum state $|I\rangle$ with itself, while the elements off the diagonal are the kernel between two different quantum states $|I\rangle$ and $|I'\rangle$. The different off-diagonal coloring of the four Gram matrices describes the overlap value of two quantum states, giving information on how the different QImRs compress the classical information into quantum states [174].

standard error of the mean (SEM) between the pair of classes: $(0, 1), (0, 2), \dots, (0, 9)$. Furthermore, to improve the reliability and significance of our findings, we use the K-fold cross-validation technique with $K = 5$ (for more details, see the Appendix A). Accuracy is obtained using the `accuracy_score` function of the Scikit-Learn library, which is calculated as in Eq. 1.5. Our results show that QImR-based kernels achieve an average accuracy comparable to that of classical linear kernels on both datasets, while requiring exponentially less computational resources to store images. The results are reported in Table 6.4.

For TNR encoding, the classification accuracy of the SVM model is also evaluated in different *bond dimensions* [195] for both the MNIST and Fashion-MNIST datasets, as shown in Figure 6.7. The x -axis represents the bond dimension values, which

Kernel	Storage Cost	Accuracy (MNIST)	Accuracy (Fashion MNIST)
TNR	9 qubits	0.99 ± 0.04	0.95 ± 0.02
FRQI	9 qubits	0.99 ± 0.00	0.95 ± 0.02
NEQR	16 qubits	0.99 ± 0.00	0.94 ± 0.02
QPIE	8 qubits	0.99 ± 0.00	0.96 ± 0.02
Linear	2 048 bits	0.99 ± 0.00	0.96 ± 0.02

Table 6.4. Mean accuracy and SEM for different quantum and classical kernels in binary classification tasks. Accuracy and the SEM are calculated using the K-fold cross-validation technique with K=5 for each pair of class 0 vs. classes 1–9 [174].

describe the level of entanglement captured by the MPS, while the y -axis shows the classification accuracy (%) achieved using the MPS-encoded kernel. The plot shows that at low bond dimensions, the accuracy is limited because of insufficient encoding capacity. However, when the bond dimension increases, the accuracy improves, which indicates that the MPS representation is able to capture more relevant image features. The model achieves its highest accuracy of 99.4% for the MNIST dataset and 95.4% for the Fashion-MNIST dataset with a bond dimension of four. Thus, moderate bond dimensions are sufficient to capture the essential image correlations. Finally, the results show that the average accuracy for MNIST is consistently higher than for Fashion-MNIST, as expected due to the simpler visual structure of the MNIST digits. Moreover, for MNIST, the accuracy converges once the bond dimension reaches 4, whereas for Fashion-MNIST it does when the bond dimension is at least 6 (or higher), as shown in Figure 6.7.

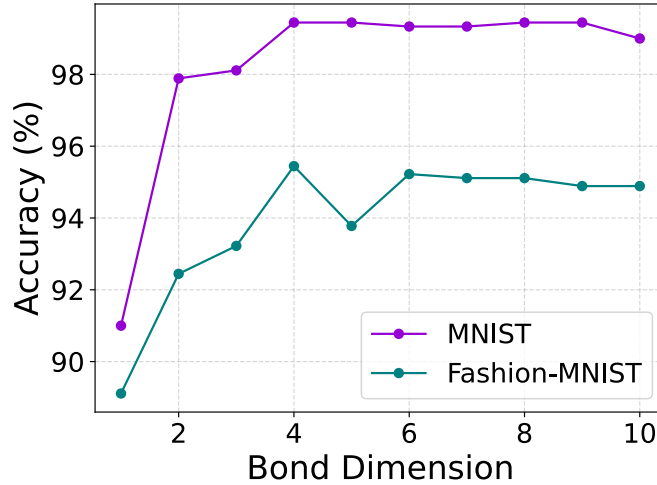


Figure 6.7. Classification accuracy of the SVM model as a function of bond dimension for MNIST and Fashion MNIST dataset. Moderate bond dimensions are sufficient to capture essential image correlations, with peak accuracy of 99.4% for MNIST and 95.4% for Fashion-MNIST at bond dimension four. Also, accuracy converges earlier for MNIST (bond dimension = 4) than for Fashion-MNIST (bond dimension > 6), reflecting the simpler visual structure of MNIST digits [174].

To conclude, in this work we compare different quantum encoding techniques for grayscale images to analyze the degree of compression when classical image information is encoded in a quantum state and to study the accuracy-memory tradeoff for binary classification problems using kernel methods.

In order to investigate compression of the classical information of an image, we compute their Gram matrices whose elements are the fidelity (overlap) between two quantum encoding states. We observe that for FRQI and QPIE, the quantum states encoding different images are very similar after the encoding procedure, whereas NEQR exhibits the opposite behavior. In other words, when classical image information is encoded in quantum systems, FRQI and QPIE achieve higher compression than NEQR. The TNR encoding represents an intermediate case between FRQI, QPIE, and NEQR.

Next, we perform binary image classification using QImR-based quantum kernels and compare their performance with the classical linear kernel. We find that the average accuracy of quantum kernels is comparable to the classical one for both the MNIST and Fashion MNIST datasets, but quantum kernels require exponentially less computational resources to store the image than their classical counterparts. However, we want to emphasize that the best-known protocols to date for loading an exact representation of data into an n qubit state require a number of quantum gates of the order $\mathcal{O}(2^n)$. This often dominates the complexity of quantum algorithms, and the efficient loading of classical data into a quantum state is still an open question.

A future research direction could be to extend the study to RGB images and compare them with other types of classical kernels. Finally, another longer-term perspective could be the possibility of studying the accuracy performance of QImR kernels when implemented on real quantum hardware and in the presence of noise effects.

Chapter 7

Quantum Diffusion Models: A Novel Paradigm in Quantum AI

This chapter is the core of this doctoral thesis and is an adapted version of two papers, i.e. the article “Quantum-Noise-Driven Generative Diffusion Models” [29], published in *Advanced Quantum Technologies*, and the arXiv preprint version “Physics-inspired Generative AI models via real hardware-based noisy quantum diffusion” [204]. This thesis is based on, to the best of our knowledge, the first generalization of classical (generative) diffusion models to the quantum domain¹. We name this novel generative paradigm *quantum diffusion models*. The chapter is structured as follows. We begin with Section 7.1, where we provide an overview of quantum generative models, discuss the motivations for extending classical diffusion models to the quantum realm, and present the subsequent works related to quantum diffusion models. In Section 7.2, we present and discuss our results. We begin by formalizing the three different scenarios within the framework of quantum diffusion models, providing numerical examples through simulations. Next, we analyze how quantum and hybrid diffusion dynamics impact the image generation task using the quantum stochastic walks formalism. Furthermore, we propose and implement a hybrid quantum-classical diffusion model based on quantum walks that uses the intrinsic quantum noise of a real NISQ device to generate MNIST images. Finally, we apply our model to medical imaging for the generation of a real-world data set.

7.1 The rise of quantum diffusion models

Quantum generative learning is the synergy between generative artificial intelligence and quantum computing. Its goal is to design and develop models that can learn complex or intractable classical probability distributions to generate new data samples using quantum resources such as quantum variational circuits, quantum operations, and quantum noise. Over the years, various approaches have been proposed within the framework of Quantum Generative Artificial Intelligence (QGenAI) to address this problem by generalizing classical GenAI models to the quantum domain. For example, Khoshaman et al. [205] introduced the Quantum Variational Autoencoder

¹The published article [29] is the final version of the arXiv preprint released on 23 August 2023, which is available at the link: <https://arxiv.org/abs/2308.12013v1>.

(QVAE) in 2018. This class of algorithms has recently been used to obtain optimal latent representations of both classical and quantum data [206]. In the same year, Lloyd and Dallaire-Demers introduced Quantum Generative Adversarial Networks (QGANs) [207, 208]. This paradigm works similarly to classical GANs, but uses quantum circuits to represent and learn data distributions. Specifically, they adopt a quantum generator and a discriminator (quantum or classical) to produce synthetic data. QGANs have been used in a variety of applications [209]. For example, they have been used to learn and load random distributions, exploiting the intrinsic stochasticity of quantum measurements as a source of noise [210]; to characterize noise in a realistic quantum device, extending the framework from quantum states to quantum operations and channels, which are referred to as SuperQGAN [211]; or as a tool for anomaly detection in high-energy physics [212].

An interesting aspect of QGenAI models is that they allow the integration of computational protocols with physical quantum devices. For example, QGANs have recently been realized using a silicon quantum photonic chip [213]. Moreover, a practical *quantum advantage* in generative modeling has been demonstrated in the data-limited scenario, comparing quantum-against-classical generative algorithms [214].

In 2020, the development of diffusion models in classical AI marked a breakthrough in generative learning. These models utilize noisy diffusion dynamics within their pipeline as essential resources for learning complex probability distributions and generating new data (see Section 2.4). At the same time, currently available quantum devices are *intrinsically* noisy, and quantum or hybrid (quantum-classical) diffusion dynamics has demonstrated better performance compared to purely classical stochastic approaches [215, 216, 217]. In this landscape, we have generalized classical diffusion models to the quantum framework in 2023, introducing a novel generative paradigm called *Quantum Diffusion Models* (QDMs) [29]. The main goal of our work is to introduce a new framework in which unique quantum features such as the non-trivial interaction between coherence and entanglement can be exploited to address the key computational challenges of classical diffusion models during inference. Furthermore, we propose considering the quantum noise present in current NISQ devices not as a problem to be detected and corrected, but rather as a key ingredient that can be leveraged to learn and generate complex probability distributions. This framework also naturally enables the generation of quantum data. In particular, in our first work [29], we have described the theoretical framework and implemented the three possible approaches of Quantum Diffusion Models (QDMs), which are summarized here:

- **Classical-Quantum:** the forward process is performed on classical data where Gaussian noise is injected through a classical diffusion dynamics. The denoising process is implemented via a variational quantum circuit.
- **Quantum-Classical:** quantum data are corrupted through quantum noise (e.g., depolarizing noise) in the forward process. Subsequently, a classical neural network is employed in the reconstruction/generation process.
- **Quantum-Quantum:** quantum data samples are corrupted with quantum noise in the diffusion process. The backward dynamics is implemented by a

variational quantum circuit.

A more detailed formalization and description of the aforementioned scenarios is provided in Section 7.2.

After our work, QDMs have been used in several scenarios ranging from medical image generation [218, 219] to the restoration and generation of pure and mixed quantum states [220, 221, 222, 223, 224]; from high-energy physics with the jet generation of quarks and gluons [225] to the development of noise mitigation techniques [226]. Furthermore, QDMs are used in Few-Shot Learning [227], many-body phase learning [228], and measurement-based approaches [229]. In addition, numerical simulations have shown that the design of hybrid quantum-classical algorithms can improve the quality of generated data [230, 231], learn complex quantum distributions [232], reduce the number of trainable parameters in the reverse denoising process [233, 234], potentially achieve sampling rates higher than those of classical DMs [235], and overcome dimensional factorization limits in discrete diffusion models [236].

These results highlight that QDMs, although still in an emerging stage, can provide a promising basis for the future development of new generative algorithms, paving the way for new scenarios in Quantum GenAI for simulated and real-world data.

7.2 Results

7.2.1 Classical-Quantum Generative Diffusion Model

The first approach of QDMs that we present consists in implementing the diffusion process classically, while the denoising procedure is performed using a quantum neural network. Next, we will use a parametric quantum circuit (PQC), but a hybrid quantum-classical neural network architecture can also be adopted. We refer to this scenario as Classical-Quantum Generative Diffusion Models (CQGDMs).

In order to realize a proof of concept for this case, we use a data set composed of 1000 data points that are distributed along a segment of the line $y = x$ in the interval $[-1, 1]$. The numerical simulation is implemented with the PennyLane and PyTorch libraries.

The diffusion process is performed by a classical Markov chain composed of a sequence of Gaussian transition kernels as follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \quad (7.1)$$

where $\beta_t = 0.3 \times \sigma(b_t)$, with $\sigma(\cdot)$ the sigmoid function and b_t is equally spaced in $[-18, 10]$ for the 40 time steps. For Gaussian noise, the reverse parameterized probability is of the form:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)), \quad (7.2)$$

where $\mu_\theta(\mathbf{x}_t, t)$ and $\Sigma_\theta(\mathbf{x}_t, t)$ are, respectively, the mean and covariance of the parametrized probability.

The denoising process starts from a normalized Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and is implemented by a PQC that is trained to estimate the mean $\mu_\theta(\mathbf{x}_t, t)$ and

the covariance $\Sigma_\theta(\mathbf{x}_t, t)$ of the denoising transition kernel of Eq. 7.2. The PQC consists of a circuit of four qubits divided into two concatenated parts called the *head* and the *tail*. The head parameters are shared among all the values of t , while the tail parameters are specific for each value of $t = 0, \dots, 39$. The head takes as input the values of the coordinates of a data sample and encodes them in the state of the first two qubits with an angle encoding, while the other two qubits are initialized in the quantum state $|0\rangle$. After the encoding stage, the circuit consists of 256 layers containing Pauli rotations along the three axes applied to all four qubits, interleaved with layers of entangling controlled-NOT gates. In particular, we adopt the *StronglyEntanglingLayers* circuit of PennyLane with default arguments. At the end of the circuit measurements are performed and the expectation values of the Pauli operator Z for each of the four qubits are computed. The tail is similarly structured, but the first operation is the angle encoding of the four expectation values previously obtained from the head. In order to simplify the model, we assume the covariance matrix of the denoising kernel is diagonal and only two values for the variance are necessary. This assumption is justified by the fact that between each couple of time steps $(t, t - 1)$ the data samples are classical and also the evolution of their distribution.

Next, the four expectation values measured from the tail are used for the predictions of the mean (the first two values) and variance (the second two values) of the classical kernel². The model is trained for 40 000 epochs using random batches of 100 points to minimize the KL divergence loss between the predicted and initial data distribution, using Adam optimizer with a learning rate of 10^{-4} .

The simulation results are presented in Figure 7.1a, which shows that the model successfully reconstructs the initial data distribution $q(\mathbf{x}_0)$ with good precision. This is quantified by the KL divergence between the original data distribution and the reconstructed one at $t = 0$, starting from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ at $t = 40$. We estimate the KL divergence values using 1 000 samples following the method described in [237]. Furthermore, in Figure 7.1b, we show the evolution of the loss function during the training process.

In conclusion, although the model presented here operates in a simplified setting, in the literature there exist demonstrations of a quantum advantage of using PQCs instead of a deep neural network. For example, it has been shown that PQCs can outperform classical neural networks in generative tasks [238] and lead to an exponential advantage in model size over neural networks in approximating high-dimensional smooth functions [239]. These results indicate that adopting PQCs could also be advantageous in the QDMs framework, providing a basis for more in-depth future studies.

7.2.2 Quantum-Classical Generative Diffusion Model

Quantum systems are never perfectly isolated, but they interact with an external environment. This means that they are easily prone to noise. Consequently, it is natural to consider the idea of implementing the diffusion process by exploiting noisy quantum dynamics in the forward process of generative diffusion models. In this

²In particular, we multiply the expectation values used for the mean by a factor 3 to enlarge the possible range, and the variance values are increased by 1 to force positivity.

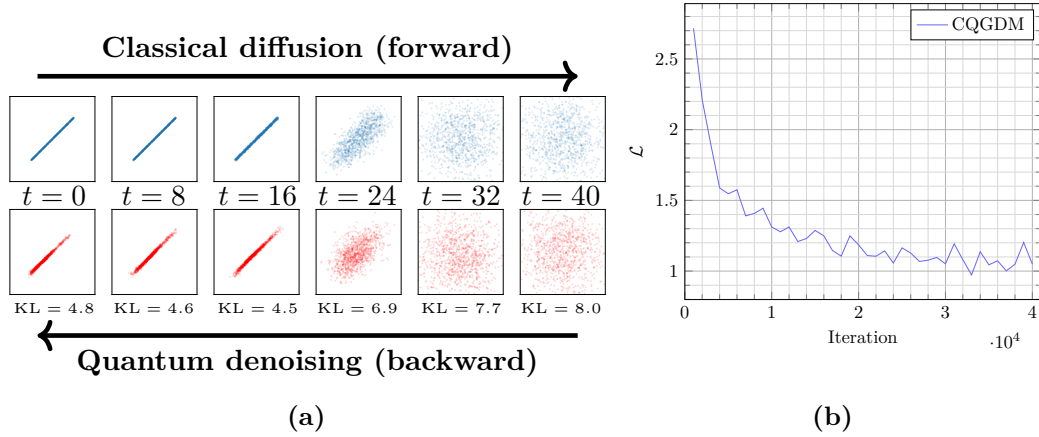


Figure 7.1. Evolution of the data distribution for a trained CQGDM (a), and its Kullback–Leibler divergence loss function $\mathcal{L}$ during training, averaged over every 10^4 iterations (b). The initial data distribution consists of two-dimensional points distributed in a line segment between -1 and 1 . The forward process is implemented via a classical diffusion dynamics that transforms the initial data distribution at time $t = 0$ into the final prior distribution, which is a two-dimensional normalized Gaussian, at time $T = 40$. The denoising is implemented via a (noiseless) simulated PQC, which reconstructs the initial data distribution from the Gaussian prior distribution. In (a), the top row shows the forward process (from left to right) for a set of 1000 points at different discrete time steps $t = 0, 8, \dots, 40$. In the bottom row, we display the denoising (from right to left) of a different set of 1000 points. Moreover the KL divergence between the data distribution $p(\mathbf{x}_0)$ and the reconstructed distribution at the corresponding time is reported [29].

context, we consider a quantum dataset, which is a collection of data samples that can be obtained in two ways: 1) from classical data information encoded in quantum states, 2) data acquired directly by quantum sensing technologies [240, 157]. In the following, quantum data are described by the density operator ρ .

As in the classical framework, quantum diffusion dynamics can occur in discrete time or in continuous time. In the first case, the process is realized and described by *quantum Markov chains*, which generalize their classical counterparts [241]. In the second case, by *quantum stochastic equations*, such as the Stochastic Schrödinger Equation (SSE) [242, 243], that model the dynamics of an open quantum system subject to an external noise source.

Formally, following the Gudder approach [241], a quantum Markov chain can be described by two elements: i) a directed graph G whose sites represent the possible states that the quantum system can occupy, and ii) a set of transition operation matrices (TOMs) $\mathcal{E}$ whose elements are completely positive maps and whose column sums form a CPTP map (see Section 4.3.3). In the one-dimensional graph, the TOMs reduce to the quantum channels introduced in Section 4.4. From a physical perspective, the elements $\mathcal{E}_{ij}$ describe the transition operation of the quantum system from the site j to the site i in one time step. Given a density operator ρ , representing the state of the system, the quantity $\mathcal{E}(\rho)$ is again a density operator. Moreover, it can be shown [241] that if $\mathcal{E}$ and $\mathcal{F}$ are two TOMs with the same dimensions acting on the same Hilbert space, then $\mathcal{E}\mathcal{F}$ is a TOM under matrix multiplication. The dynamics of a quantum system after n discrete time steps can be described as

follows

$$\rho_n = \mathcal{E}^n(\rho) = \mathcal{E}(\mathcal{E}^{n-1}(\rho)), \quad (7.3)$$

with $n = 2, 3, \dots$, and where ρ and ρ_n are, respectively, the initial and final state of the system after n time steps.

The continuous-time approach adopts SSEs to describe the dynamics of quantum diffusion. In this framework, given a system in the state $\rho(t)$, its stochastic evolution is determined by a SSE that takes the form:

$$\dot{\rho}(t) = -i[H(t), \rho(t)], \quad (7.4)$$

with $\hbar = 1$, and where the Hamiltonian $H(t) = H_s(t) + H_p(t)$ consists of the sum of the system Hamiltonian $H_s(t)$ and the stochastic term $H_p(t)$, which represents the stochastic dynamics to which the quantum system is subject. A particular example of a stochastic equation is the Kossakowski-Lindblad-Gorini master equation described in Section 4.5.

In the following of this section, we present two implementations of discrete-time quantum noisy dynamics in the forward process of QDMs, while we will employ the continuous-time approach in Subsection 7.2.4.

We begin with the practical implementation of the Quantum-Classical Generative Diffusion Model (QCGDM) model, i.e., where the forward process is realized through quantum dynamics, and the denoising process is implemented by a classical neural network. Specifically, the forward dynamics is modeled as a quantum Markov chain in which an initial quantum state ρ_0 is progressively degraded by successive injections of depolarizing noise until it reaches the maximally mixed state $\rho_T \equiv \frac{\mathbf{I}}{d}$, where d denotes the dimension of the Hilbert space where the quantum system lives.

The forward process is simulated using the PennyLane library, where a single-qubit circuit is initialized in a pure quantum state and subsequently degraded by depolarizing quantum noise for $t \in [1, T \equiv 5]$. Recall that the action of the depolarizing channel on a quantum state is given by:

$$\rho_{t+1} = (1 - p_t)\rho_t + p_t \frac{\mathbf{I}}{d}, \quad (7.5)$$

where $p_t \in [0, 1]$ is the probability of depolarization at the time step t . The values of p_t are linearly scheduled and uniformly distributed in the interval $[1, T]$. All states $\rho_0, \dots, \rho_T$ are collected and used for training the backward denoising process³.

The gradual loss of information in the quantum state can be quantified by the von Neumann entropy $S(\rho)$, which is zero for *pure* states, $\rho = |\psi\rangle\langle\psi|$, strictly positive for *mixed* states, and maximal, equal to $S(\frac{\mathbf{I}}{d}) = \log_2 d$, for the maximally mixed state (as described in Section 4.2.5).

The denoising process is implemented with classical neural networks trained to simulate the inverse dynamics of forward, in order to obtain an approximation $\hat{\rho}_0$ of the initial pure state ρ_0 , starting from the maximally mixed state $\rho_T \equiv \frac{\mathbf{I}}{2}$. As in

³In our simulations, the forward process is classically simulated, so all states ρ_t are obtained directly from the simulator. However, on real NISQ devices, quantum state tomography is required to reconstruct ρ_t and subsequently compute the loss between ρ_t and the predicted state $\hat{\rho}_t$ generated by the neural network. As a possible alternative to tomography, one could employ the *swap test* algorithm [28], a common method for estimating quantum fidelity between quantum states.

classical diffusion models, we use classical neural networks to predict the state $\hat{\rho}_t$ given the state $\hat{\rho}_{t+1}$. The entire process is implemented using the PyTorch library and employs T different neural networks that are specialized to reconstruct the mixed state $\hat{\rho}_t$. The inputs and outputs of the neural networks are mixed quantum states of a single qubit, which can be described by only three real numbers⁴. These three values can be expressed by spherical coordinates, namely radius $r \in [0, 1]$, polar angle $\theta \in [0, \pi]$, and azimuthal angle $\phi \in [0, 2\pi)$. Therefore, both the input and output layers of the neural networks have dimension 3, and the final activation functions for the three neurons are, respectively, $\sigma(r)$, $\pi\sigma(\theta)$, and $2\pi\sigma(\phi)$, where $\sigma(\cdot)$ denotes the sigmoid function. Concerning the internal structure of the neural networks, they consist of a single hidden layer with 5 neurons and ReLU activation, for a total of 32 parameters, and are trained to minimize the loss function between the forward state ρ_t and the predicted state $\hat{\rho}_t$ when the input is the state ρ_{t+1} . More precisely, the loss is calculated and minimized for batches of $n \equiv 16$ reconstructed states $\hat{\rho}_t$ with random values of t . Formally, the loss function is defined as follows:

$$\mathcal{L}(\hat{\rho}_t, \rho_t) = \frac{1}{n} \sum_{t=1}^n (1 - F(\hat{\rho}_t, \rho_t)), \quad (7.6)$$

where $F(\cdot, \cdot)$ is the quantum fidelity introduced in Section 4.2.5. The optimization is performed with Adam optimizer with a learning rate of 10^{-2} .

In Figure 7.2a, we show the simulation on a single-qubit system with ten QCGDMs and $T = 5$. Our simulations demonstrate that the neural network is able to reconstruct the information of the initial state ρ_0 . In particular, the average fidelity over a sample of 100 different random states is 0.997 ± 0.013 . Furthermore, in Figure 7.2c, the red curves illustrate the evolution of the loss for data batches during training for one of the single states, as well as the loss between $\hat{\rho}_0$ and ρ_0 (shown in dashed lines).

7.2.3 Quantum-Quantum Generative Diffusion Model

In this final subsection, we present the third and last approach within QDMs, namely the case in which both the diffusion and denoising processes are quantum in nature. We refer to this framework as QQGDMs. Specifically, the forward process can be implemented and formally described through a quantum Markov chain or SSEs, while the backward process can be realized using PQC or hybrid quantum-classical neural networks.

Analogously to the QCGDM model, we propose a practical implementation in which the forward quantum noise dynamics employs a depolarizing channel to degrade the initial state ρ_0 until it becomes the maximally mixed state $\rho_T = \mathbf{I}/d$. However, in this case, the backward process cannot be implemented by unitary dynamics but requires that the backward quantum dynamics also be non-unitary. Indeed, a purely unitary dynamics cannot reconstruct an initial state different from the maximally mixed state:

$$\frac{\mathbf{I}}{d} \mapsto U \frac{\mathbf{I}}{d} U^\dagger = \frac{\mathbf{I}}{d}. \quad (7.7)$$

⁴The quantum state of a single qubit is described by a 2×2 Hermitian, positive semi-definite (complex) matrix with unit trace, consequently, it can be fully characterized by three independent real parameters.

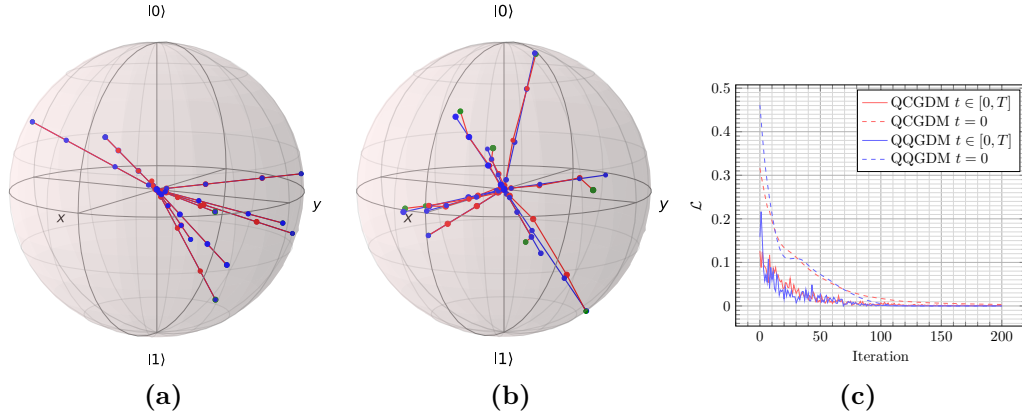


Figure 7.2. Reconstruction of 10 random one-qubit pure states for trained (a) QCGDMs and (b) Quantum-Quantum Generative Diffusion Models (QQGDMs), and (c) the evolution of the quantum infidelity loss function for both models for one of the 10 states. In (a) and (b), the blue segments represent the evolution of the forward noisy dynamics implemented by depolarizing channel. In (a) the red segments are the evolutions of the reverse process implemented with a classical neural network and in (b) with a parameterized quantum circuit. The green points are the final reconstructed states. In (c) the solid lines describe the evolution of the loss used during the training, while the dashed lines the loss calculated only on the final reconstructed states. The red lines refers to the QCGDM and the blue lines to QQGDM [29].

Therefore, in order to obtain a non-unitary dynamics, we implement the backward process following the approach introduced in Section 4.3.1, namely by coupling the principal system to an external environment that is traced out at each time step.

More precisely, in our numerical simulations, we consider a single qubit coupled via a parametrized circuit to another single qubit that acts as the environment. During training, a PQC receives the state $\hat{\rho}_{t+1}$ as input and, after tracing out the environment, produces the state $\hat{\rho}_t$. Our simulations are classically implemented, and we can collect the quantum states ρ_t and $\hat{\rho}_t$ at each time step, allowing the calculation of the loss as in Eq. 7.6. For implementations on real NISQ devices, the strategies described above, such as the swap test, can be used to compute the fidelity between two quantum states. The PQC adopted is the *StronglyEntanglingLayers* circuit with 5 layers and default parameters. The optimization is performed using Adam with a learning rate of 10^{-2} to minimize the loss, which is calculated for batches composed of 16 pairs $(\hat{\rho}_t, \rho_t)$ with random values of t , where ρ_t and $\hat{\rho}_t$ are the forward and predicted backward states, respectively.

The evolution of the loss during the training procedure is depicted in blue in Figure 7.2c, and the resulting simulations for 10 different initial pure quantum states are presented in Figure 7.2b. Our results indicate that the model successfully learns the reverse dynamics from the maximally mixed state ρ_T (with $T = 5$) to the state $\hat{\rho}_0$. Finally, we calculate the average quantum fidelity for 100 random pure quantum states, obtaining 0.996 ± 0.009 .

Before concluding, we emphasize that the quantum noise is more general than its classical counterpart. In particular, the noise distributions employed in QCGDMs can be expressed in more general forms than the typical Gaussian distributions

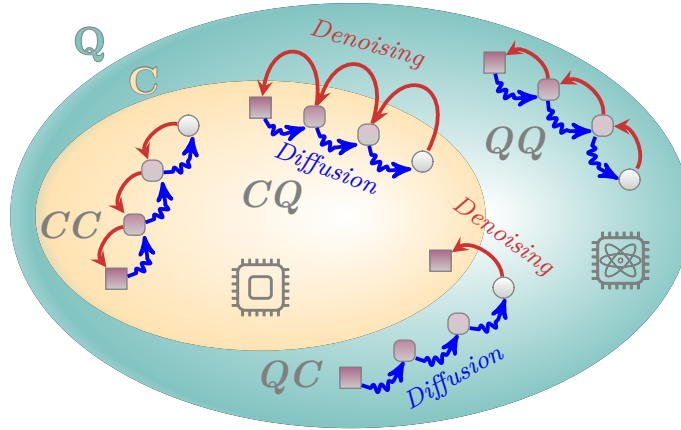


Figure 7.3. Illustration of the space of the probability distributions that are tractable with classical computation (C) and instead only with quantum computation (Q). We depict the trajectories arising from the mappings between probability distributions (colored and white shapes) during the diffusion (blue wavy arrows) and denoising (red arrows) processes for the four different combination: CC, CQ, QC and QQ indicating whether the diffusion (first letter) and the denoising (second letter) are classical or quantum. The initial data distribution (squares) is progressively transformed during diffusion (changing color and shape) to an uninformative distribution represented by the white circles, and vice versa during denoising [29].

commonly used in classical DMs. For instance, in this framework, it is possible to obtain non-classical distributions associated with entangled states, which have no analogue in the classical domain. In other words, there exist probability density distributions that are purely quantum. An illustration of the three approaches and the space of the probability distributions is shown in Figure 7.3. This property could be used to implement diffusion processes that are not classically realizable or to enhance the performance of generation tasks. We analyze and discuss this topic in depth in Section 7.2.4, where we study and compare image generation performance across different types of diffusion dynamics: purely quantum, hybrid quantum-classical, and purely classical.

7.2.4 Quantum stochastic dynamics for image generation

The concepts of probability and stochasticity play a crucial role in generative models (see Section 2.3.2). Here, we investigate the performance of diffusion model in image generation task when the classical stochastic dynamics in the forward chain are replaced or interact with quantum stochastic diffusion processes. For this purpose, we decide to adopt the formalism of Quantum Stochastic Walks (QSWs), which provides a useful tool to study the transition from classical to quantum diffusion dynamics. This is motivated and supported by previous works in which QSWs has been used to identify an optimal combination of classical and quantum dynamics to improve performance in the information transport task [215, 216, 217]. In addition, we adopt the categorical diffusion model framework introduced in Section 2.4.3.

Before presenting our results, we briefly introduce the formalism of QSWs introduced by Whitfield et al. [244]. Formally, a continuous-time QSW dynamics

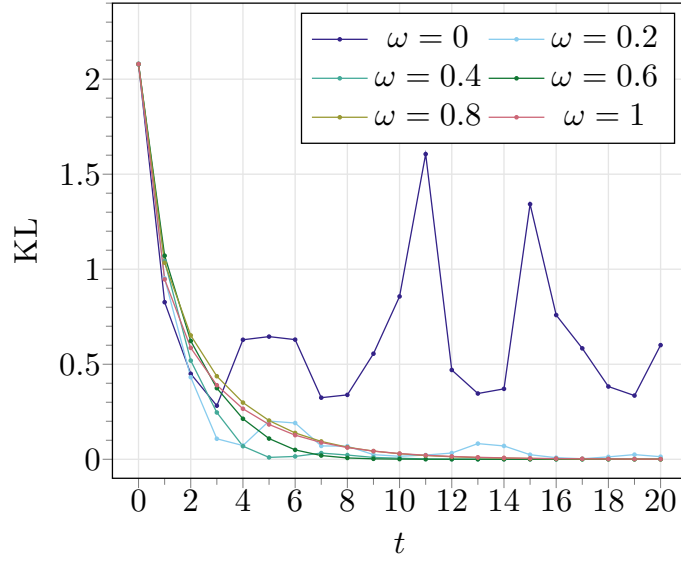


Figure 7.4. The KL divergence between the populations of a single QSW and the uniform distribution on a cycle graph of 8 nodes for different value of ω at $t = 0, \dots, 20$ time steps. The walker is initially in the node 0, corresponding to the state $\rho = |0\rangle\langle 0|$ and its evolution over the graph is obtained by solving the Eq. 7.8 [204].

on a graph is described by the Kossakowski–Lindblad–Gorini master equation as follows:

$$\frac{d\rho}{dt} = -(1 - \omega)i[H, \rho] + \omega \sum_j \left(L_j \rho L_j^\dagger - \frac{1}{2} \{L_j L_j^\dagger, \rho\} \right), \quad (7.8)$$

with $\hbar = 1$. Here, ρ is the walker density matrix, H is the Hamiltonian of the system that describes the coherent evolution, and L_j are Lindblad operators responsible for the incoherent dynamics, which represent the interactions of the system with an external environment. The continuous parameter $\omega \in [0, 1]$ quantifies the interaction between coherent and incoherent evolution. For example, if $\omega = 0$, Eq. 7.8 describes the evolution of a pure Quantum Walk (QW). In contrast, setting $\omega = 1$ and choosing $L_{ij} = S_{ij} |i\rangle\langle j|$, Eq. 7.8 describes the dynamics of a Classical Random Walk (CRW) where $|i\rangle$ is the quantum basis state associated with node i of a graph and S_{ij} is the transition matrix of the walker from node j to i . For a formal description of CRW and QW, see Appendix B.

In the following, we associate a categorical data sample with a quantum stochastic walker that evolves on a cycle graph of 8 nodes representing the categories of the sample. In this framework, the density operator ρ of Eq. 7.8 represents the state evolution of the walker on the graph. In particular, the diagonal elements ρ_{ii} give the probabilities that the walker is at the node i , while the off-diagonal elements ρ_{ij} contain information on the peculiar quantum mechanical effects (coherence) during diffusion dynamics. Adopting the terminology of quantum information theory, we refer to the diagonal elements of the density matrix ρ as *populations*, while the off-diagonal elements are called *coherences*. Figure 7.4 illustrates the evolution of the KL divergence between the populations of a quantum stochastic walker on a cycle graph with 8 nodes and the corresponding uniform distribution, for different values

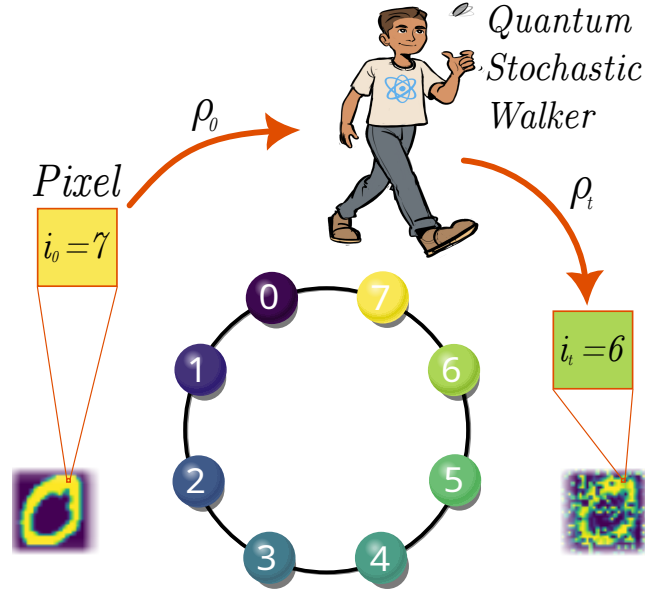


Figure 7.5. Illustration of our model. Each pixel of an image represents an independent quantum stochastic walker moving on a cycle graph of 8 nodes that correspond to the gray intensity values. The walker moves on the graph by Eq. 7.8 [204].

of ω . In the case of pure quantum diffusion dynamics ($\omega = 0$), the KL divergence exhibits an oscillatory behavior. In contrast, for the classical dynamics ($\omega = 1$), it converges smoothly to zero. In hybrid scenarios ($\omega \in (0, 1)$) the incoherent term in Eq. 7.8 dampens the oscillations observed in the quantum case, resulting in a faster convergence compared to the classical scenario.

Image generation

To investigate how different QSWs dynamics shown in Figure 7.4 impact the generation of new samples, we perform image generation using the MNIST dataset. However, to successively compare the results with the implementation on real quantum hardware described in Section 7.2.5, we rescale the grayscale values of the pixels from $[0, 255]$ to $[0, 7]$.

Here, we model a forward dynamics in which each pixel of the image is treated as an independent QSW evolving on a cycle graph with 8 nodes, which correspond to the possible gray intensity values of the pixel. An illustration of our model is shown in Figure 7.5.

In the next, for simplicity, we describe the forward dynamics for a single QSW. The walker is initially at node i_0 , where $i_0 \in [0, 7]$. The walker is described by the quantum state $\rho_0 = |i_0\rangle\langle i_0|$ that evolves according to Eq. 7.8 where we consider discrete time steps with $\delta t = 6 \cdot 10^{-1}$. At each time step, we collect the populations of states $\rho_1, \dots, \rho_T$, and use them to define categorical distributions from which to

sample the walker positions as follows:

$$i_t \sim \text{Cat}(\mathbf{p} = \text{diag}(\rho_t)), \quad (7.9)$$

$$\rho_t = |i_t\rangle\langle i_t|, \quad (7.10)$$

where i_t is the position of the walker on the graph at time t and ρ_t is the corresponding quantum state. The backward process is implemented with a MLP that takes as input the one-hot encoding of the position of the walker at time t for all pixels in the image, and is trained to predict their positions at time $t - 1$.

To quantify the performance of different quantum stochastic dynamics in image generation, we compute the Fréchet inception distance (FID) metric [32], which evaluates the quality of images produced by a generative model. The FID measures the distance between the original and generated datasets and is defined as:

$$\text{FID} = \|\mu - \mu'\|_2^2 + \text{tr}(\Sigma + \Sigma' - 2(\Sigma\Sigma')^{\frac{1}{2}}), \quad (7.11)$$

where μ and μ' are the mean of the multivariate normal distribution of the features extracted by a pretrained convolutional model⁵ from respectively the original and generated image datasets, with Σ and Σ' the corresponding variances. A higher value of FID indicates lower performance of the generative model. Furthermore, to statistically evaluate the performance of different models, we implement 10 independent trainings of the same model for each value of $\omega \in \{0, 0.1, 0.2, \dots, 1\}$. In Figure 7.6 we show the box plots⁶ of FID values of the generated digits 0. The mean and standard error of the mean are also reported in Figure 7.6.

From Figure 7.6 we see that for a hybrid quantum-classical stochastic dynamics ($\omega = 0.3$), the mean value of FID is lower than in the classical case ($\omega = 1$). Moreover, in the hybrid scenario the FID values of the 10 repetitions are closely distributed around the median, while in the classical case most of the FID values are above it. Except for the single upper outlier, all repetitions for $\omega = 0.3$ result in better FIDs than half of the runs for $\omega = 1$. This means that the model with that specific interplay of quantum-classical dynamics is able to steadily generate better samples and is also statistically more robust than the classical one. Furthermore, in Figure 7.4 we have seen that the hybrid dynamics $\omega = 0.3$ converges to a uniform distribution with a smaller number of time steps.

The better performance of the hybrid dynamics for $\omega = 0.3$ can be interpreted by noting that a balance between quantum coherence and stochastic noise enables faster and more efficient information transport across the graph. In addition, the value is consistent with the values of ω reported in [215, 216, 217]. In particular, it is shown in [215] that for a large class of graphs (including random graphs, small-worlds, rings, stars, etc.) with diameter⁷ $D > 3$, the transport efficiency of a QSW reaches its highest value typically around $\omega \sim 0.1$, whereas purely

⁵Usually, convolutional neural networks with an *Inception* architecture are used for feature extraction [245].

⁶In descriptive statistics, the box plots require at least five samples to provide reliable statistical information [246].

⁷The diameter D of a graph is the length $\max_{(u,v)} d(u,v)$ of the longest shortest path between any two vertices u and v , where $d(u,v)$ is a graph distance [247].

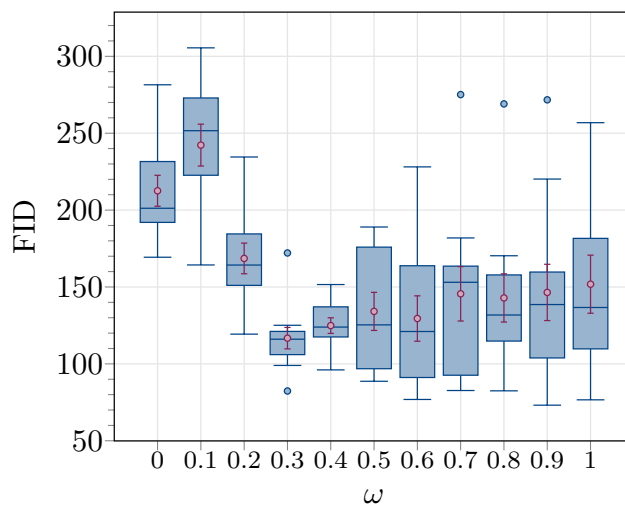


Figure 7.6. Box plot of FID values for different values of ω . Each box is obtained from 10 different repetitions of the model training for the same value of ω with $T = 20$. Mean and standard error of the mean are also reported in red. The plot shows how the hybrid quantum-classical dynamics with $\omega = 0.3$ generates statistically better image datasets [204].

classical dynamics ($\omega = 1$) is optimal for fully-connected graphs ($D = 1$). Since the eight-node cycle graph considered here satisfies $D = 4 > 3$, the better performance of the hybrid dynamics with $\omega = 0.3$ observed in Figure 7.6 can be explained within this framework. More efficient transport results in better diffusion over the graph, allowing for a more complete exploration of the state space. This leads to more stable and better-structured representations, a crucial aspect also in diffusion models, where the quality of the representation depends on the properties of the diffusion process. In this sense, the improvement observed in the hybrid dynamics for $\omega = 0.3$ reflects an increase in the efficiency of the diffusion process, with beneficial effects on the quality of the learned representations.

In Figure 7.7, we show nine generated images of the digit “0” obtained using quantum, hybrid, and classical stochastic dynamics. The equivalent figure for the other digits of MNIST is available in Appendix C.

In summary, we demonstrate that the impact of quantum noise within the forward diffusion models dynamics influences the effectiveness generation process. More precisely, we find that the model with a particular hybrid diffusion dynamics described via QSWs produces sample sets with a lower FID and is also statistically more robust than its classical counterpart.

Implementation details

The implementation is carried out in Python 3 using the open-source software QuTiP [248] for the simulation of QSWs dynamics, and the PyTorch library for the implementation of denoising and generative processes. In particular, we use the *mesolve* function of QuTiP to compute the evolution of the quantum stochastic walker state for different values of ω .

The backward process is implemented via a MLPs composed of PyTorch linear

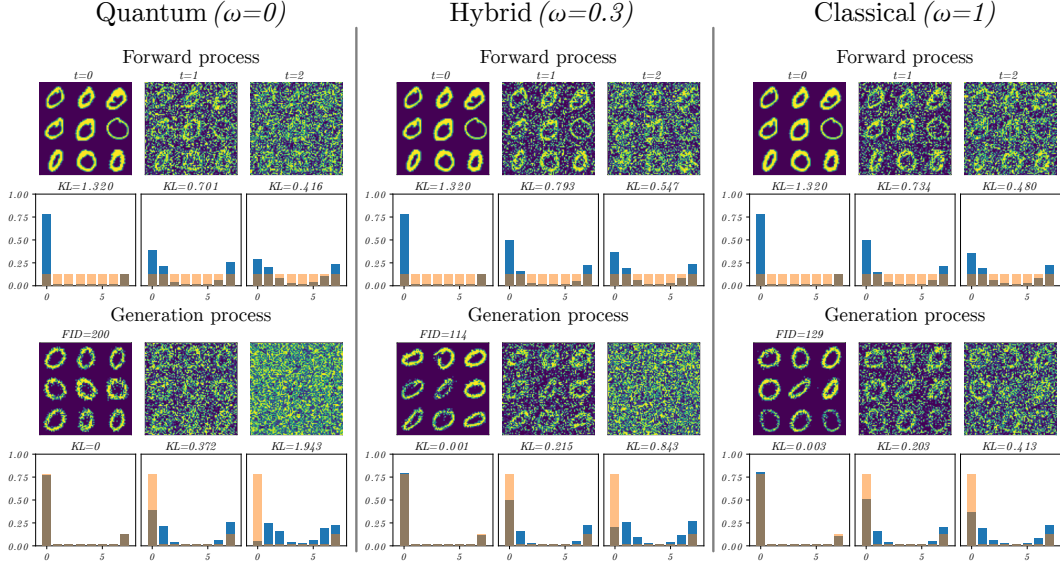


Figure 7.7. Image generation with a quantum ($\omega = 0$), hybrid ($\omega = 0.3$) and a classical ($\omega = 1$) forward diffusion dynamics for $T = 20$ time steps. We report the first two steps of the forward chain (in the first row) of nine different samples, and the evolution of the distribution of the pixel intensity values for the entire dataset of digits “0” of MNIST (blue bins in the second row) that is compared with the final uniform prior (orange bins in the second row). The KL divergence value between the two distributions is reported on the top. We also show the final two steps of nine generated samples, and the FID score between the entire training dataset and generated dataset is reported (third row). In the last row, we compare the distribution of the pixel values of the entire generated dataset (blue) with the training dataset distribution (orange), and KL divergence between them [204].

layers with ReLU activation functions. More precisely, the architecture is structured with a head layer of size 800 shared between all time steps t and two equally-sized tail layers specialized for each step. The input layer performs a one-hot encoding of the walker positions at time t for all image pixels, while the final layer predicts the logits at the previous time step $t - 1$. In the optimization procedure, the categorical cross-entropy loss is minimized over 10^4 epochs using the Adam optimizer, with a batch size of 16 samples and a learning rate set to 10^{-3} .

7.2.5 Real hardware-based noisy quantum diffusion

A hybrid QSW can be seen as a quantum walker that interacts with an external noisy environment. From this idea, in this section, we perform image generation by implementing a QW that harnesses the intrinsic noise of a NISQ device in the forward process.

For this purpose, we adopt the efficient circuit implementation of a QW on a cycle graph proposed by Razzoli et al. [249]. The efficiency of the algorithm allows us to modulate the amount of noise introduced in the forward chain by delay operations. Here, each pixel is a QW moving on a cycle graph of 8 nodes. Furthermore, the rotation-invariant property of the cycle graph allows us to run each walker as if starting from the same initial state, and then re-mapping the outcome to the specific

value of the pixel color by a shift operation. In this way, the forward chain is executed only once for all QWs, a condition that is essential for implementing our algorithm given the limited availability of current quantum processors.

Mapping pixel intensity values onto the positions of the graph allows us to introduce quantum effects into the pixel dynamics and, at the same time, makes our model scalable and independent with respect to image size compared to other QDMs approaches for image generation [230, 250]. In particular, our model requires only 4 qubits to generate 28×28 grayscale MNIST images (normalized to 8 gray values): 3 qubits encode the position of the QW on the 8 graph nodes, while 1 qubit represents the coin degree of freedom of the quantum walker. This modeling is particularly well-suited for QPUs of IBM quantum devices, due to its maximum connectivity degree of 3 and the direct interaction between the coin qubit and all position qubits. It also supports the parallel execution of multiple walkers in distinct regions of the same QPU.

In Figure 7.8 we depict the quantum circuit used for the implementation of QW. At the end of the circuit, we perform measurements on the position qubits to obtain the walker distribution for each time step $t \in [1, T]$. This is used to define the categorical function from which we sample the new position of walker on the graph. The noise is injected with *delay* operations expressed in seconds as follows:

$$\text{delay} = c \times \text{scaling} \times dt, \quad (7.12)$$

$$\text{scaling} = \left\lfloor \frac{\sin^2(\frac{\pi}{2} \frac{t}{T-1})}{8} \right\rfloor 8, \quad (7.13)$$

where the value of *scaling* is truncated to the nearest integer multiple of 8 for hardware reasons, and $dt = 5 \cdot 10^{-10}$ is the time length in seconds of a single operation on the used device.

We choose $c = 5 \cdot 10^4$ to ensure the convergence of the forward process to a uniform distribution after $T=20$ time steps. The scaling of the injected noise is chosen in analogy to the cosine schedule of noise in the classical diffusion models

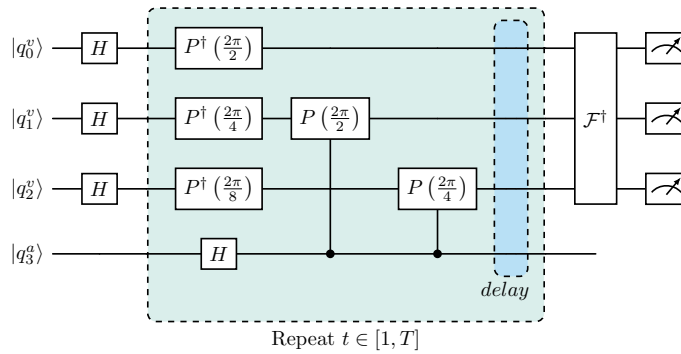


Figure 7.8. Implementation of discrete time QW on cycle graph using quantum circuits. The H and P are respectively Hadamard and phase gates, and $\mathcal{F}$ is the quantum Fourier transform [78]. The outer shaded block indicates a single time step of the walker and is repeated for t times up to T . The inner shaded block modulates the amount of noise in the circuit by a delay operation scaled as in Eq. 7.12 [204].

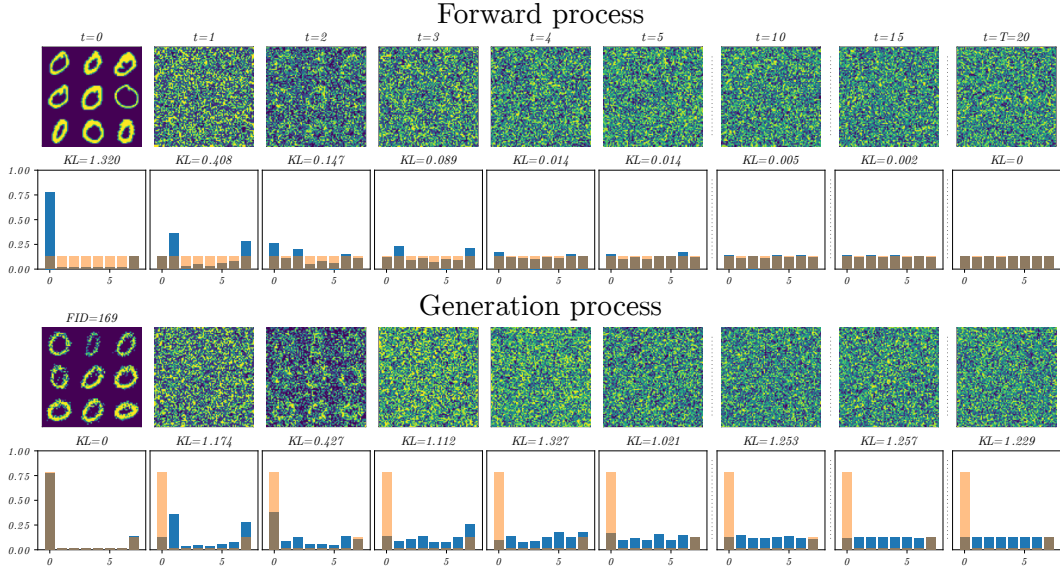


Figure 7.9. Images generation with QW-based DM with noise from real *ibm_brisbane* NISQ device. Evolution of nine samples of the real dataset in forward (first row) and nine generated samples in backward (third row). In the second row we show the evolution of the pixel values distributions (blue) for the entire training dataset reporting the KL divergence with the uniform distribution (orange). In the fourth row we show the value distributions of a generated dataset of the same dimension of the training set comparing it with the original dataset using the KL divergence. Also, we report for $t = 0$ the value of FID equal to 169 between the original and generated dataset [204].

of [251]. The backward process is implemented with a MLP analogous to the one used for the QSW case described previously.

We train our model with 6 903 full-size 28×28 MNIST images of the digit “0” with the forward implemented in Qiskit library initially simulated on *fake_brisbane* with 10^5 shots and then run on the real device *ibm_brisbane* with 10^4 shots. The number of shots is chosen as the maximum available for the simulator, and reduced considering the computational resources available for the running of the algorithm on the NISQ hardware.

In Figure 7.9 we illustrate the forward and generation processes with diffusion implemented on the real IBM quantum machine. The equivalent figure obtained using the simulator is available in Appendix C together with the other digits of MNIST. In the first and third rows are, respectively, illustrated the quantum forward evolution via QWs (from left to right) and the image generation by a trained classical MLP (from right to left) at different time steps t for nine images of the original and generated dataset. In the second row, the transformation of the initial distribution of the pixel values (in blue) for the full training data set is overlapped with the desired final uniform distribution (in orange). The final row shows the distribution of the pixel values generated images (in blue) overlapped with the original distribution of the training data set (in orange). The KL divergence between the two distributions is also reported. Finally, we report the value of FID= 169 calculated between the original and generated data sets.

In conclusion, we propose a model that allows for the generation of images of

any size by exploiting the noise of a real NISQ device and taking into account the topology and connectivity of the QPU. More precisely, we generate 28×28 grayscale MNIST images using only 4 qubits, efficiently implementing a quantum walk dynamics on a cycle graph whose nodes are the color values of a single image pixel. The shift-invariant property allows walkers for individual pixels to be run independently, starting from the same initial state and then remapping the result to the specific pixel color value. Furthermore, our forward protocol, thanks to the small number of required qubits, allows for the implementation of multiple walkers in parallel and even to run multiple shots of only a single walker to estimate the expectation value distribution to be used to sample the noise of multiple qubits. This enables the implementation on the currently available NISQ devices, as well as a reduction in the number of executions performed on the machine.

Some future research directions could focus on further integrating the QPU topology with the QW topology implemented at the circuit level. In particular, we can further leverage the benefits of connectivity to increase the range of pixel intensity values. Furthermore, error correction or mitigation techniques could be used to better control the noise level of the quantum forward process. Another interesting perspective is the possibility of experimentally realizing the quantum walks, making our model directly applicable to quantum data, and then using quantum neural networks in the reverse process. This could be a valuable tool for applications such as quantum sensing, metrology, and chemistry.

7.2.6 Applications to medical and real-world datasets

In this section, we extend the applicability of our model to color images and real-world datasets [252]. This cross-disciplinary work stems from an international collaboration (during my 3-month research visit) with BCNMedTech of the Universitat Pompeu Fabra (UPF) and the Barcelona Supercomputing Center (BSC) and may provide an application of interest to the field of medical imaging. In particular, we adopt two medical datasets: BloodMNIST [33] and BraTS2020 [34]. As discussed in Section 7.2.5, our model is capable of generating grayscale images with values in the range $[0, 2^q - 1]$ of arbitrary size while requiring at most $q + 1$ qubits (q for pixel intensity and 1 for the coin). Furthermore, it is possible to easily generalize the model to color images using independent forward walkers for each channel and each pixel. This characteristic makes it suitable for implementation both on current quantum hardware (consisting of tens to a few hundred qubits) and on real-world datasets. In our implementations, we exploit the intrinsic noise of the *ibm_torino* real quantum device, while also applying circuit optimization to reduce the depth and number of gates. The latter procedure significantly improves the results obtained from the execution of circuits and is available in Qiskit via the *transpiling optimization level* function, which we set to 3. This optimization allows us to reduce the level of noise in the initial time steps and therefore increase the grayscale-intensity range of each pixel with respect to the MNIST simulations. Furthermore, due to the more complex nature of the two datasets, in the backward process we used a more powerful neural network with a U-Net [253] architecture. In the optimization procedure, the categorical cross-entropy loss is minimized over 10^3 epochs using the Adam optimizer with a learning rate set to 10^{-3} . Next, in Figure 7.10 we present the preliminary

results obtained during the preparation of this thesis.

In particular, Figure 7.10b illustrates nine 64×64 generated images from the BloodMNIST data set, with pixel intensities for each RGB channel in the range $[0, 31]$. The FID between the real data set and the generated data set is equal to 250. In Figure 7.10d, we show the generation of nine 200×200 brain images from the BraTS2020 dataset, where the pixel intensities are in the range $[0, 63]$. The FID between the real and generated datasets is equal to 273.

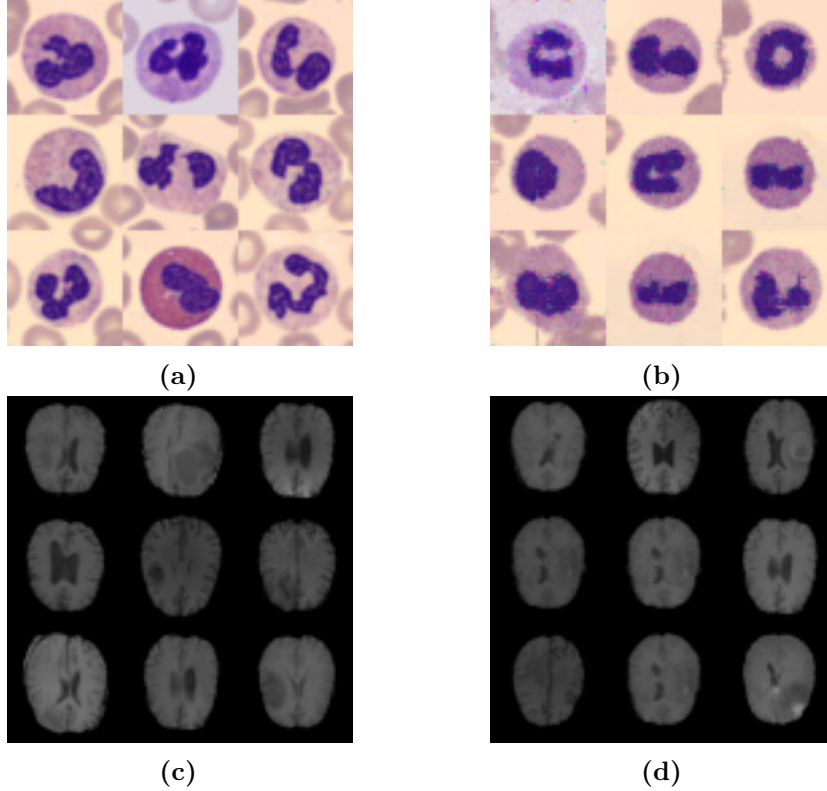


Figure 7.10. Illustration of real (a, c) and generated (b, d) medical images. The top row shows nine samples from the BloodMNIST dataset, while the bottom row depicts nine samples of 3D brain slices ($z=1$, layer=90) from the BraTS2020 dataset [252].

We conclude this section by observing that, although FID is a commonly used metric for the evaluation of synthetic images, its applicability to medical imaging remains an open question. In fact, due to its limitations in capturing clinically meaningful features [254], generated images can achieve high FID values while still being visually realistic and clinically relevant. In this regard, two recent works on quantum diffusion models for medical image generation [218, 219] have employed additional metrics to better assess the generation capabilities of their models: the Structural Similarity Index Measure (SSIM) [255] and the CLIP Maximum Mean Discrepancy (CMMD) [256]. SSIM evaluates the structural similarity between the generated and reference images, ensuring that clinically relevant anatomical features are preserved. CMMD is a more advanced statistical metric that measures the consistency between the distributions of real and generated images in a feature

space. A natural outlook is to explore and adopt these metrics, or alternative techniques tailored to medical applications, to more accurately assess the quality of the generated images produced by our model.

Chapter 8

Conclusions

Artificial Intelligence and quantum computing are two fascinating fields of contemporary science. In this thesis, we have investigated the intersections between these fields, in particular introducing also a novel generative paradigm that we have called the Quantum Diffusion Models. The manuscript is structured in seven chapters, each contributing to the theoretical and experimental foundations of the study.

In Chapter 1, we have introduced the fundamental concepts of artificial intelligence, machine learning, and deep learning, providing the essential tools for the construction and implementation of structured artificial neural networks such as multilayer perceptron (MLP). We also presented kernel methods that represent powerful techniques for classification tasks, capable of extending data representation beyond linear models. Chapter 2 was dedicated to generative artificial intelligence, where we have provided an overview of the most widely used models. In particular, we focused on diffusion models, which currently represent the state of the art in this domain, and inspired our generalization of them to the quantum domain. Next, in Chapter 3, we have introduced the main tools of quantum computing in a noiseless scenario, highlighting its differences and advantages over classical computing, such as the ability to exploit superposition and entanglement to represent and process information in ways inaccessible to classical systems. Nevertheless, current quantum devices consist of only a few hundreds of qubits and are inherently noisy. For this reason, in Chapter 4 we presented the formalism of open quantum systems, which provides a useful framework for modeling the dynamics of such devices. Chapter 5 has introduced the emerging field of quantum machine learning, a promising area of research that combines machine learning techniques with the unique features of quantum mechanics. We discussed the main approaches, strategies for encoding classical data into quantum states, the construction of quantum neural networks, and the use of quantum kernels.

In the last two chapters, the results obtained during my doctoral research have been presented. In particular, in Chapter 6, we initially analyzed four quantum representation techniques (Tensor Network Representation (TNR), Novel Enhanced Quantum Representation (NEQR), Flexible Representation of Quantum Images (FRQI), and Quantum Probability Image Encoding (QPIE)) for grayscale images, with the aim of qualitatively evaluating the degree of compression of classical information when these are encoded in quantum states. By computing Gram matrices,

we have observed that the FRQI and QPIE representations produce quantum states that are very close to each other, indicating high compression, whereas the NEQR representation shows the opposite behavior. The TNR encoding is in an intermediate scenario between the other three. Next, we have performed binary image classification via the Support Vector Machines (SVM) methods using quantum kernels based on these representations and compared their performance with the classical linear kernel on MNIST and Fashion-MNIST datasets. The results have shown that the mean accuracy of quantum kernels is comparable to that of the classical counterpart, but with a significant advantage in terms of computational resources, as quantum kernels require exponentially fewer resources to store images information. However, the challenge of an efficient data loading remains open, since current protocols for exact data encoding into an n -qubit state require $\mathcal{O}(2^n)$ quantum gates, often overcoming the computational cost of quantum algorithms. Future research directions could include extending the study to RGB images, comparing quantum kernels with other classical kernels, and evaluating the performance of quantum kernels on real quantum hardware where noise effects occur.

Chapter 7 constitutes the core of the manuscript. The chapter introduced the generalization of classical generative diffusion models to the quantum domain, presenting the novel paradigm of Quantum Diffusion Models (QDMs). In particular, we have formalized three scenarios within this framework: (1) a hybrid case with classical forward dynamics and quantum backward dynamics via parameterized quantum circuits; (2) a case with quantum forward dynamics based on quantum Markov chains or quantum stochastic equations and classical backward dynamics; and (3) a fully quantum scenario. For each case, we have provided numerical examples and simulations.

We then have analyzed how quantum and hybrid diffusion dynamics affect image generation, adopting the formalism of Quantum Stochastic Walks combined with categorical diffusion models. We demonstrated that a specific interplay between classical and quantum dynamics in the forward process yields statistically more robust models, capable of generating MNIST images with a lower Fréchet Inception Distance (FID) compared to purely classical dynamics.

Based on these results, we have proposed and implemented a hybrid quantum-classical diffusion model based on quantum walks, executed on real NISQ hardware. More precisely, we have exploited the intrinsic noise of the *ibm_brisbane* quantum device to generate MNIST images. The main advantage of our model is that it requires a maximum of $q + 1$ qubits to generate grayscale images with pixel intensity values in the range $[0, 2^q - 1]$ of arbitrary size. This feature makes it suitable for implementation on current quantum hardware, which ranges from a few dozen to several hundred qubits. In addition, this has enabled us to apply our model to medical imaging for the generation of realistic datasets. The preliminary results obtained at the time of writing this thesis include the generation of color images of size 64×64 from the BloodMNIST dataset, with pixel intensities for each RGB channel in the range $[0, 31]$, as well as the generation of grayscale brain MRI images of size 200×200 from the BraTS2020 dataset, where the intensity of each pixel is in the range $[0, 63]$. For these implementations, the *ibm_torino* quantum machine has been used.

An interesting perspective concerns the possible physical realization of our model

through the experimental implementation of quantum walks. Such an approach would allow the model to be applied directly to quantum data, without the need for classical embedding in the initial stage, followed by the use of quantum neural networks in the reverse process. This strategy could prove particularly fruitful in contexts such as quantum sensing, metrology, and chemistry.

Finally, we hope that our study may serve as a starting point for new scenarios in large-scale quantum generative AI where quantum noise is not mitigated or corrected but is instead harnessed as a valuable resource to assist AI applications.

Appendix A

Cross-validation

Cross-validation is a technique in machine learning and statistics that is used to assess the ability of a model to generalize to unseen data. Its goal is to provide a reliable estimate of the performance of the model in order to avoid the overfitting scenario and to ensure that the results are also valid for new data samples. In particular, the method involves dividing the data set into multiple subsets, known as *folds*, and repeatedly training and testing the model by rotating the fold used for validation. The most common technique is *K-fold cross-validation*, in which the data set is partitioned into K folds of the same size. An illustration of the procedure is shown in Figure A.1. At each iteration, one fold is used as the test set, while the model is trained on the remaining $K - 1$ folds. This process is repeated K times, ensuring that each fold serves exactly once as the test set. After all iterations, the performance metrics obtained from each fold are aggregated, typically by computing the average, to obtain a more robust estimate of the model accuracy. This approach is particularly advantageous when we have limited data, as it enables full utilization of the data set for both training and validation of results.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Figure A.1. Illustration of partitioning a dataset using the K-fold cross-validation procedure with $K = 5$. Each row represents one iteration, and each column corresponds to one fold.

Appendix B

Classical random walks and quantum walks

Here we provide a formal description of discrete-time classical random walks and quantum walks on a graph.

Classical random walks on a graph

A graph is formally defined as a pair $G = (V, E)$, where V denotes a finite non-empty set whose elements v_i , $i = 1, \dots, |V|$, are referred to as vertices, and E is a non-empty set of unordered pairs of vertices $e_{ij} = \{v_i, v_j\}$, called edges. The *order* and *size* of a graph correspond, respectively, to the cardinality $|V|$ and $|E|$ of the sets V and E . The number d_v of edges connected to the vertex v is called the *degree* of the vertex. A graph is said to be *undirected* if its edges do not have a direction; otherwise, it is said to be *directed*. The structure of a graph is characterized by its *adjacency matrix* A , which encodes the topology of the graph. The entries of A are defined as:

$$[A]_{i,j} = \begin{cases} 1, & \text{if } e_{ij} \in E, \\ 0, & \text{if } e_{ij} \notin E. \end{cases} \quad (\text{B.1})$$

In a discrete-time classical random walk (CRW) on an undirected graph G , at each step $t \in \mathbb{N}$ a walker jumps between two connected nodes v_i, v_j with some probability described by the stochastic transition matrix $S = [S]_{i,j}$ [257]. The matrix S is related to the adjacency matrix by the equation: $[S]_{i,j} = [A]_{i,j}/d_i$. Mathematically, a walker is represented by a discrete random variable X and its trajectory after the T time steps is the set $\{x_1, \dots, x_T\}$, where the value x_t , with $t = 1, \dots, T$, corresponds to the vertex occupied by the walker at time t . A CRW is a Markov process, i.e., the distribution at time $t+1$ depends only on the distribution at time t . Let $\mathbf{p}_t = \{p_i^{(t)}\}$ denote the probability distribution of the walker on the nodes v_i after t time steps. The evolution of the distribution at time $t+1$ is then expressed as:

$$\mathbf{p}_{t+1} = S\mathbf{p}_t. \quad (\text{B.2})$$

The distribution of a CRW converges to a limiting *stationary* solution:

$$\pi = \left\{ \frac{d_i}{2|V|} \right\}, \quad t \rightarrow \infty, \quad (\text{B.3})$$

which holds independently of the initial distribution $\mathbf{p}_0$. In the case of a d -regular graph, where all vertices share the same degree d , the stationary distribution becomes *uniform* across the nodes of the graph.

Discrete-Time quantum walks on a graph

In quantum information theory, quantum walks (QWs) were first introduced by [258] as the quantum counterparts of classical random walks. In classical random walks, the dynamics is purely stochastic at each time step, whereas QWs evolve deterministically, with stochasticity arising only from measurements. Moreover QWs involve peculiar quantum mechanical properties such as coherent superposition, entanglement, and quantum interference, resulting in a faster spread (ballistic for the quantum case, while diffusive for the classical one). In the following, we provide a formal description of discrete-time quantum walks on a graph.

Given a graph $G(V, E)$, let $\mathcal{H}_V$ be the Hilbert space spanned by the position states $\{|v\rangle : v = 1, \dots, |V|\}$, and let $\mathcal{H}_A$ be an auxiliary Hilbert space spanned by the coin states $\{|\downarrow\rangle, |\uparrow\rangle\}$. The total Hilbert space $\mathcal{H}$ associated with a QW is obtained by the tensor product between the auxiliary space and the position space:

$$\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_V. \quad (\text{B.4})$$

A general quantum state is of the form:

$$|\psi\rangle = |a\rangle \otimes |v\rangle, \quad |a\rangle \in \mathcal{H}_A \quad |v\rangle \in \mathcal{H}_V. \quad (\text{B.5})$$

The evolution of the quantum walker is governed by the unitary single time-step operator $\hat{U}$ acting on the total Hilbert space:

$$\hat{U} = \hat{S} \cdot (\hat{C} \otimes \hat{I}), \quad (\text{B.6})$$

where $\hat{I}$ and $\hat{C}$ are the identity and coin operator acting, respectively, on the position space and the auxiliary space. $\hat{S}$ is a shift operator that acts on the position space

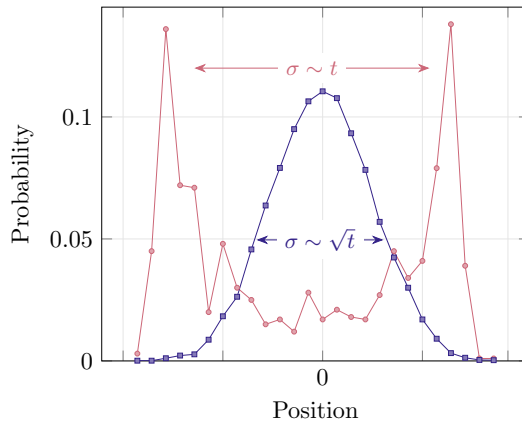


Figure B.1. Comparison between the Gaussian probability distribution (violet squares) obtained from a CRW and probability distribution of a QW with Hadamard coin (red circles).

and moves the walker from state $|v\rangle$ to state $|v+1\rangle$, or $|v-1\rangle$. Formally, the shift operator is as follows:

$$\hat{S} = \sum_v |\uparrow\rangle\langle\uparrow| \otimes |v+1\rangle\langle v| + |\downarrow\rangle\langle\downarrow| \otimes |v-1\rangle\langle v|. \quad (\text{B.7})$$

The coin operator $\hat{C}$ can be chosen in the family of unitary operators and its choice leads to symmetric (unbiased walk) or asymmetric (biased walk) distributions. A common choice for $\hat{C}$ is the Hadamard quantum gate which leads to an unbiased walk. The state of a quantum walk after t discrete time steps is given by:

$$|\psi(t)\rangle = \hat{U}^{\otimes t} |\psi(0)\rangle. \quad (\text{B.8})$$

In general, QWs do not converge to any limiting distribution in contrast to the classical ones: the limit $\lim_{t \rightarrow \infty} |\psi(t)\rangle$ does not exist due to the unitary evolution. Moreover, the interference effects lead a quantum walker to spread *quadratically* faster with respect to its classical counterpart. In fact, in the classical case after t time steps the expected distance from the origin is of the order $\sigma \sim \sqrt{t}$, while in the quantum case it is of the order $\sigma \sim t$, as shown in Figure B.1.

Appendix C

MNIST image generation

In this appendix, we provide additional results on image generation. In Figure C.1 we show the forward and generation processes for nine images of the digit “0” of the MNIST dataset, implemented with the IBM simulator *fake_brisbane*. Figure C.2 illustrates the generated images for the digits $1, 2, \dots, 9$ of MNIST obtained via different dynamics: a QSW dynamics with $\omega = 0.3$ simulated with QuTiP, and with a QW dynamics implemented on the *fake_brisbane* IBM simulator and on the real IBM quantum hardware *ibm_brisbane*. Finally, the real IBM QPU *ibm_brisbane* topology is shown in Fig. C.3.

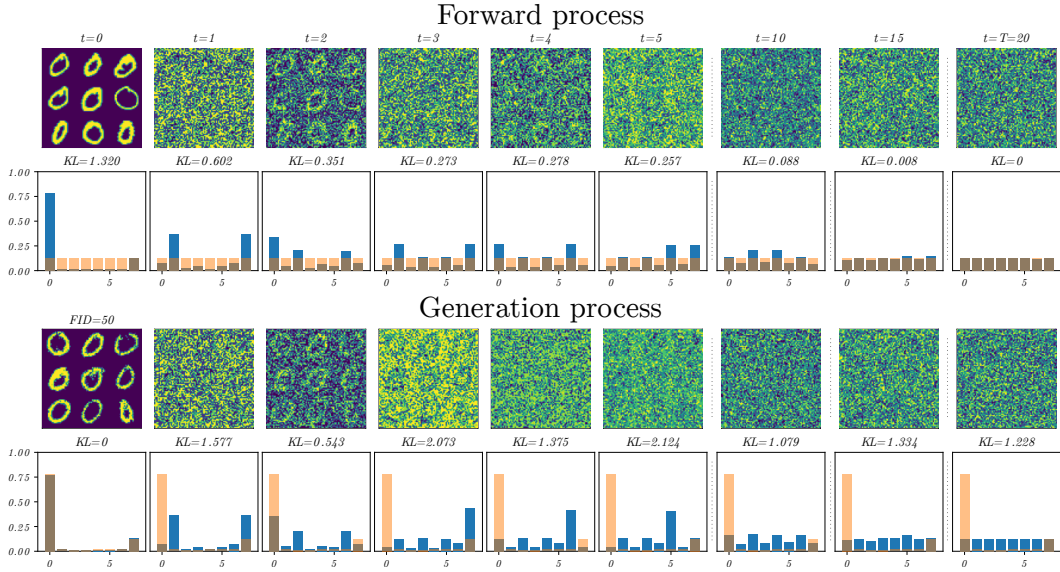


Figure C.1. Images generation using a noisy QW dynamics executed on the IBM simulator *fake_brisbane*. Evolution of nine samples of the real dataset in forward (first row) and nine generated samples in backward (third row). We also show the evolution of the pixel values distributions during forward (second row) reporting the KL divergence values compared to the final desired uniform distribution and the evolution of the same distribution during the backward (fourth row) comparing it with the one from the initial data set. Finally, the FID value between the original dataset and the generated one is reported for $t = 0$ [204].

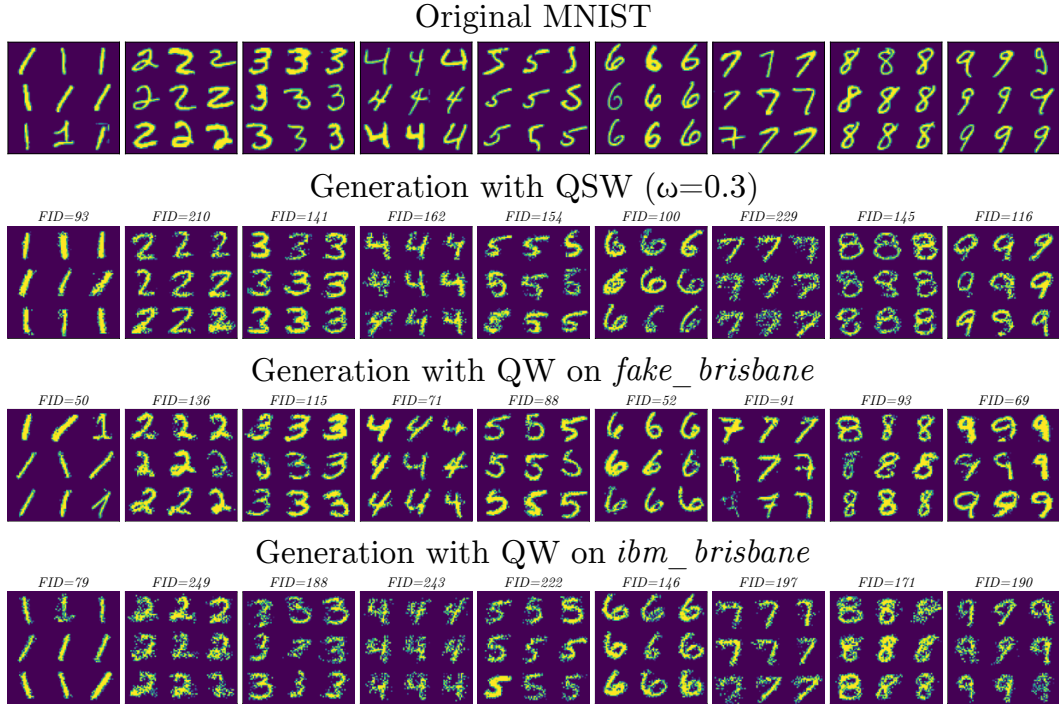


Figure C.2. Image generation with hybrid QSW dynamics and a QW with noise from a simulated and a real NISQ device for the MNIST digits from “1” to “9”. We illustrate nine different samples for each digit. In the first row, the samples are taken from the original MNIST dataset. In the second row, the samples are generated using a forward implementation with QSW dynamics simulated with $\omega = 0.3$. In the third and fourth rows, the samples are generated using a forward implemented with QWs and run on the *fake_brisbane* simulator and the real *ibm_brisbane* quantum machine, respectively. We also report the values of the FID metric between the original full training set and a generated dataset of the same size [204].

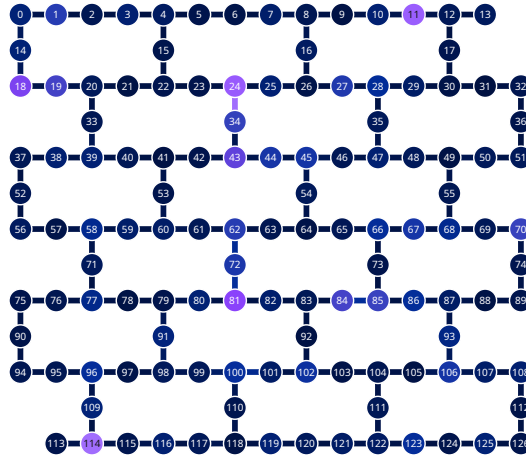


Figure C.3. Connectivity map of the real QPU of the *ibm_brisbane* quantum hardware (courtesy of IBM). Each circle denotes a qubit, while the lines indicate their connections. Colors represent readout errors (circles) and connection errors (lines): dark blue corresponds to low error rates, whereas purple indicates higher errors.

Bibliography

- [1] Marco Parigi, Stefano Martina, and Filippo Caruso. Back cover: Quantum-noise-driven generative diffusion models (adv. quantum technol. 12/2025). *Advanced Quantum Technologies*, 8(12):e70122, 2025.
- [2] Charles Babbage. On the mathematical powers of the calculating engine (1837). In Brian Randell, editor, *The Origins of Digital Computers: Selected Papers*, pages 19–54. Springer, 1973.
- [3] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [4] Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950.
- [5] Frank Rosenblatt. The perceptron—a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957. Available as PDF.
- [6] Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, March 1960.
- [7] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [9] Ian J. 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.
- [10] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.
- [11] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference*

- on *Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.
- [12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
 - [13] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.
 - [14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.
 - [15] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794, 2021.
 - [16] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.
 - [17] DALL-E. Official website. <https://openai.com/blog/dall-e>. Accessed: 2025-10-03.
 - [18] Stability AI: Stable Diffusion. Official website. <https://stability.ai/blog/stable-diffusion-public-release>. Accessed: 2025-10-03.
 - [19] Midjourney. Official website. <https://www.midjourney.com/home>. Accessed: 2025-10-03.
 - [20] Google DeepMind. Official website. <https://deepmind.google>. Accessed: 2025-12-13.
 - [21] Google DeepMind: Gemini Diffusion. Official website. <https://deepmind.google/models/gemini-diffusion>. Accessed: 2025-12-13.
 - [22] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980.
 - [23] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

- [24] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [25] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.
- [26] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.
- [27] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press, 2014.
- [28] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Springer, 2021.
- [29] Marco Parigi, Stefano Martina, and Filippo Caruso. Quantum-noise-driven generative diffusion models. *Advanced Quantum Technologies*, 8(12):2300401, 2025. The first preprint version was released on 23 August 2023 and is available at arXiv:2308.12013v1.
- [30] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [31] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
- [32] 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 (CVPR)*, pages 2818–2826, June 2016.
- [33] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.
- [34] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, and et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015.
- [35] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence. <https://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>, 1955. Unpublished manuscript.

- [36] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2021.
- [37] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [38] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2012.
- [39] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci U S A*, 79(8):2554–2558, 1982.
- [40] Christopher M. Bishop and Hugh Bishop. *Deep Learning*. Springer Cham, 1 edition, 2023.
- [41] Tom M. Mitchell. *Machine Learning*. McGraw-Hill International Editions: Computer Science Series. McGraw-Hill, illustrated edition, 1997.
- [42] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.
- [43] Fei-Fei Li and Ehsan Adeli. Lecture 13: Generative models. CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University., 2024.
- [44] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763. Communicated by Mr. Price in a letter to John Canton.
- [45] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, NY, 2 edition, 2009.
- [46] Stephen A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.
- [47] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pages 177–186. Springer, 2010.
- [48] Robert Kleinberg, Yuanzhi Li, and Yang Yuan. An alternative view: When does sgd escape local minima? In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2698–2707. PMLR, 2018.
- [49] A.B.J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.
- [50] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [51] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

- [52] Bernhard Schölkopf, Ralf Herbrich, and Alexander J. Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426. Springer, 2001.
- [53] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [54] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.
- [55] Joseph Weizenbaum. *Computer Power and Human Reason: From Judgment to Calculation*. W. H. Freeman and Company, 1976.
- [56] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [57] OpenAI. Official website. <https://openai.com>. Accessed: 2025-10-03.
- [58] OpenAI: ChatGPT. Official website. <https://openai.com/chatgpt>. Accessed: 2025-10-03.
- [59] Stability AI. Official website. <https://stability.ai>. Accessed: 2025-10-03.
- [60] David Kuo Chuen Lee, Chong Guan, Yinghui Yu, and Qinxu Ding. A comprehensive review of generative ai in finance. *FinTech*, 3(3):460–478, 2024.
- [61] Kang Zhang, Xin Yang, Yifei Wang, Yunfang Yu, Niu Huang, Gen Li, Xiaokun Li, Joseph C. Wu, and Shengyong Yang. Artificial intelligence in drug development. *Nature Medicine*, 31(1):45–59, Jan 2025.
- [62] Marium M. Raza, Kaushik P. Venkatesh, and Joseph C. Kvedar. Generative ai and large language models in healthcare: pathways to implementation. *npj Digital Medicine*, 7(1):62, Mar 2024.
- [63] Maximos Kaliakatsos-Papakostas, Andreas Floros, and Michael N. Vrahatis. Chapter 13 - artificial intelligence methods for music generation: a review and future perspectives. In Xin-She Yang, editor, *Nature-Inspired Computation and Swarm Intelligence*, pages 217–245. Academic Press, 2020.
- [64] Mridul Sharma and Mandeep Kaur. A review of deepfake technology: An emerging ai threat. In G. Ranganathan, Xavier Fernando, Fuqian Shi, and Youssef El Alloui, editors, *Soft Computing for Security Applications*, pages 605–619, Singapore, 2022. Springer Singapore.
- [65] Mark Coeckelbergh. Artificial intelligence: Some ethical issues and regulatory challenges. *Technology and Regulation*, 2019:31–34, May 2019.
- [66] Yingzhou Lu, Lulu Chen, Yuanyuan Zhang, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, Tianfan Fu, and Wenqi Wei. Machine learning for synthetic data generation: A review, 2025.

- [67] Mauro Giuffrè and Dennis L. Shung. Harnessing the power of synthetic data in healthcare: innovation, application, and privacy. *npj Digital Medicine*, 6(186), 2023.
- [68] Mandeep Goyal and Qusay H Mahmoud. A systematic review of synthetic data generation techniques using generative ai. *Electronics*, 13(17):3509, 2024.
- [69] André Bauer, Simon Trapp, Michael Stenger, Robert Leppich, Samuel Kounev, Mark Leznik, Kyle Chard, and Ian Foster. Comprehensive exploration of synthetic data generation: A survey. *arXiv preprint arXiv:2401.02524*, 2024.
- [70] Matteo Paiano, Stefano Martina, Carlotta Giannelli, and Filippo Caruso. Transfer learning with generative models for object detection on limited datasets. *Machine Learning: Science and Technology*, 5(3):035041, 2024.
- [71] Sergey I Nikolenko et al. *Synthetic data for deep learning*, volume 174. Springer, 2021.
- [72] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.
- [73] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12454–12465. Curran Associates, Inc., 2021.
- [74] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc., 2021.
- [75] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.
- [76] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.
- [77] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, page 204, Tel Aviv, Israel, 2019.
- [78] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.
- [79] Benjamin Schumacher. Quantum coding. *Phys. Rev. A*, 51:2738–2747, Apr 1995.

- [80] John Clarke and Frank Wilhelm. Superconducting quantum bits. *Nature*, 453:1031–42, 07 2008.
- [81] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Rev. Mod. Phys.*, 79:135–174, Jan 2007.
- [82] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74:4091–4094, May 1995.
- [83] Chetan Nayak, Steven H. Simon, Ady Stern, Michael Freedman, and Sankar Das Sarma. Non-abelian anyons and topological quantum computation. *Rev. Mod. Phys.*, 80:1083–1159, Sep 2008.
- [84] Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Number 27 in International Series of Monographs on Physics. Oxford university press, 1981.
- [85] Ettore Majorana. Teoria relativistica di particelle con momento intrinseco arbitrario. *Il Nuovo Cimento*, 9:43–50, 1932.
- [86] A. S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973.
- [87] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117, 1985.
- [88] Josh Schneider and Ian Smalley. What is a qpu (quantum processing unit)? <https://www.ibm.com/think/topics/qpu>, 2024. Accessed: 2025-08-01.
- [89] E. Schrödinger. Discussion of probability relations between separated systems. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31(4):555–563, 1935.
- [90] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, Mar 1993.
- [91] Charles H. Bennett and Stephen J. Wiesner. Communication via one- and two-particle operators on einstein-podolsky-rosen states. *Phys. Rev. Lett.*, 69:2881–2884, Nov 1992.
- [92] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995.
- [93] P Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. A new universal and fault-tolerant quantum basis. *Information Processing Letters*, 75(3):101–107, 2000.

- [94] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.
- [95] Yaoyun Shi. Both toffoli and controlled-not need little help to do universal quantum computation. *arXiv preprint quant-ph/0205115*, 2002.
- [96] A. Yu. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.
- [97] Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm, 2005.
- [98] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000.
- [99] Heinz-Peter Breuer and Francesco Petruccione. *The theory of open quantum systems*. Oxford University Press, 2002.
- [100] Michel H Devoret, Andreas Wallraff, and John M Martinis. Superconducting qubits: A short review. *e-print arXiv:cond-mat/0411174.*, 2004.
- [101] John Clarke and Frank K Wilhelm. Superconducting quantum bits. *Nature*, 453(7198):1031–1042, 2008.
- [102] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Phys. Rev. A*, 76:042319, Oct 2007.
- [103] J. A. Schreier, A. A. Houck, Jens Koch, D. I. Schuster, B. R. Johnson, J. M. Chow, J. M. Gambetta, J. Majer, L. Frunzio, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Suppressing charge noise decoherence in superconducting charge qubits. *Phys. Rev. B*, 77:180502, May 2008.
- [104] Ibm quantum experience. <https://quantum-computing.ibm.com>.
- [105] Rigetti computing. <https://www.rigetti.com>.
- [106] Google quantum ai. <https://quantumai.google>.
- [107] Xanadu quantum technologies. <https://xanadu.ai>.
- [108] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Rev. Mod. Phys.*, 79:135–174, Jan 2007.
- [109] Ionq. <https://ionq.com>.
- [110] Stewart Allen, Jungsang Kim, David L. Moehring, and Christopher R. Monroe. Reconfigurable and programmable ion trap quantum computer. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–3, 2017.

- [111] C. Monroe and J. Kim. Scaling the ion trap quantum processor. *Science*, 339(6124):1164–1169, 2013.
- [112] Pasqal. <https://www.pasqal.com>.
- [113] Quera. <https://www.quera.com>.
- [114] Loïc Henriët, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020.
- [115] John Preskill. Fault-tolerant quantum computation. In *Introduction to quantum computation and information*, pages 213–269. World Scientific, 1998.
- [116] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 55:900–911, Feb 1997.
- [117] Earl Campbell. A series of fast-paced advances in quantum error correction. *Nature Reviews Physics*, 6(3):160–161, Mar 2024.
- [118] Rajeev Acharya, AI Google Quantum, and Collaborators. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, Feb 2025.
- [119] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, feb 2016.
- [120] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermann Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.*, 94:015004, Feb 2022.
- [121] Sreetama Das, Jingfu Zhang, Stefano Martina, Dieter Suter, and Filippo Caruso. Quantum pattern recognition on real quantum processing units. *Quantum Machine Intelligence*, 5(1):16, Apr 2023.
- [122] Alexander Geng, Ali Moghiseh, Claudia Redenbach, and Katja Schladitz. A hybrid quantum image edge detector for the nisq era. *Quantum Machine Intelligence*, 4(2):15, Jun 2022.
- [123] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):4213, Jul 2014.
- [124] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, Sep 2017.

- [125] Guillermo González-García, Rahul Trivedi, and J. Ignacio Cirac. Error propagation in nisq devices for solving classical optimization problems. *PRX Quantum*, 3:040326, Dec 2022.
- [126] Kun Zhang, Pooja Rao, Kwangmin Yu, Hyunkyung Lim, and Vladimir Korepin. Implementation of efficient quantum search algorithms on nisq computers. *Quantum Information Processing*, 20(7):233, Jul 2021.
- [127] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors, 2018.
- [128] Xiaosi Xu, Jinzhao Sun, Suguru Endo, Ying Li, Simon C Benjamin, and Xiao Yuan. Variational algorithms for linear algebra. *Science Bulletin*, 66(21):2181–2188, 2021.
- [129] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. Near-term quantum algorithms for linear systems of equations, 2019.
- [130] Dorit Aharonov, Jordan Cotler, and Xiao-Liang Qi. Quantum algorithmic measurement. *Nature communications*, 13(1):887, 2022.
- [131] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 2022.
- [132] A. Uhlmann. The “transition probability” in the state space of a $*$ -algebra. *Reports on Mathematical Physics*, 9(2):273–279, 1976.
- [133] A. Kossakowski. On quantum statistical mechanics of non-hamiltonian systems. *Reports on Mathematical Physics*, 3(4):247–274, 1972.
- [134] G. Lindblad. On the generators of quantum dynamical semigroups. *Communications in Mathematical Physics*, 48(2):119–130, Jun 1976.
- [135] Vittorio Gorini, Andrzej Kossakowski, and E. C. G. Sudarshan. Completely positive dynamical semigroups of n -level systems. *Journal of Mathematical Physics*, 17(5):821–825, 05 1976.
- [136] K. Kraus. *States, Effects, and Operations: Fundamental Notions of Quantum Theory*, volume 190 of *Lecture Notes in Physics*. Springer-Verlag, Berlin, 1983.
- [137] M.-D. Choi. Completely positive linear maps on complex matrices. *Linear Algebra and Its Applications*, 10:285–290, 1975.
- [138] Ángel Rivas and Susana F. Huelga. *Open Quantum Systems*. SpringerBriefs in Physics. Springer Berlin, Heidelberg, 1 edition, 2012.
- [139] Filippo Caruso, Vittorio Giovannetti, Cosmo Lupo, and Stefano Mancini. Quantum channels and memory effects. *Rev. Mod. Phys.*, 86:1203–1259, 2014.
- [140] Daniel A. Lidar. Lecture notes on the theory of open quantum systems, 2020.

- [141] Dan Ventura and Tony Martinez. A quantum associative memory based on grover’s algorithm. In *Artificial Neural Nets and Genetic Algorithms*, pages 22–27. Springer Vienna, 1999.
- [142] Hartmut Neven, Vasil S. Denchev, Garrett Rose, and William G. Macready. Training a large scale classifier with the quantum adiabatic algorithm. *arXiv preprint arXiv:0912.0779*, 2009.
- [143] D-Wave Quantum Inc. Quantum computing | d-wave. <https://www.dwavequantum.com/learn/quantum-computing/>, 2025. Accessed: 2025-11-09.
- [144] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.
- [145] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.
- [146] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.
- [147] Ettore Canonici, Stefano Martina, Riccardo Mengoni, Daniele Ottaviani, and Filippo Caruso. Machine learning based noise characterization and correction on neutral atoms nisc devices. *Advanced Quantum Technologies*, 7(1):2300192, 2024.
- [148] Sreetama Das and Filippo Caruso. Permutation-equivariant quantum convolutional neural networks. *Quantum Science and Technology*, 10(1):015030, 2024.
- [149] Giacomo Cappiello and Filippo Caruso. Quantum AI for alzheimer’s disease early screening. *Neurocomputing*, 647:130565, 2025.
- [150] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Carsten Blank, Keri McKiernan, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.
- [151] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.
- [152] Google AI Quantum team. Cirq: An open source framework for nisc algorithms. <https://quantumai.google/cirq>.
- [153] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J. Martinez, Jae Hyun Yoo, Sergei V. Isakov, Paul Massey, Masoud Yuezhen Niu, Ramin Halavati, Evan Peters, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

- [154] Marco Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J Coles. Challenges and opportunities in quantum machine learning. *Nature computational science*, 2(9):567–576, 2022.
- [155] Giovanni Acampora, Andris Ambainis, Natalia Ares, Leonardo Banchi, Pallavi Bhardwaj, Daniele Binosi, G Andrew D Briggs, Tommaso Calarco, Vedran Dunjko, Jens Eisert, et al. Quantum computing and artificial intelligence: status and perspectives. *arXiv preprint arXiv:2505.23860*, 2025.
- [156] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *arXiv preprint arXiv:2008.08605*, 2020.
- [157] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature communications*, 12(1):2631, 2021.
- [158] M Cerezo, A Arrasmith, R Babbush, SC Benjamin, S Endo, K Fujii, JR McClean, K Mitarai, X Yuan, L Cincio, and PJ Coles. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.
- [159] Marcello Benedetti, Edward Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.
- [160] E.C. Behrman, L.R. Nash, J.E. Steck, V.G. Chandrashekar, and S.R. Skinner. Simulations of quantum neural networks. *Information Sciences*, 128(3–4):257–269, 2000.
- [161] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.
- [162] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633*, 2018.
- [163] Marco Cerezo, Kunal Sharma, Andrew Arrasmith, Lukasz Cincio, and Patrick J Coles. Variational quantum state eigensolver. *npj Quantum Information*, 8:113, September 2022.
- [164] Leo Zhou, Shengtao Wang, Soonwon Choi, Hannes Pichler Kim, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, June 2020.
- [165] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms. *arXiv preprint arXiv:1701.01450*, 2017.
- [166] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *arXiv preprint arXiv:1803.00745*, 2018.

- [167] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. Estimating the gradient and higher-order derivatives on quantum hardware. *arXiv preprint arXiv:2008.06517*, 2020.
- [168] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.
- [169] Ryan Sweke, Frederik Wilde, Johannes Jakob Meyer, Maria Schuld, Philipp K. Fährmann, Benjamin Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314, 2020.
- [170] Martín Larocca et al. Diagnosing barren plateaus with tools from quantum optimal control. *arXiv preprint arXiv:2105.14377*, May 2021.
- [171] Michael Ragone et al. A lie algebraic theory of barren plateaus for deep parameterized quantum circuits. *Nature Communications*, 15, Aug 2024.
- [172] Riccardo Mengoni and Alessandra Di Pierro. Kernel methods in quantum machine learning. *Quantum Machine Intelligence*, 1(3):65–71, 2019.
- [173] Maria Schuld. Quantum machine learning models are kernel methods. *arXiv preprint*, 2021.
- [174] Marco Parigi, Mehran Khosrojerdi, Filippo Caruso, and Leonardo Banchi. Analysis of quantum image representations for supervised classification. *AVS Quantum Science*, 8(1):013801, 2026.
- [175] Jose I. Latorre. Image compression and entanglement, 2005.
- [176] Phuc Q. Le, Fangyan Dong, and Kaoru Hirota. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing*, 10(1):63–84, Feb 2011.
- [177] Yi Zhang, Kai Lu, Yinghui Gao, and Mo Wang. Neqr: a novel enhanced quantum representation of digital images. *Quantum Information Processing*, 12(8):2833–2860, Aug 2013.
- [178] Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, Wenqiang Zheng, Jianzhong Li, Meisheng Zhao, Xinhua Peng, and Dieter Suter. Quantum image processing and its application to edge detection: Theory and experiment. *Phys. Rev. X*, 7:031041, Sep 2017.
- [179] Salvador E. Venegas-Andraca. *Discrete Quantum Walks and Quantum Image Processing*. Dphil thesis, University of Oxford, 2005.
- [180] Fei Yan, Abdullah M. Iliyasu, and Salvador E. Venegas-Andraca. A survey of quantum image representations. *Quantum Information Processing*, 15(1):1–35, Jan 2016.

- [181] Marina Lisnichenko and Stanislav Protasov. Quantum image representation: a review. *Quantum Machine Intelligence*, 5(1):2, Dec 2022.
- [182] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. Ieee, 2015.
- [183] Leonardo Banchi, Jason Pereira, and Stefano Pirandola. Generalization in quantum machine learning: A quantum information standpoint. *PRX Quantum*, 2(4):040321, 2021.
- [184] Leonardo Banchi, Jason Luke Pereira, Sharu Theresa Jose, and Osvaldo Simeone. Statistical complexity of quantum learning. *Advanced Quantum Technologies*, page 2300311, 2024.
- [185] Donovan Slabbert and Francesco Petruccione. Classical-quantum approach to image classification: Autoencoders and quantum svms. *AVS Quantum Science*, 7(2):023804, 06 2025.
- [186] Norhan Nasr, Ahmed Younes, and Ashraf Elsayed. Efficient representations of digital images on quantum computers. *Multimedia Tools and Applications*, 80(25):34019–34034, Oct 2021.
- [187] Alexander Geng, Ali Moghiseh, Claudia Redenbach, and Katja Schladitz. Quantum image processing on real superconducting and trapped-ion based quantum computers. *tm - Technisches Messen*, 90(7-8):445–454, 2023.
- [188] E. M. Stoudenmire and David J. Schwab. Supervised learning with tensor networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16*, page 4806–4814, Red Hook, NY, USA, 2016. Curran Associates Inc.
- [189] Li Xu, Xiao-yu Zhang, Ming Li, and Shu-qian Shen. Quantum classifiers with a trainable kernel. *Phys. Rev. Appl.*, 21:054056, May 2024.
- [190] Carsten Blank, Daniel K Park, June-Koo Kevin Rhee, and Francesco Petruccione. Quantum classifier with tailored quantum kernel. *npj Quantum Information*, 6(1):41, 2020.
- [191] Nathan X Kodama, Alex Bocharov, and Marcus P da Silva. Image classification by combining quantum kernel learning and tensor networks. *Physical Review A*, 111(1):012630, 2025.
- [192] Aswiga RV, Sridevi S, and Indira B. Leveraging quantum kernel support vector machine for breast cancer diagnosis from digital breast tomosynthesis images. *Quantum Machine Intelligence*, 6(2):40, 2024.
- [193] Shivani Mahashakti Pillay, Ilya Sinayskiy, Edgar Jembere, and Francesco Petruccione. A multi-class quantum kernel-based classifier. *Advanced Quantum Technologies*, 7(1):2300249, 2024.

- [194] Mehran Khosrojerdi, Jason L Pereira, Alessandro Cuccoli, and Leonardo Banchi. Learning to classify quantum phases of matter with a few measurements. *Quantum Science and Technology*, 10(2):025006, jan 2025.
- [195] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Info. Comput.*, 7(5):401–430, July 2007.
- [196] Manuel S Rudolph, Jacob Miller, Danial Motlagh, Jing Chen, Atithi Acharya, and Alejandro Perdomo-Ortiz. Synergistic pretraining of parametrized quantum circuits via tensor networks. *Nature Communications*, 14(1):8367, 2023.
- [197] Phuc Le, Abdullah Ilyasu, Fangyan Dong, and Kaoru Hirota. Efficient color transformations on quantum images. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 15:698–706, 05 2011.
- [198] Phuc Le, Abdullah Ilyasu, Dong Fangyan, and Hirota Kaoru. Fast geometric transformations on quantum images. *IAENG International Journal of Applied Mathematics*, 40, 08 2010.
- [199] Sahel Ashhab. Quantum state preparation protocol for encoding classical data into the amplitudes of a quantum information processing register’s wave function. *Phys. Rev. Res.*, 4:013091, Feb 2022.
- [200] Mikko Möttönen, Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa. Transformation of quantum states using uniformly controlled rotations. *Quantum Info. Comput.*, 5(6):467–473, September 2005.
- [201] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [202] Gary Bradski. The opencv library. *Dr. Dobb’s Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.
- [203] Johnnie Gray and Stefanos Kourtis. Hyper-optimized tensor network contraction. *Quantum*, 5:410, 2021.
- [204] Marco Parigi, Stefano Martina, Francesco Aldo Venturelli, and Filippo Caruso. Physics-inspired generative ai models via real hardware-based noisy quantum diffusion. *arXiv preprint arXiv:2505.22193*, 2025.
- [205] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, Hossein Sadeghi, and Mohammad H Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1):014001, 2018.
- [206] Gaoyuan Wang, Jonathan Warrell, Prashant S Emani, and Mark Gerstein. Quantum variational autoencoder utilizing regularized mixed-state latent representations. *Physical Review A*, 111(4):042416, 2025.
- [207] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Phys. Rev. Lett.*, 121:040502, Jul 2018.

- [208] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Phys. Rev. A*, 98:012324, Jul 2018.
- [209] Tuan A Ngo, Tuyen Nguyen, and Truong Cong Thang. A survey of recent advances in quantum generative adversarial networks. *Electronics*, 12(4):856, 2023.
- [210] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):103, 2019.
- [211] Paolo Braccia, Leonardo Banchi, and Filippo Caruso. Quantum noise sensing by generating fake noise. *Physical Review Applied*, 17(2):024002, 2022.
- [212] Elie Bermot, Christa Zoufal, Michele Grossi, Julian Schuhmacher, Francesco Tacchino, Sofia Vallecorsa, and Ivano Tavernelli. Quantum generative adversarial networks for anomaly detection in high energy physics. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pages 331–341. IEEE, 2023.
- [213] Haoran Ma, Liao Ye, Xiaoqing Guo, Fanjie Ruan, Zichao Zhao, Maohui Li, Yuehai Wang, and Jianyi Yang. Quantum generative adversarial networks in a silicon photonic chip with maximum expressibility. *Advanced Quantum Technologies*, page 2400171, 2024.
- [214] Mohamed Hibat-Allah, Marta Mauri, Juan Carrasquilla, and Alejandro Perdomo-Ortiz. A framework for demonstrating practical quantum advantage: comparing quantum against classical generative models. *Communications Physics*, 7(1):68, Feb 2024.
- [215] Filippo Caruso. Universally optimal noisy quantum walks on complex networks. *New Journal of Physics*, 16(5):055015, 2014.
- [216] Filippo Caruso, Andrea Crespi, Anna Gabriella Ciriolo, Fabio Sciarrino, and Roberto Osellame. Fast escape of a quantum walker from an integrated photonic maze. *Nature Communications*, 7(1):11682, 2016.
- [217] Nicola Dalla Pozza, Lorenzo Buffoni, Stefano Martina, and Filippo Caruso. Quantum reinforcement learning: the maze problem. *Quantum Machine Intelligence*, 4(1):11, 2022.
- [218] Chi-Sheng Chen, Wei An Hou, Hsiang-Wei Hu, and Zhen-Sheng Cai. Quantum generative models for image generation: Insights from mnist and medmnist. *arXiv preprint arXiv:2504.00034*, 2025.
- [219] Kübra Yeter-Aydeniz, Nora M Bauer, Pranay Jain, and Max Masnick. Hybrid quantum-classical latent diffusion models for medical image generation. *arXiv preprint arXiv:2508.09903*, 2025.
- [220] Chuangtao Chen, Qinglin Zhao, MengChu Zhou, Zhimin He, Zhili Sun, and Haozhen Situ. Quantum generative diffusion model: A fully quantum-mechanical model for generating quantum state ensemble, 2024.

- [221] Gino Kwun, Bingzhi Zhang, and Quntao Zhuang. Mixed-state quantum denoising diffusion probabilistic model. *Physical Review A*, 111(3):032610, 2025.
- [222] Haitao Huang, Chuangtao Chen, and Qinglin Zhao. Continuous-variable quantum diffusion model for state generation and restoration. *arXiv preprint arXiv:2506.19270*, 2025.
- [223] Shivalee R. K. Shah and Mayank Vatsa. Enhancing quantum diffusion models with pairwise bell state entanglement. In Apostolos Antonacopoulos, Subhasis Chaudhuri, Rama Chellappa, Cheng-Lin Liu, Saumik Bhattacharya, and Umapada Pal, editors, *Pattern Recognition*, pages 347–361, Cham, 2025. Springer Nature Switzerland.
- [224] Qin-Sheng Zhu, Geng Chen, Lian-Hui Yu, Xiaodong Xing, and Xiao-Yu Li. Channel-constrained markovian quantum diffusion model from open system perspective. *arXiv preprint arXiv:2511.12221*, 2025.
- [225] Mariia Baidachna, Rey Guadarrama, Gopal Ramesh Dahale, Tom Magorsch, Isabel Pedraza, Konstantin T Matchev, Katia Matcheva, Kyoungchul Kong, and Sergei Gleyzer. Quantum diffusion model for quark and gluon jet generation. *arXiv preprint arXiv:2412.21082*, 2024.
- [226] Hoang-Quan Nguyen, Xuan Bac Nguyen, Samuel Yen-Chi Chen, Hugh Churchill, Nicholas Borys, Samee U Khan, and Khoa Luu. Diffusion-inspired quantum noise mitigation in parameterized quantum circuits. *Quantum Machine Intelligence*, 7(1):55, 2025.
- [227] Ruhan Wang, Ye Wang, Jing Liu, and Toshiaki Koike-Akino. Quantum diffusion models for few-shot learning. In *2025 IEEE International Conference on AI and Data Analytics (ICAD)*, pages 1–8. IEEE, 2025.
- [228] Bingzhi Zhang, Peng Xu, Xiaohui Chen, and Quntao Zhuang. Generative quantum machine learning via denoising diffusion probabilistic models. *Phys. Rev. Lett.*, 132:100602, Mar 2024.
- [229] Xinyu Liu, Jingze Zhuang, Wanda Hou, and Yi-Zhuang You. Measurement-based quantum diffusion models. *arXiv preprint arXiv:2508.08799*, 2025.
- [230] Michael Kölle, Gerhard Stenzel, Jonas Stein, Sebastian Zielinski, Björn Ommer, and Claudia Linnhoff-Popien. Quantum denoising diffusion models, 2024.
- [231] Francesca DeFalco, Andrea Ceschini, Alessandro Sebastianelli, Bertrand LeSaux, and Massimo Panella. Quantum latent diffusion models. *Quantum Machine Intelligence*, 6(2):85, Nov 2024.
- [232] Andrea Cacioppo, Lorenzo Colantonio, Simone Bordoni, and Stefano Giagu. Quantum diffusion models for quantum data learning in high-energy physics. *QTML 2024 Conference*, 2024.

- [233] Francesca De Falco, Andrea Ceschini, Alessandro Sebastianelli, Bertrand Le Saux, and Massimo Panella. Quantum hybrid diffusion models for image synthesis. *KI - Künstliche Intelligenz*, Aug 2024.
- [234] Xuefen Zhang and Chuangtao Chen. Parameter-efficient quantum denoising diffusion probabilistic models with temporal encoding. *Future Generation Computer Systems*, page 107981, 2025.
- [235] Nikolai Theien Kivijervi. Quantum diffusion model. Master’s thesis, University of Oslo, 2024.
- [236] Chuangtao Chen, Qinglin Zhao, MengChu Zhou, Dusit Niyato, Zhimin He, and Haozhen Situ. Overcoming dimensional factorization limits in discrete diffusion models through quantum joint distribution learning. *arXiv preprint arXiv:2505.05151*, 2025.
- [237] Fernando Perez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pages 1666–1670, 2008.
- [238] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Expressive power of parametrized quantum circuits. *Phys. Rev. Res.*, 2:033125, Jul 2020.
- [239] Zhan Yu, Qiu hao Chen, Yuling Jiao, Yinan Li, Xiliang Lu, Xin Wang, and Jerry Zhijian Yang. Provable advantage of parameterized quantum circuit in function approximation. *e-print arXiv:2310.07528*, 2023.
- [240] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Information-theoretic bounds on quantum advantage in machine learning. *Phys. Rev. Lett.*, 126:190505, May 2021.
- [241] Stanley Gudder. Quantum Markov chains. *Journal of Mathematical Physics*, 49(7), 07 2008. 072105.
- [242] E Brian Davies and John T Lewis. An operational approach to quantum probability. *Communications in Mathematical Physics*, 17(3):239–260, 1970.
- [243] Howard M Wiseman. Quantum trajectories and quantum measurement theory. *Quantum and Semiclassical Optics: Journal of the European Optical Society Part B*, 8(1):205, 1996.
- [244] James D Whitfield, César A Rodríguez-Rosario, and Alán Aspuru-Guzik. Quantum stochastic walks: A generalization of classical random walks and quantum walks. *Physical Review A—Atomic, Molecular, and Optical Physics*, 81(2):022323, 2010.
- [245] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

- [246] Martin Krzywinski and Naomi Altman. Visualizing samples with box plots. *Nature Methods*, 11(2):119–120, Feb 2014.
- [247] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.
- [248] J.R. Johansson, P.D. Nation, and Franco Nori. Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, 2013.
- [249] Luca Razzoli, Gabriele Cenedese, Maria Bondani, and Giuliano Benenti. Efficient implementation of discrete-time quantum walks on quantum computers. *Entropy*, 26(4), 2024.
- [250] Andrea Cacioppo, Lorenzo Colantonio, Simone Bordoni, and Stefano Giagu. Quantum diffusion models, 2023.
- [251] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021.
- [252] Stefano Martina, Francesco Aldo Venturelli, Marco Parigi, Filippo Caruso, Alba Cervera-Lierta, and Miguel A. González Ballester. Manuscript in preparation, 2025.
- [253] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
- [254] McKell Woodland, Austin Castelo, Mais Al Taie, Jessica Albuquerque Marques Silva, Mohamed Eltaher, Frank Mohn, Alexander Shieh, Suprateek Kundu, Joshua P. Yung, Ankit B. Patel, and Kristy K. Brock. Feature Extraction for Generative Medical Imaging Evaluation: New Evidence Against an Evolving Trend . In *proceedings of Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, volume LNCS 15012. Springer Nature Switzerland, October 2024.
- [255] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [256] Sadeep Jayasumana et al. Rethinking fid: Towards a better evaluation metric for image generation. *arXiv preprint arXiv:2401.09603*, 2024.
- [257] George H. Weiss and Robert J. Rubin. *Random Walks: Theory and Selected Applications*, pages 363–505. John Wiley & Sons, Ltd, 1982.

-
- [258] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Phys. Rev. A*, 48:1687–1690, Aug 1993.



Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA



UNIVERSITÀ
DEGLI STUDI
FIRENZE

La borsa di dottorato è stata finanziata con risorse dell'Unione Europea – *NextGeneration EU 2022*
Piano Nazionale di Ripresa e Resilienza Missione 4 – Componente 2 – Investimento 1.3 –
Progetto *PE4-NQSTI* – CUP *B83C22004940006*